HUMBOLDT-UNIVERSITÄT ZU BERLIN
INSTITUT FÜR INFORMATIK
D-10099 BERLIN

Einführung

in

Graphen und Algorithmen

Th. Emden-Weinert,

S. Hougardy,

B. Kreuter,

H.J. Prömel,

A. Steger

Vorläufige Fassung

© *13. September 1996*

Inhaltsverzeichnis

Vorwort **4**

Notation **5**

1 Einführung **8**
1.1 Grundlegende Begriffe 8
1.1.1 Partite Graphen 12
1.1.2 Operationen auf Graphen 13
1.1.3 Bäume, Artikulationen, Schnitte 14
1.1.4 Zufällige Graphen 18
1.2 Die Komplexität von Algorithmen 21
1.2.1 Entscheidbarkeit 21
1.2.2 Die Klasse P oder: zugängliche Probleme 22
1.3 Algorithmische Prinzipien 26
1.3.1 Breiten- und Tiefensuche 26
1.3.2 Dynamisches Programmieren: das kürzeste-Wege-Problem 31
1.3.3 Greedy: minimale aufspannende Bäume 34
1.4 Die Klasse NP oder: polynomielle Verifizierbarkeit 36
1.4.1 NP-Vollständigkeit 38
1.4.2 NP-schwere Probleme und Optimierung 46
1.5 Zum Umgang mit NP-schweren Problemen: Algorithmen für INDEPENDENT SET 49

2 Netzwerkflüsse und Zusammenhang **59**
2.1 Der Markierungsalgorithmus von FORD und FULKERSON 59
2.2 Der Satz von Menger nebst Folgerungen 64
2.3 2-Zusammenhang, Blockstruktur und Tiefensuche 67
2.4 ⋆ Die Zusammenhangszahlen 74

3 Durchlaufbarkeit **81**
3.1 Eulersche Graphen 81
3.2 Hamiltonsche Graphen 86

4 Matching **94**
4.1 Die Sätze von BERGE und GALLAI 94
4.2 Matching in bipartiten Graphen 99
4.3 Matching in allgemeinen Graphen 114

5 Färbung **128**
5.1 Die chromatische Zahl . . . 128
5.2 Greedy-Färbung und der Satz von Brooks . . . 130
5.3 Chromatische Zahl und Taillenweite . . . 136
5.4 Die Komplexität des Färbungsproblems . . . 139
5.5 Färbungsalgorithmen . . . 146
5.6 Kantenfärbung: der Satz von Vizing . . . 153
5.7 ⋆ Kantenfärben in bipartiten Graphen . . . 160

6 Topologische Graphentheorie **167**
6.1 Planare Graphen . . . 167
6.1.1 Ebene Graphen . . . 168
6.1.2 Die Eulersche Formel . . . 171
6.1.3 Die Sätze von Kuratowski und Wagner . . . 174
6.1.4 Dualität bei planaren Graphen . . . 180
6.2 Ein Einbettungsalgorithmus . . . 187
6.3 Die 4-Farben-Vermutung . . . 188
6.4 ⋆ Graphen höheren Geschlechts . . . 204
6.5 Der 4-Farben-Satz und Komplexitätsfragen . . . 213
6.6 ⋆ Kreuzungszahl, Dicke, Arborizität und Buchdicke . . . 218

7 Perfekte Graphen **225**
7.1 Einleitung . . . 225
7.2 Chordale Graphen . . . 227
7.3 Vergleichbarkeitsgraphen . . . 236
7.4 Intervallgraphen . . . 241
7.5 Graphen mit perfekter Ordnung . . . 242
7.6 Im Umfeld der Perfekte-Graphen-Vermutung . . . 243
7.7 Optimierung auf perfekten Graphen . . . 247

8 Separatoren **248**
8.1 Einleitung . . . 248
8.2 Separatoren in planaren Graphen . . . 249
8.3 INDEPENDENT SET auf planaren Graphen . . . 254
8.4 Separatoren für andere Graphenklassen . . . 258
8.5 Cliquenseparatoren . . . 260

9 Baumweite **270**
9.1 Baumweite und k-Bäume . . . 270
9.2 Optimierung auf Graphen beschränkter Baumweite . . . 275
9.3 Randomisierung I: Subgraph Isomorphismus . . . 278

10 Approximationsalgorithmen **280**
10.1 Einleitung . . . 280
10.2 Knotenüberdeckung . . . 282
10.3 Steiner-Bäume . . . 285
10.4 Das Traveling Salesman Problem . . . 287

10.5 Das Zentrumsproblem 289
10.6 Randomisierung II: MAX CUT und MAX SAT 292
10.7 Lokale Verbesserung 305
10.8 Ein PAS für PLANAR INDEPENDENT SET 308
10.9 GRAPH COLORING und INDEPENDENT SET 311

11 Nicht-Approximierbarkeit **322**
11.1 MAX-SNP und PCP 323
11.2 CLIQUE 323
11.3 GRAPH COLORING 323

12 Überdeckungsprobleme **324**
12.1 Cliquenüberdeckung und Schnittgraph-Darstellung 324
12.2 Erkennung von Linegraphen 330
12.3 Primal-dual Approximationsalgorithmen 331
12.3.1 Der primal-dual Algorithmus der Linearen Optimierung 331
12.3.2 HITTING SET, NODE COVER und DOMINATING SET 332
12.3.3 SHORTEST PATH 336
12.3.4 Das GENERALIZED STEINER TREE Problem 338
12.3.5 SURVIVABLE NETWORK DESIGN mit Mehrfachkanten 345
12.4 Approximation von SET COVER und DOMINATING SET 347

13 Zufällige Graphen und probabilistische Algorithmen **353**
13.1 Die Cliquenzahl 353
13.2 Die chromatische Zahl 360

14 Ein Blick in die extremale und asymptotische Graphentheorie **369**
14.1 Die Sätze von Turán und von Erdős und Stone 369
14.2 Fast alle dreiecksfreien Graphen sind bipartit 374
14.3 Mehr über dreiecksfreie Graphen 379

A Anhang **383**
A.1 Grundbegriffe der Wahrscheinlichkeitstheorie 383
A.2 Hinweise und Lösungen zu ausgewählten Übungen 386
A.3 Die Grenze zwischen P und NP-vollständig 398
A.4 Übersicht über Graphenparameter 399
A.5 Weiterführende Literatur 400

Literaturverzeichnis **403**

Vorwort

Der algorithmische Aspekt der Graphentheorie ist in den vergangenen Jahren gerade im Spannungsfeld zwischen Mathematik und Informatik stark in den Vordergrund getreten. Um dieser Entwicklung Rechnung zu tragen, entstand bereits 1989 anläßlich eines Vorlesungszyklus, den ich seinerzeit an der Universität Bonn gehalten habe, die Idee, ein Skriptum zu schreiben. Im Laufe der letzten 7 Jahre sind nun verschiedene Teile, teils nach Vorlesungen, teils für Vorlesungen, teils auch unabhängig von Vorlesungen, entstanden. Die jetzt vorliegende Fassung ist durch das große Engagement und den Einsatz von Herrn Emden-Weinert entstanden. Ihm gebührt dafür mein ganz besonderer Dank.

Diese Fassung ist noch unvollständig, unausgewogen und bestimmt auch noch mit Fehlern behaftet, und es ist noch ein weiter Weg bis das, was intendiert ist, auch in geschriebener Form vorliegt. Dennoch haben wir uns entschlossen, sie in dieser Form bereits allgemein zugänglich zu machen, nicht zuletzt in der Hoffnung, zahlreiche Kommentare, Anregungen und Verbesserungsvorschläge zu erhalten.

Ich möchte an dieser Stelle noch einmal allen Mitarbeitern, die im Laufe der Jahre sowohl an der Universität Bonn als auch an der Humboldt-Universität zu Berlin durch Ideen, Diskussionen und viel Schreibarbeit am Werden dieser inzwischen fast 500 Seiten mitgewirkt haben, ganz herzlich danken.

Berlin, im September 1996 Hans Jürgen Prömel

Notation

Referenzen. Wo es uns möglich war, haben wir Sätzen, Korollaren u.ä. die Namen der Urheber mit Jahresangabe hinzugefügt. Der Leser findet dann entsprechende Verweise auf Originalliteratur im Literturverzeichnis unter einem Kürzel, das sich aus den ersten drei Buchstaben des Namens des Autors bzw. den Initialen bei mehreren Autoren sowie den letzten zwei Ziffern der Jahreszahl zusammensetzt. Das Literaturverzeichnis ist nach diesen Kürzeln alphabetisch sortiert (ohne Unterscheidung von Groß- und Kleinbuchstaben).

Bezeichnungen und Abkürzungen. Um das Studium der aktuellen Literatur nicht unnötig zu erschweren, haben wir Bezeichnungen für Entscheidungsprobleme und Symbole für Graphenparameter (siehe Tabelle auf Seite 399) so gewählt, wie sie zumeist (wenn auch nicht durchgängig) im angelsächsischen Sprachraum verwendet werden, siehe z.B. [GJ79, Lov79]. Leider gibt es jedoch auch dort keine einheitliche Nomenklatur.

Mathematische Termini. $\lfloor x \rfloor$ ist definiert als die größte ganze Zahl $\leq x$, $\lceil x \rceil$ als die kleinste ganze Zahl $\geq x$.
Wenn wir von der *Parität* einer natürlichen Zahl sprechen, heben wir darauf ab, ob die Zahl gerade oder ungerade ist.
$|x|$ bezeichnet den Absolutbetrag der Zahl x.
„log“ bezeichnet stets den Logarithmus zur Basis 2, „ln“ den natürlichen Logarithmus.
Für zwei Zahlen $a, b \in \mathbb{N}$ bezeichnet $a \bmod b$ den ganzzahligen Rest von a bei Division durch b, d.h. es ist $a \bmod b := s$, falls a die Darstellung $a = r \cdot b + s$ hat für $r, s \in \mathbb{N}_0$ mit $0 \leq s < b$.
$x \equiv y \pmod{a}$ bedeutet, daß die Gleichung für die Reste modulo a gilt.

$\emptyset$, $\mathbb{N}$, $\mathbb{Q}$ und $\mathbb{R}$ bezeichnen die leere Menge, die Menge der natürlichen, der rationalen bzw. der reellen Zahlen; $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$ und $\mathbb{R}_+ := \{x \in \mathbb{R} : x > 0\}$.
Die *symmetrische Differenz* $A \Delta B$ zweier Mengen A und B ist definiert als

$$A \Delta B \;:=\; (A \cup B) \setminus (A \cap B) \;=\; (A \setminus B) \cup (B \setminus A).$$

Ferner schreiben wir kurz $M - e := M \setminus \{e\}$ und $M + e := M \cup \{e\}$. Die Schreibweise $\dot{\bigcup}_{i \in I} M_i$ deutet an, daß die Mengen M_i der Vereinigung paarweise disjunkt sind. $|M|$ bezeichnet die *Kardinalität* oder Mächtigkeit (d.h. die Anzahl der Elemente) der Menge M. Mit $2^M := \{T : T \subseteq M\}$ bezeichnen wir die *Potenzmenge* (d.h. die Menge aller Teilmengen) der Menge M – beachte, es ist $|2^M| = 2^{|M|}$. Die Fakultätsfunktion ist für $n \in \mathbb{N}$ definiert als $n! := \prod_{k=1}^{n} k = n \cdot (n-1) \cdot \ldots \cdot 1$. Für $k, n \in \mathbb{N}$ ist der Binomialkoeffizient definiert als $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. Für ein $k \in \mathbb{N}_0$ sei $\binom{M}{k} := \{T \subseteq M : |T| = k\}$ die Menge der k-elementigen Teilmengen von M – beachte, es ist $|\binom{M}{k}| = \binom{|M|}{k}$ und $2^{|M|} = \sum_{k=0}^{|M|} \binom{|M|}{k}$. Sprechen wir von *zwei* Elementen x und y einer Menge, so meinen wir stets zwei verschiedene.

Für eine endliche Menge M und eine Funktion $f : M \to \mathbb{R}$ bezeichnet $argmin\ \{f(m) : m \in M\}$ ein $m^* \in M$ mit $f(m^*) = \min\{f(m) : m \in M\}$ (entsprechend $argmax$).

Wie im Englischen bezeichnen auch wir der Kürze und Eindeutigkeit zuliebe eine Teilmenge S einer Menge M als *maximal* bezüglich einer Eigenschaft P, wenn keine echte Obermenge von S mehr die betrachtete Eigenschaft hat („inklusionsmaximal“ oder lokal maximal), und als *maximum*, wenn S eine größte Teilmenge von M mit der betrachteten Eigenschaft ist („kardinalitätsmaximal“ oder global maximal).

Eine Abbildung $f : M \to N$ zwischen Mengen M und N heißt *injektiv*, falls für $x, y \in M$ gilt: $f(x) = f(y) \Rightarrow x = y$; Sie heißt *surjektiv*, falls gilt: $\forall y \in N\ \exists x \in M\ (f(x) = y)$. Eine Abbildung heißt *bijektiv* genau dann, wenn sie injektiv und surjektiv ist. Wir werden diese Begriffe hauptsächlich bei Zählargumenten verwenden: Ist $f : M \to N$ injektiv, so gilt offenbar $|N| \geq |M|$, ist $f : M \to N$ surjektiv, so folgt $|N| \leq |M|$, und ist $f : M \to N$ bijektiv, so gilt $|N| = |M|$.

Eine *Permutation* von $\{1, \ldots, n\}$ ist eine bijektive Abbildung von $\{1, \ldots, n\}$ nach $\{1, \ldots, n\}$. Die Permutationen von $\{1, \ldots, n\}$ bilden die symmetrische Gruppe $\mathcal{S}_n$ der Ordnung $n!$. Eine (0,1)-Matrix ist eine Matrix mit Einträgen aus $\{0, 1\}$. Eine $n \times n$-Matrix $P = (p_{ij})_{ij}$ heißt *Permutationsmatrix*, falls eine Permutation $\sigma \in S_n$ existiert, so daß

$$p_{ij} = \begin{cases} 1 & \text{falls } \sigma(i) = j, \\ 0 & \text{sonst.} \end{cases}$$

Eine *Äquivalenzrelation* auf einer Menge M ist eine zweistellige Relation „$\sim$“ (d.h. eine Teilmenge $R \subseteq M \times M$ mit der Interpretation $(x, y) \in R \Leftrightarrow x \sim y$), für die gilt:

(i) Reflexivität: $x \sim x$ für alle $x \in M$;

(ii) Symmetrie: $x \sim y \ \Rightarrow\ y \sim x$;

(iii) Transitivität: $x \sim y \ \wedge\ y \sim z \ \Rightarrow\ x \sim z$;

Eine Äquivalenzrelation auf einer Menge M partitioniert also M in die sogenannten *Äquivalenzklassen* von „$\sim$“, deren Elemente jeweils paarweise untereinander in Relation stehen. Umgekehrt definiert jede Partition von M eine Äquivalenzrelation.

Eine Aussage A heißt *notwendig* bzw. *hinreichend* für eine Aussage B, falls gilt $B \Rightarrow A$ bzw. $A \Rightarrow B$. Die Aussage $A \wedge B$ ist wahr genau dann, wenn A und B wahr sind, die Aussage $A \vee B$ ist wahr genau dann, wenn mindestens eine der beiden Aussagen wahr ist.

Die Landauschen Symbole. Die Landauschen Symbole dienen dazu, das asymptotische Wachstumsverhalten von Funktionen zu vergleichen. Seien $g, f : \mathbb{N} \to \mathbb{R}$ zwei reellwertige Funktionen. Dann heißt

$$\begin{array}{lcll}
f(n) & = & \mathcal{O}(g(n)) & :\Leftrightarrow \quad \exists C \in \mathbb{R}^+ \, \exists n_0 \in \mathbb{N} : \, |f(n)| \leq C|g(n)| \, \forall n \geq n_0, \\
f(n) & = & o(g(n)) & :\Leftrightarrow \quad \forall c \in \mathbb{R}^+ \, \exists n_0 \in \mathbb{N} : \, |f(n)| \leq c|g(n)| \, \forall n \geq n_0, \\
f(n) & = & \Omega(g(n)) & :\Leftrightarrow \quad \exists c \in \mathbb{R}^+ \, \exists n_0 \in \mathbb{N} : \, |f(n)| \geq c|g(n)| \, \forall n \geq n_0, \\
f(n) & = & \omega(g(n)) & :\Leftrightarrow \quad \forall C \in \mathbb{R}^+ \, \exists n_0 \in \mathbb{N} : \, |f(n)| \geq C|g(n)| \, \forall n \geq n_0, \\
f(n) & = & \Theta(g(n)) & :\Leftrightarrow \quad \exists c, C \in \mathbb{R}^+ \, \exists n_0 \in \mathbb{N} : \, c|g(n)| \leq |f(n)| \leq C|g(n)| \, \forall n \geq n_0, \\
f(n) & \sim & g(n) & :\Leftrightarrow \quad f(n) = (1 + o(1))g(n).
\end{array}$$

Beispiele. $f(n) = o(1)$ besagt gerade, daß f Nullfolge ist. Die Wachstumsordnung eines Polynoms ist durch den führenden Term bestimmt: $a_0 + a_1 n + \cdots + a_k n^k = \Theta(n^k)$.

Man beachte, daß eine „Gleichung“ wie $f(n) = \mathcal{O}(g(n))$ nur von links nach rechts gelesen werden kann. Das Gleichheitszeichen ist hier vielmehr als ein „$\in$“ zu lesen, wobei $\mathcal{O}(g(n))$ als die Menge der Funktionen von derselben Größenordnung wie $g(n)$ zu definieren wäre, siehe [Knu76, Bra85].

Die Landauschen Symbole erfüllen folgende Eigenschaften:

1. $f(n) = \mathcal{O}(f(n))$;
2. $\mathcal{O}(\mathcal{O}(f(n))) = \mathcal{O}(f(n))$;
3. $c \cdot \mathcal{O}(f(n)) = \mathcal{O}(f(n))$;
4. $\mathcal{O}(f(n)) \cdot \mathcal{O}(g(n)) = \mathcal{O}(f(n) \cdot g(n))$;
5. $\mathcal{O}(f(n) \cdot g(n)) = f(n) \cdot \mathcal{O}(g(n))$;
6. $o(f(n)) + o(f(n)) = o(f(n))$;
7. $o(f(n) \cdot g(n)) = f(n) \cdot o(g(n))$;
8. $f(n) \sim g(n) \;\Rightarrow\; f(n) = \Theta(g(n))$ (aber nicht umgekehrt).

Übungen.

a) $\binom{n}{k} = \Theta(n^k)$; für festes $k \in \mathbb{N}$.

b) $\binom{n}{2} \sim \frac{1}{2}n^2$;

c) $n + \log n = \Theta(n)$;

d) $(\log n)^k = o(n^\epsilon)$ für alle $\epsilon > 0$ und $k \in \mathbb{N}$;

e) $n^k = o(\epsilon^n)$ für alle $k \in \mathbb{N}$ und $\epsilon > 1$;

f) Für jedes feste $k \in \mathbb{N}_0$ gilt: $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$.

Kapitel 1

Einführung

1.1 Grundlegende Begriffe

Ein (endlicher) *Graph* ist ein Tupel $G = (V, E)$, wobei V eine endliche Menge (o.B.d.A. einfach $\{1, \ldots, n\}$) und $E \subseteq \binom{V}{2}$ eine Teilmenge der zweielementigen Teilmengen von V ist.[1] Die Elemente $v \in V$ heißen die *Knoten* (auch Ecken, engl. vertices oder nodes), die Elemente aus $e \in E$ die *Kanten* (engl. edges) von G. Ein besonderer Reiz der Graphentheorie ist geht davon aus, daß sich ihre Objekte leicht graphisch veranschaulichen lassen (man blättere mal eben durch das Buch). Die Anzahl der Knoten eines Graphen G wird als die *Ordnung* von G bezeichnet, die wir durchweg mit $n := n(G) := |V|$ notieren. Die Anzahl seiner Kanten, in Zeichen $m := m(G) := |E|$, heißt auch die *Größe* von G. Ein Graph heißt *gerade* bzw. *ungerade* je nach Parität seiner Knotenmenge. Für eine Kante $\{u, v\} \in E$ werden die Knoten u und v ihre *Endpunkte* genannt. Das *Komplement* $(V, \binom{V}{2} \setminus E)$ eines Graphen notieren wir mit $\overline{G}$. Ein Graph $(V, \emptyset)$ ohne Kanten heißt auch *leer* oder *trivial*. Sein Komplement, der Graph $K_n := (V, \binom{V}{2})$, heißt *vollständig*; bei n Knoten hat er $m = \binom{n}{2}$ Kanten.

Offenbar gibt es $2^{\binom{n}{2}}$ viele verschiedene Möglichkeiten, Kanten zwischen n Knoten zu setzen oder nicht zu setzen; es gibt also gerade ebenso viele verschiedene Graphen auf n (numerierten oder markierten, d.h. unterschiedenen) Knoten. Viele dieser Graphen unterscheiden sich jedoch lediglich in der Numerierung ihrer Knotenmenge: Zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ heißen *isomorph*, in Zeichen $G_1 \cong G_2$, falls es eine Bijektion $f : V_1 \to V_2$ gibt, so daß gilt: $\{u, v\} \in E_1 \Leftrightarrow \{f(u), f(v)\} \in E_2$.

Ein Knoten $v \in V$ und eine Kante $e \in E$ in einem Graphen $G = (V, E)$ *inzidieren* oder *überdecken* einander, falls $v \in e$. Zwei Knoten $u, v \in V$ heißen *adjazent*, *verbunden* oder *benachbart*, falls $\{u, v\} \in E$. Die *Nachbarschaft* eines Knotens $v \in V$ ist die Menge $\Gamma(v) := \{u \in V \mid \{u, v\} \in E\}$ der *Nachbarn* von v. Der *Grad* $d(v) := |\Gamma(v)|$ eines Knotens $v \in V$ zählt die Kanten, die in dem Graphen mit v inzidieren. Wir schreiben $d_G(v)$, um zu betonen, daß sich der Grad auf den Graphen G bezieht. Offenbar gilt, da jede Kante

[1] Es werden hier fast ausschließlich nur sogenannte *einfache* Graphen betrachtet, d.h. Graphen ohne Schlingen (Kanten, deren Endpunkte zusammenfallen) und ohne mehrfache (parallele) Kanten zwischen zwei Knoten. Gerichtete Graphen treten nur in Abschnitt 2.1 auf.

an ihren beiden Endpunkten einen Beitrag von 1 zu deren Grad liefert:

$$\sum_{v \in V} d(v) \;=\; 2m. \tag{1.1}$$

Daß auf einem Empfang gerade viele Gäste ungerade vielen anderen Gästen die Hand schütteln ist die Knobelversion der leichten

Übung 1.1.1 *Ein Graph hat eine gerade Anzahl von Knoten ungeraden Grades.*

Ein Graph G heißt *r-regulär*, falls für ein $r \in \mathbb{N}_0$ gilt: $d(v) = r$ für alle $v \in V$. Ein Knoten $v \in V$ mit $d(v) = 0$ heißt *isoliert*. In einem leeren Graphen sind also alle Knoten isoliert, während der vollständige Graph K_n $(n-1)-$regulär ist. Ein 3-regulärer Graph heißt auch *kubisch*. Der *maximale Grad* in einem Graphen wird mit $\Delta(G) := \max\{d(v) : \; v \in V\}$, der *minimale* mit $\delta(G) := \min\{d(v) : \; v \in V\}$ notiert.

Adjazenzmatrix. Sei $G = (V, E)$ ein Graph. Dann heißt die folgende symmetrische $n \times n$ Matrix $A = A(G) = (a_{ij})$, $1 \le i, j \le n$, mit

$$a_{ij} := \begin{cases} 1 & \text{falls } \{i,j\} \in E, \\ 0 & \text{sonst} \end{cases}$$

die *Adjazenzmatrix* von G. Die i-te Zeilen- oder Spaltensumme von A ist also gerade $d(v_i)$; die Anzahl der Einsen in A ist $\sum_{v \in V} d(v) = 2m$. Wir beobachten ferner, daß zwei Graphen G_1 und G_2 der Ordnung n genau dann isomorph sind, wenn es eine $n \times n$ Permutationsmatrix P gibt, so daß für die zugehörigen Adjazenzmatrizen gilt: $A_1 = P^T A_2 P$.

Subgraphen. Ein Graph $H = (V_H, E_H)$ heißt *(schwacher) Subgraph* eines Graphen $G = (V_G, E_G)$, in Zeichen $H \subseteq G$, falls eine injektive Abbildung $f : V_H \to V_G$ existiert, so daß gilt $\{u, v\} \in E_H \Rightarrow \{f(u), f(v)\} \in E_G$. Insbesondere ist also jeder Graph H mit $V_H \subseteq V_G$ und $E_H \subseteq \binom{V_H}{2} \cap E_G$ Subgraph von G. Ein Subgraph H eines Graphen G heißt durch V_H *induziert*, falls er alle Kanten von G enthält, die beide Endpunkte in V_H haben, d.h. falls $E_H = \{\{u, v\} \in E_G : \; u, v \in V_H\} = \binom{V_H}{2} \cap E_G$. Den von einer Knotenmenge $S \subseteq V$ in einem Graphen G induzierten Subgraphen bezeichnen wir mit $G[S]$. Ein Subgraph H eines Graphen G heißt *spannend*, falls $V(H) = V(G)$.

Eine Knotenmenge $C \subseteq V$ heißt *Clique* in einem Graphen G, falls der von C induzierte Subgraph $G[C]$ vollständig ist. Eine Knotenmenge $S \subseteq V$ heißt *unabhängig* oder *stabil*, falls keine zwei Knoten aus S benachbart sind, d.h. falls der von S in G induzierte Graph $G[S]$ leer ist. Die Größe einer kardinalitätsmaximalen Clique in einem Graphen G heißt die *Cliquenzahl* $\omega(G)$, die Größe einer kardinalitätsmaximalen unabhängigen Knotenmenge die Stabilitäts- oder *Unabhängigkeitszahl* $\alpha(G)$. Da eine Knotenmenge $C \subseteq V$ offenbar eine Clique in einem Graphen G ist genau dann, wenn sie im Komplementgraphen $\overline{G}$ stabil ist, gilt also $\omega(G) = \alpha(\overline{G})$. Für den vollständigen Graphen K_n auf n Knoten gilt beispielsweise $\omega(K_n) = n$ und $\alpha(K_n) = 1$. Sei G ein Graph und v ein Knoten in einer maximum Clique C von G. Wegen $|\Gamma(v)| \le \Delta(G)$ folgt

$$1 \;\le\; \omega(G) \;\le\; \Delta(G) + 1. \tag{1.2}$$

Wenn dagegen $S \subseteq V$ eine (inklusions-) maximale stabile Menge ist, dann ist jeder Knoten in $V \setminus S$ zu einem Knoten aus S adjazent (denn sonst könnte man ihn zu S hinzufügen

und erhielte eine noch größere stabile Menge) und daher $V \subseteq S \cup \bigcup_{s \in S} \Gamma(s)$. Es folgt $n \leq |S| + |S|\Delta(G)$ oder

$$\frac{n}{\Delta(G)+1} \leq |S| \leq \alpha(G) \leq n. \tag{1.3}$$

Gleichheit gilt in (1.2) bzw. (1.3) jeweils z.B. beim leeren bzw. beim vollständigen Graphen.

Wege, Pfade, Zusammenhang. Ein *Weg* W (engl. walk) ist eine Folge $v_0, v_1, \ldots, v_\ell$ von (nicht notwendig verschiedenen) Knoten $v_i \in V$, $0 \leq i \leq \ell$, so daß für $i = 0, \ldots, \ell - 1$ gilt: $\{v_i, v_{i+1}\} \in E$. Seine *Länge* ist die Anzahl ℓ der Kanten, die er enthält. Ein Weg $W = (v_0, v_1, \ldots, v_\ell)$, bei dem (falls $\ell \geq 1$) alle Knoten v_i, $0 \leq i \leq \ell$, paarweise verschieden sind, heißt *Pfad* (engl. path). Ein Pfad auf $n \in \mathbb{N}$ Knoten wird mit P_n notiert – beachte: er hat die Länge $n - 1$. Der Pfad P_1, der nur aus einem einzigen Knoten besteht, heißt trivial. Der Knoten v_0 heißt der Anfangsknoten des Pfades $P = (v_0, v_1, \ldots, v_\ell)$ und v_ℓ sein Endknoten; alle anderen Knoten heißen die inneren Knoten des Pfades P. Ein $u-v$–Pfad ist ein Pfad mit Anfangsknoten u und Endknoten v (mit $v \neq u$ falls $\ell \geq 1$).

Übung 1.1.2 *Jeder $u-v$–Weg ($u \neq v$) enthält einen $u-v$–Pfad.*
Ein kürzester $u-v$–Weg ist ein $u-v$–Pfad.

Mit $dist(u, v)$ bezeichnen wir den *Abstand* (die Distanz) zweier Knoten $u, v \in V$ im Graphen G, wobei $dist(u, v)$ definiert ist als die Länge eines kürzesten $u-v$–Pfades in G – falls es keinen $u-v$–Pfad in G gibt, so sei $dist(u, v) := \infty$.[2] Ein Graph G heißt *zusammenhängend*, wenn zwischen je zwei Knoten $u, v \in V$ ein $u-v$–Weg in G existiert. Die (Knoten-) maximalen zusammenhängenden Subgraphen eines Graphen heißen seine *(Zusammenhangs-)Komponenten*. Offenbar gehört jeder Knoten zu einer Komponente; verschiedene Komponenten wiederum sind knotendisjunkt. $c(G)$ bezeichne die Anzahl der Komponenten eines Graphen G. Es ist also $c(G) = 1$ genau dann, wenn G zusammenhängt. Die Relation $u \sim v \Leftrightarrow dist(u, v) < \infty$ bildet offensichtlich eine Äquivalenzrelation auf V, ihre Äquivalenzklassen sind gerade die Zusammenhangskomponenten von G.

Zykeln und Kreise. Ein Weg $W = v_0, v_1, \ldots, v_\ell$ in einem Graphen heißt geschlossen oder *Zykel*, falls $v_0 = v_l$. Wie wir in Abschnitt 3.1 sehen werden, sind die Zykeln gerade die Subgraphen von G, in denen jeder Knoten geraden Grad hat. Ein kreuzungsfreier Zykel (bei dem die Knoten $v_0, \ldots, v_{\ell-1}$ also paarweise verschieden sind) heißt *Kreis*. Der Kreis der Länge n wird mit C_n notiert. Der Kreis $C_3 \cong K_3$ heißt auch *Dreieck*. Intuitiv sollte ein dichter Graph (d.h. ein Graph mit $m/n \gg 1$) „lange“ Kreise enthalten; die folgende Proposition macht dies präzise.

Proposition 1.1.3 (Dirac 1952a) *Jeder Graph G vom Minimalgrad $\delta(G) \geq 2$ enthält einen Kreis der Länge mindestens $\delta(G) + 1$. Ein Graph von der Dichte $d := m/n \geq 2$ enthält einen Kreis der Länge $> d$.*

Beweis. Sei $\{v_0, v_1\} \in E$. Verlängere den Pfad $P = (v_0, v_1)$ so lange über v_1 hinaus, bis die Nachbarn des Endknotens v_ℓ des erhaltenen Pfades $P = (v_0, \ldots, v_\ell)$ allesamt wieder auf P liegen. Sei v_k der Nachbar von v_ℓ mit dem kleinsten Index $k \in \{0, \ldots, \ell - 1\}$. Wegen $\delta(G) \geq 2$ gilt $k < \ell - 1$, und die Knoten $v_k, \ldots, v_\ell$ bilden einen Kreis, der offenbar mindestens $\delta(G) + 1$ viele Knoten enthält.

[2] Die Distanzabbildung $dist : V \times V \to \mathbb{N} \cup \{0, \infty\}$ definiert eine Metrik auf der Menge der Knoten eines Graphen [es gilt $dist(u, v) = 0 \Leftrightarrow u = v$ (Definitheit), $dist(u, v) = dist(v, u)$ für alle $u, v \in V$ (Symmetrie) sowie $dist(u, w) \leq dist(u, v) + dist(v, w)$ für alle $u, v, w \in V$ (Dreiecksungleichung)].

Zum Beweis der zweiten Aussage entferne rekursiv Knoten vom Grad $< d$ aus dem Graphen. Da dabei weniger als $nd = m$ Kanten entfernt werden, verbleibt ein nicht-leerer Graph vom Minimalgrad mindestens d. □

Die *Taillenweite* $g(G)$ (von engl. girth) eines Graphen G bezeichnet die Länge eines kürzesten Kreises in G, bzw. ist als ∞ definiert, falls G kreisfrei ist. Ein „sehr dichter" Graph sollte intuitiv auch „sehr kurze" Kreise enthalten; in diesem Zusammenhang gilt folgendes:

Proposition 1.1.4 *Ein Graph G mit $\delta(G) > n/2$ hat Taillenweite: $g(G) = 3$.*

Beweis. Es ist zu zeigen, daß G unter dieser Voraussetzung ein Dreieck besitzt. Sei $\{u, v\} \in E$. Wegen $d(u) > n/2$ und $d(v) > n/2$ können $\Gamma(u)$ und $\Gamma(v)$ nicht disjunkt sein. Also gibt es einen Knoten $w_0 \in V - u - v$, der sowohl mit u also auch mit v verbunden ist. (Wir haben also sogar gezeigt, daß jede Kante in einem Dreieck liegt). □

Schon der C_4 zeigt, daß die Aussage bestmöglich hinsichtlich der Bedingung an den Minimalgrad ist.

Übungen

Übung 1.1.5 *Die Isomorphie „$\cong$" von Graphen ist eine Äquivalenzrelation auf der Menge aller Graphen.*

Übung 1.1.6 *Ein Graph heißt* selbstkomplementär, *falls $G \cong \overline{G}$. Man finde den (einzigen) selbstkomplementären Graphen auf 4 und die beiden auf 5 Knoten.*

Übung 1.1.7 **a)** *Jeder Graph ($n \geq 2$) hat zwei Knoten desselben Grades.*
b) *Bestimme Graphen auf $n \geq 2$ Knoten, die $n-1$ verschiedene Grade haben.*

Übung 1.1.8 **a)** *Ein kubischer Graph hat gerade viele Knoten.*
b) *Es gibt kubische Graphen der Ordnung n für jedes gerade $n \geq 4$.*

Übung 1.1.9 [AKS80] *Jeder Graph $G = (V, E)$ enthält einen Knoten (einen sogenannten „groupie") $v \in V$ mit $t(v) := \frac{1}{d(v)} \sum_{u \in \Gamma(v)} d(u) \geq \frac{2m}{n}$.*

Übung 1.1.10 *Der Eintrag a_{ij}^k der k-ten Potenz A^k der Adjazenzmatrix eines Graphen G entspricht der Anzahl der $i-j-$Wege in G der Länge k.*

Übung 1.1.11 *Sei H eine Komponente des Graphen G und $v \in V$ ein Knoten, der in H liegt. Dann ist H die Vereinigung aller Wege in G, die in v starten.*

Der *Durchmesser* $diam(G)$ eines zusammenhängenden Graphen G ist als die Länge eines längsten kürzesten Weges in G definiert, d.h. $diam(G) := \max_{u,v \in V} dist(u, v)$. Der *Radius* $rad(G)$ eines zusammenhängenden Graphen G ist hingegen $rad(G) := \min_{u \in V} \max_{v \in V} dist(u, v)$.

Übung 1.1.12 *Für jeden zusammenhängenden Graphen G gilt:*

$$rad(G) \leq diam(G) \leq 2\, rad(G).$$

Man finde für jede dieser Ungleichungen Graphen, bei denen Gleichheit gilt.

Übung 1.1.13 *Mindestens einer der Graphen G und $\bar{G}$ ist zusammenhängend.*

Übung 1.1.14 *Ein Graph mit $m > \binom{n-1}{2}$ ist zusammenhängend.*

Übung 1.1.15 *Ein Graph mit $\Delta(G) \leq 2$ ist die disjunkte Vereinigung von Kreisen und Pfaden.*

1.1.1 Partite Graphen

Ein Graph G heißt *bipartit*[3] genau dann, wenn sich seine Knotenmenge so in zwei Teile U und V partitionieren läßt, daß Kanten nur zwischen U und V verlaufen, oder äquivalent, daß U und V stabile Mengen in G induzieren. $U \dot\cup V$ nennen wir dann eine *Bipartition* des bipartiten Graphen G, die Mengen U und V *Bipartitionsmengen* oder auch einfach *Teile*. Einen bipartiten Graphen G notieren wir daher auch als Tripel $G = (U, V, E)$ mit $E \subseteq U \times V := \{\{u, v\} : u \in U, v \in V\}$. Die $|U| \times |V|$ Matrix $B = (b_{uv})$, $u \in U$, $v \in V$, mit

$$b_{uv} := \begin{cases} 1 & \text{falls und } \{u, v\} \in E, \\ 0 & \text{sonst} \end{cases}$$

heißt die *Bipartitionsmatrix* von G. Der *vollständig bipartite* Graph mit Teilen der Größe $|U| = r$ und $|V| = s$ wird mit $K_{r,s}$ notiert; er hat rs viele Kanten. Der Graph $K_{2,2}$ beispielsweise ist isomorph zum C_4. Der Graph Q_d, dessen Knotenmenge aus den 2^d 0-1-Vektoren der Länge $d \in \mathbb{N}$ besteht, wobei zwei solche 0-1-Vektoren durch eine Kante verbunden sind genau dann, wenn sie sich in genau einer Koordinate unterscheiden, heißt der *d-dimensionale Hyperwürfel*.[4]

Proposition 1.1.16 *Der d-dimensionale Hyperwürfel Q_d ist bipartit.*

Beweis. Unterteile die 0-1-Vektoren $x \in V(Q_d)$ der Länge d in zwei Klassen U_0 und U_1 gemäß ihres HAMMING-Gewichts $||x||_H := \sum_{i=1}^{d} x_i$ modulo 2. □

Übung 1.1.17 *Kreise sind bipartit genau dann, wenn sie gerade sind.*

Wenn man die Kanten eines Kreises in einem bipartiten Graphen $G = (U, V, E)$ abläuft, springt man ständig zwischen U und V hin und her. Also ist jeder Kreis in einem bipartiten Graphen gerade. Tatsächlich charakterisiert die Eigenschaft, höchstens gerade Kreise zu besitzen, bipartite Graphen bereits. KÖNIG hat dies 1936 im ersten Graphentheoriebuch überhaupt veröffentlicht.

Satz 1.1.18 (KÖNIG 1936)
Ein Graph ist bipartit genau dann, wenn er keine ungeraden Kreise enthält.

Beweis. Die Notwendigkeit der Bedingung sahen wir bereits ein. Besitze G also keine ungeraden Kreise. Da es genügt, nachzuweisen, daß jede Komponente von G bipartit ist, sei G o.B.d.A. zusammenhängend. Wähle einen beliebigen Knoten $u \in V$ aus, dann ist also für alle $v \in V$ die Größe $dist(u, v)$ wohldefiniert. Definiere für $i = 0, 1$ die Knotenmengen $U_i := \{v \in V | dist(u, v) \equiv i \pmod 2\}$. Da G zusammenhängt, folgt $V = U_0 \dot\cup U_1$. Es bleibt zu zeigen, daß U_i für $i \in \{0, 1\}$ stabil ist. Angenommen, $v_1, v_2 \in U_i$ für ein $i \in \{0, 1\}$ und $\{v_1, v_2\} \in E$. Sei ohne Einschränkung $dist(u, v_1) \leq dist(u, v_2)$. Wegen $dist(u, v_2) \leq dist(u, v_1) + 1$ folgt $dist(u, v_1) = dist(u, v_2)$. Sei v der erste gemeinsame Knoten zweier kürzester $v_1 - u -$ bzw. $v_2 - u -$Pfade. Dann bilden die in ihnen enthaltenen Teilpfade von v_1 bzw. v_2 nach v gemeinsam mit der Kante $\{v_1, v_2\}$ einen ungeraden Kreis, Widerspruch. □

[3] in der deutschen Literatur manchmal auch *paar*

[4] Zur Bedeutung der Hyperwürfelgraphen insbesondere bei Supercomputer-Architekturen siehe z.B. [HHW88, Lei92]

Proposition 1.1.19 (ERDŐS 1965)
Jeder Graph $G = (V, E)$ enthält einen bipartiten Subgraphen B mit $m(B) \geq m(G)/2$.

Beweis. Sei $V = V_1 \dot\cup V_2$ eine beliebige Partition der Knoten von G in zwei Teile. Solange es einen Knoten $v \in V$ gibt, der mehr Nachbarn in dem Teil, zu dem er gehört, als im anderen Teil hat, schlage man diesen Knoten dem anderen Teil zu. Bei jeder solchen Operation erhöht sich die Anzahl der Kanten, die zwischen V_1 und V_2 verlaufen, um mindestens 1. Also bricht dieses Verfahren nach höchstens m Schritten mit einer Partition $V = V_1 \dot\cup V_2$ ab, in der jeder Knoten mindestens ebenso vielen Nachbarn im gegenüber liegenden Teil hat wie in dem Teil, dem er zugehört. Sei F die Menge der zwischen V_1 und V_2 verlaufenden Kanten. Dann hat also in dem bipartiten Graphen $B = (V_1, V_2, F)$ jeder Knoten v einen Grad $d_B(v) \geq d_G(v)/2$. Mithin wird

$$|F| = \frac{1}{2} \sum_{v \in V_1 \cup V_2} d_B(v) \geq \frac{1}{2} \sum_{v \in V} \frac{d(v)}{2} = m(G)/2. \qquad \square$$

Allgemeiner heißt ein Graph *p-partit*, $p \in \mathbb{N}$, wenn sich seine Knotenmenge so in p Teile partitionieren läßt, daß Kanten nur zwischen Knoten aus verschiedenen Teilen verlaufen. Ein Graph ist also bipartit genau dann, wenn er 2-partit ist, und 1-partit genau dann, wenn er leer ist. Der vollständige p-partite Graph mit n_i Knoten in der i-ten Partitionsklasse, $1 \leq i \leq p$, wird mit $K_{n_1,\ldots,n_p}$, $n = \sum_{i=1}^{p} n_i$, notiert; bei diesem Graphen sind zwei Knoten benachbart genau dann, wenn sie in verschiedenen Partitionsklassen liegen. Der Graph $K_{2,2,2}$ beispielsweise ist der Gerüstgraph des Oktaeders, vgl. die Abbildung auf Seite 181.

Übungen

Übung 1.1.20 **a)** $rad(Q_d) = d = diam(Q_d)$. **b)** $\alpha(Q_d) = 2^{d-1}$.
c) *Eine nichtleere Menge enthält gleich viele gerade wie ungerade Teilmengen.*

Übung 1.1.21 *Ein Graph G ist bipartit genau dann, wenn $\alpha(H) \geq |H|/2$ für alle (induzierten) Subgraphen $H \subseteq G$.*

1.1.2 Operationen auf Graphen

Für eine Knotenmenge $S \subseteq V$ bezeichne $G - S := G[V \setminus S]$. Für einen Knoten $v \in V$ entsteht also $G - v := G[V \setminus \{v\}]$ durch Entfernen des Knotens v und aller mit ihm inzidierenden Kanten aus G. Für eine Kantenmenge $F \subseteq E$ bezeichne $G \setminus F$ den Graphen $(V, E \setminus F)$. Für ein $e \in \binom{V}{2} \setminus E$ sei $G + e := (V, E \cup \{e\})$. Für eine Kante $e \in E$ sei $G - e := (V, E \setminus \{e\})$; Der Graph G/e entsteht aus G durch *Kontraktion* der Kante e, d.h. die Endknoten der Kante $e = \{x, y\}$ werden identifiziert, so daß der Graph G/e anstatt der beiden Knoten x und y nun einen Knoten v_e enthält, der mit all jenen (von x und y verschiedenen) Knoten verbunden ist, die vorher mit x *oder* y verbunden waren: $\Gamma_{G/e}(v_e) = (\Gamma_G(x) \cup \Gamma_G(y)) \setminus \{x, y\}$.

Seien $G = (V_G, E_G)$ und $H = (V_H, E_H)$ zwei Graphen. $G \cup H := (V_G \cup V_H, E_G \cup E_H)$ ist die *disjunkte Vereinigung* der Graphen G und H. Beispielsweise ist also $K_{r,s} = \overline{K_r \cup K_s}$.

Der Graph $G * H := (V_G \cup V_H, E_G \cup E_H \cup \{\{v_G, v_H\} : v_G \in V_G \wedge v_H \in V_H\})$ heißt die *Verbindung* der Graphen G und H. Beispielsweise ist also $K_{n+1} = K_n * K_1$ oder $K_{r,s} = \overline{K_r} * \overline{K_s}$.

Der Graph, der entsteht, wenn man jeden Knoten eines Graphen G durch einen Graphen H ersetzt und verschiedene Kopien von H vollständig bipartit verbindet, wenn die

zugehörigen Knoten in G benachbart sind, wird als die *Komposition* $G[H]$ bezeichnet. Der Graph $G[H]$ hat also die Knotenmenge $V_G \times V_H$, und zwei seiner Knoten (g_1, h_1) und (g_2, h_2) sind benachbart genau dann, wenn $\{g_1, g_2\} \in E_G$ oder $g_1 = g_2$ und $\{h_1, h_2\} \in E_H$.

Das *kartesische Produkt* $G_1 \times G_2 = (V, E)$ zweier Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ ist definiert durch $V := V_1 \times V_2 := \{(v_1, v_2) : \ v_1 \in V_1, v_2 \in V_2\}$ mit $\{(u_1, u_2), (v_1, v_2)\} \in E$ genau dann, wenn $u_1 = v_1$ und $\{u_2, v_2\} \in E_2$ oder $u_2 = v_2$ und $\{u_1, v_1\} \in E_1$. Es ist also beispielsweise $C_4 = K_2 \times K_2$ und $P_r \times P_s$ der (zweidimensionale) $r \times s$ Gittergraph.

Die k-te *Potenz* G^k eines Graphen G hat dieselbe Knotenmenge wie G, und zwei Knoten u und v sind in G^k benachbart genau dann, wenn $dist_G(u, v) \leq k$.

Der Kanten- oder *Linegraph* $L(G)$ eines Graphen $G = (V, E)$ hat die Kanten von G als Knotenmenge, und zwei Kanten $e, f \in E$ sind in $L(G)$ als Knoten benachbart genau dann, wenn sie in G inzidieren (wenn also $e \cap f \neq \emptyset$ gilt). Der Linegraph des K_4 beispielsweise ist der Gerüstgraph $K_{2,2,2}$ des Oktaeders. Da die mit einem Knoten $v \in V$ inzidierenden Kanten in $L(G)$ eine Clique bilden, gilt $\omega(L(G)) \geq \Delta(G)$. Die Anzahl der Kanten von $L(G)$ ist

$$m(L(G)) = \sum_{v \in V} \binom{d(v)}{2} = \frac{1}{2} \sum_{v \in V} d(v)(d(v) - 1) = \frac{1}{2} \sum_{v \in V} d(v)^2 - m.$$

Offenbar haben der $K_{1,3}$ und der K_3 beide den K_3 als Linegraphen. Nach einem Satz von Whitney [Whi32a] sind dies jedoch die beiden einzigen nicht-trivialen, nicht-isomorphen zusammenhängenden Graphen, die isomorphe Linegraphen besitzen.

Übung 1.1.22 *Ein zusammenhängender Graph ist zu seinem Linegraphen isomorph genau dann, wenn er ein Kreis ist.*

1.1.3 Bäume, Artikulationen, Schnitte

Eine weitere einfache, aber wichtige Graphenklasse ist die der Bäume. Bäume sind beispielsweise als Datenstrukturen von großer Bedeutung, siehe z.B. [OW93, Mah92, HSA93, Wei95]. Historisch gesehen stammt das Interesse an ihnen aus der Chemie, als man Kohlenwasserstoffverbindungen zählte.

Definition 1.1.23 *Ein* Baum *ist ein zusammenhängender und kreisfreier Graph.*

Bäume sind also insbesondere bipartit. Ein Knoten v in einem Graphen heißt *Blatt* (oder auch Endknoten), falls $d(v) = 1$.

Lemma 1.1.24 *Jeder Baum hat mindestens zwei Blätter.*

Beweis. Beginnend in einem beliebigen Startknoten v_0 eines Baumes T, konstruiere man einen Weg $W = v_0, v_1, \ldots, v_\ell$, indem man in jedem Knoten $v_i \in W$ eine beliebige, aber bisher noch nicht abgeschrittene Kante $\{v_i, v_{i+1}\}$ wählt. Da T kreisfrei ist, trifft man nie auf einen bereits besuchten Knoten. Da T andererseits nur endlich viele Knoten hat, bricht der Prozeß irgendwann in einem Knoten v_ℓ ab. Da der Weg W in v_ℓ nicht fortgesetzt werden kann, ist v_ℓ aber ein Blatt. Wiederholt man dieses Verfahren vom Startknoten v_ℓ aus, findet man ein weiteres Blatt. □

Übung 1.1.25 *Ein nichtleerer Baum T ist ein Pfad genau dann, wenn $\forall v \in V\ (d(v) \leq 2)$.*

Bäume lassen sich auf mannigfaltige Weise charakterisieren.

Satz 1.1.26 *Für einen Graphen $G = (V, E)$ auf $|V| = n$ Knoten sind äquivalent:*

(1) G ist ein Baum*;*

(2) G ist zusammenhängend und hat $n - 1$ viele Kanten;

(3) G ist kreisfrei und hat $n - 1$ viele Kanten;

(4) G ist (Kanten-) maximal kreisfrei;

(5) G ist (Kanten-) minimal zusammenhängend;

(6) Je zwei Knoten aus V sind durch genau einen Weg verbunden.

Beweis. (1) $\Rightarrow$ (2): durch Induktion nach $n = |V|$. Für $n = 1, 2$ ist die Aussage offensichtlich. Sei G nun ein Baum auf $n \geq 3$ Knoten. Nach Lemma 1.1.24 besitzt G ein Blatt $b \in V$. Sei $e \in E$ die mit b inzidierende Kante. Dann ist auch der Graph $T := G - b$ zusammenhängend und kreisfrei, also ein Baum, und hat nach Induktionsvoraussetzung $m_T = n_T - 1$ viele Kanten. Also hat G genau $m = m_T + 1 = n_T = n - 1$ viele Kanten.
(2) $\Rightarrow$ (3): Angenommen, es gibt einen Kreis C in G. Da G zusammenhängt, gibt es zu jedem der $n - |C|$ Knoten v, die nicht auf dem Kreis liegen, einen kürzesten $v - C$–Pfad. Sei e_v die jeweils erste Kante auf dem $v - C$–Pfad. Dann sind alle e_v, $v \in V \setminus V(C)$, verschieden, und G hat mindestens $|C| + |V \setminus V(C)| = n$ Kanten, Widerspruch.
(3) $\Rightarrow$ (1): Da G kreisfrei ist, ist jede Komponente von G ein Baum. Seien H_i, $i = 1, \ldots, c(G)$, die Komponenten von G mit n_i Knoten und m_i Kanten. Nach (2) hat G also

$$m \;=\; \sum_{i=1}^{c(G)} m_i \;=\; \sum_{i=1}^{c(G)} (n_i - 1) \;=\; n - c(G) \tag{1.4}$$

viele Kanten. Nach Voraussetzung ist also $c(G) = 1$, d.h. G ist zusammenhängend.
(1) $\Rightarrow$ (6): In einem Baum ist jeder Weg ein Pfad, da Selbstüberkreuzungen zu Kreisen führen würden. Da G zusammenhängt, gibt es also zu je zwei Knoten x und y in G einen $x - y$–Pfad P_1. Angenommen nun, es gäbe einen weiteren, von P_1 verschiedenen $x - y$–Pfad P_2 in G. Der Teilweg von x bis zu einem Knoten a sei das maximale gemeinsame Anfangsstück beider Wege und der Knoten b der erste, beiden Pfaden wieder gemeinsame Knoten hinter a. Dann bildet der $a - b$–Teilpfad auf P_1 zusammen mit dem in umgekehrter Richtung durchlaufenen $a - b$–Teilpfad von P_2 einen Kreis in G.
(6) $\Rightarrow$ (5): Nach Voraussetzung ist G zusammenhängend. Angenommen $G - e$ wäre noch zusammenhängend für eine Kante $e = \{x, y\} \in E$. Dann gäbe es neben e noch einen weiteren $x - y$–Pfad in G, Widerspruch.
(5) $\Rightarrow$ (4): Sei G minimal zusammenhängend. Angenommen, G besäße einen Kreis. Dann ließe sich eine beliebige Kreiskante aus G entfernen, ohne den Zusammenhang zu zerstören. Es bleibt zu zeigen, daß durch Hinzufügen einer Kante $\{x, y\}$ zwischen zwei nicht benachbarten Knoten x und y in G ein Kreis entsteht. Dies folgt daraus, daß G zusammenhängend ist und folglich einen $x - y$–Pfad enthält, den die Kante $\{x, y\}$ zu einem Kreis schließt.
(4) $\Rightarrow$ (1): Angenommen, G wäre nicht zusammenhängend. Dann könnte man zwischen

zwei beliebigen Knoten aus zwei verschiedenen Komponenten von G eine Kante einfügen, ohne einen Kreis zu erzeugen. □

Ein kreisfreier Graph heißt *Wald*. Jede Komponente eines Waldes ist also ein Baum. In Gleichung (1.4) haben wir also gezeigt:

Korollar 1.1.27 *Ein Wald W mit $c(W)$ Komponenten hat $m = n - c(W)$ viele Kanten. Ein Graph G enthält einen Kreis genau dann, wenn $m > n - c(G)$.* □

Wir erinnern daran, daß ein Baum $T \subseteq G$ (auf)spannend[5] heißt, falls $V(T) = V(G)$.

Korollar 1.1.28 (KIRCHHOFF 1847) *Ein Graph G ist zusammenhängend genau dann, wenn er einen aufspannenden Baum T (als Subgraphen) besitzt.*

Beweis. „⇐": Nach Satz 1.1.26 (6) ist schon der Subgraph T von G zusammenhängend. „⇒": Ist G selbst kein Baum, so enthält er einen Kreis. Entferne eine Kante e aus diesem Kreis. Dann ist der verbleibende Graph $G - e$ noch immer zusammenhängend. Wiederhole diese Prozedur, bis nach Satz 1.1.26 (5) ein Baum T übrigbleibt. □

Definition 1.1.29 *Sei G ein Graph und u, v zwei Knoten aus V. Eine Knotenmenge $A \subset V - u - v$ (bzw. eine Kantenmenge $F \subseteq E$) heißt $u - v-$*trennend*, falls u und v in G, nicht aber in $G - A$ (bzw. $G - F$) durch einen Pfad verbunden sind. Eine Knotenmenge $A \neq \emptyset$ (bzw. eine Kantenmenge $F \neq \emptyset$) heißt* trennend*, falls es Knoten u und v gibt, so daß A (bzw. F) $u - v-$trennend ist. Eine trennende Kante heißt auch* Brücke*, ein trennender Knoten auch* Artikulation *oder* Schnittpunkt*. Die leere Knoten- oder Kantenmenge heißt trennend genau dann, wenn der Graph unzusammenhängend ist.*

Beispiele. Der vollständige Graph enthält offenbar keine trennende Knotenmenge. Die Menge der mit einem Knoten inzidierenden Kanten ist stets eine trennende Kantenmenge. Eine nicht-leere Kantenmenge F in einem Graphen G ist trennend genau dann, wenn ihr Entfernen aus G die Anzahl der Zusammenhangskomponenten erhöht. Nach Satz 1.1.26 (5) bzw. (6) ist beispielsweise jede Kante eines Baumes eine Brücke und jeder Knoten, der kein Blatt ist, eine Artikulation. □

Die folgende Aussage werden wir häufig benutzen, ohne daß wir dies noch explizit erwähnen werden.

Lemma 1.1.30 *Sei A eine (inklusionsweise) minimal trennende Knotenmenge in einem zusammenhängenden Graph G. Dann ist jeder Knoten in A zu jeder Komponente von $G - A$ verbunden.*

Beweis. Für jedes $v \in A$ ist nach Voraussetzung $G - (A - v)$ noch zusammenhängend. Da es zwischen den Komponenten von $G - A$ keine Kanten gibt, sind sie offenbar in $G - (A - v)$ (und mithin in G) alle mit v verbunden. □

Proposition 1.1.31 *Für einen Knoten a in einem zusammenhängenden Graphen G sind äquivalent:*

(i) a ist eine Artikulation in G;

(ii) Es gibt zwei Knoten $u, v \in V - a$, so daß a auf jedem $u - v-$Weg liegt;

[5] in der deutschen Literatur auch *Gerüst* von G

(iii) Es gibt eine Partition $V - a = X \dot\cup Y$ mit $X \neq \emptyset \neq Y$, so daß für je zwei Knoten $x \in X$ und $y \in Y$ der Knoten a auf jedem $x - y-$Weg liegt.

Beweis. (i) $\Leftrightarrow$ (ii) gilt nach Definition einer trennenden Knotenmenge. Da (iii) $\Rightarrow$ (ii) trivial ist, genügt es, noch (i) $\Rightarrow$ (iii) zu zeigen. $G - a$ habe Zusammenhangskomponenten H_i, $i = 1, \ldots, k$, $k \geq 2$, $V(H_i) \neq \emptyset$. Setze $X := V(H_1)$ und $Y := \bigcup_2^k V(H_i)$. Da es zwischen verschiedenen Zusammenhangskomponenten eines Graphen keine Kanten gibt, führt für $x \in X$ und $y \in Y$ jeder $x - y-$Weg über a. □

Analog lassen sich Brücken charakterisieren.

Übung 1.1.32 *Sei $G = (V, E)$ ein zusammenhängender Graph, $e \in E$ eine Kante. Dann sind äquivalent:*

1. *e ist eine Brücke in G;*
2. *e liegt nicht auf einem Kreis in G;*
3. *Es gibt zwei Knoten $u, v \in V$, so daß e in jedem $u - v-$Weg enthalten ist;*
4. *Es gibt eine Partition $V = X \dot\cup Y$ von V, so daß für je zwei Knoten $x \in X$ und $y \in Y$ die Kante e in jedem $x - y-$Weg enthalten ist.*

Proposition 1.1.33 *Sei G ein zusammenhängender Graph. Eine Kantenmenge $F \subseteq E$ ist trennend genau dann, wenn $F \cap E(T) \neq \emptyset$ für alle spannenden Bäume T in G.*

Beweis. „$\Rightarrow$“: Angenommen, $F \cap E(T) = \emptyset$ für einen spannenden Baum T von G. Dann enthält $G - F$ also noch den spannenden Baum T und ist daher nach Lemma 1.1.28 noch zusammenhängend, Widerspruch.
„$\Leftarrow$“: Angenommen, F ist nicht trennend, d.h. $G - F$ ist zusammenhängend. Dann enthält $G - F$ nach Lemma 1.1.28 einen aufspannenden Baum T, Widerspruch. □

Sei G ein Graph und $A \dot\cup B$ eine Partition von V, $A \neq \emptyset \neq B$. Die Menge aller zwischen A und B verlaufenden Kanten bezeichnen wir dann als den *Schnitt*

$$\langle A, B \rangle \;:=\; \{\{a, b\} \in E : \; a \in A \wedge b \in B\}.$$

Ein Graph ist also unzusammenhängend genau dann, wenn es einen leeren Schnitt gibt.

Proposition 1.1.34 *Eine Kantenmenge $\emptyset \neq F \subseteq E$ ist trennend genau dann, wenn sie einen nicht-leeren Schnitt enthält.*

Beweis. „$\Rightarrow$“: Sei F $a - b-$trennend für gewisse $a, b \in V$, und sei $G[A]$, $A \subset V$, die Komponente von $G - F$, die a enthält, sowie $B := V - A$. Nach Definition von A und B gilt $a \in A \neq \emptyset \neq B \ni b$ und $F \supset \langle A, B \rangle$. Da F $a - b-$trennend ist, enthält G einen $a - b-$Pfad. Dieser muß notwendig den Schnitt $\langle A, B \rangle$ kreuzen. Also ist auch $\langle A, B \rangle \neq \emptyset$.
„$\Leftarrow$“: Sei $F \supseteq \langle A, B \rangle \ni \{a, b\}$. Dann ist F $a - b-$trennend. □

Übungen

Übung 1.1.35 (WAGNER 1970) Ein Graph G heißt *geodätisch*, wenn es für alle $(u, v) \in V \times V$ genau einen kürzesten $u - v-$Pfad in G gibt. *Ein Graph G ist ein Baum genau dann, wenn G bipartit und geodätisch ist.*

Übung 1.1.36 *Ein Baum hat* $2 + \sum_{v:\, d(v)\geq 3} (d(v) - 2)$ *viele Blätter.*

Übung 1.1.37 *Ein Graph* G *ist ein Baum genau dann, wenn es eine (sogenannte Abpflück-) Anordnung* $v_1, \ldots, v_n$ *seiner Knoten gibt, so daß für* $i = n, \ldots, 1$ *gilt:* v_i *ist Blatt in* $G[\{v_1, \ldots, v_i\}]$.

Übung 1.1.38 *Sei* G *ein Graph mit Minimalgrad* $\delta(G) = k$. *Dann ist jeder Baum* T *mit* **a)** k *Kanten Subgraph von* G; **b)**[Mih92] $k+1$ *Kanten Subgraph von* G *mit Ausnahme des* $T = K_{1,k+1}$ *für reguläres* G. [*Hinweis:* Induktion nach $|T|$].
c) *Sei* G *ein Graph mit mehr als* $(k-1)(n-1)$ *Kanten; dann ist jeder Wald auf höchstens* $k+1$ *Knoten Subgraph von* G, *und ein solcher Subgraph kann in Zeit* $\mathcal{O}(kn)$ *bestimmt werden.*

Das *Zentrum* eines zusammenhängenden Graphen ist die Menge der Knoten $v \in V$ mit $\max_{u \in V-v} d(u, v) = rad(G)$.

Übung 1.1.39 [Jor69] *Das Zentrum eines Baumes besteht aus einer Ecke oder einer Kante.*

Übung 1.1.40 [?] *Sei* S *eine* n*-elementige Menge und* $\mathcal{A} = \{A_1, \ldots, A_n\}$ *eine Familie von Teilmengen von* S. *Dann existiert ein* $x \in S$ *derart, daß die Mengen* $A_i \cup \{x\}$, $i = 1, \ldots, n$, *alle verschieden sind.* [*Hinweis:* betrachte die Kanten des Graphen $G = (\mathcal{A}, E)$, in dem $\{A_i, A_j\} \in E$ genau dann, wenn $|A_i \Delta A_j| = 1$.]

Übung 1.1.41 *Jeder nicht-triviale, zusammenhängende Graph besitzt mindestens zwei Knoten, die keine Artikulationen sind, und es gilt Gleichheit genau für die Pfade.*

Übung 1.1.42 *Sei* G *ein zusammenhängender Graph. Eine Kantenmenge* $F \subseteq E$ *in* G *ist (inklusionsweise) minimal trennend genau dann, wenn* F *ein Schnitt* $\langle A, B\rangle$ *ist, wobei* $G[A]$ *und* $G[B]$ *zusammenhängend sind.*

Übung 1.1.43 (Graham, Entringer, Székely) *Im* d*-dimensionalen Würfel* $Q_d = (V, E)$ *hat jeder spannende Baum* T *Durchmesser* $diam(T) \geq 2d - 1$. [*Hinweis:* der Antipode eines Knotens $v \in V$ sei der eindeutige Knoten $\overline{v} \in V$ mit $dist(v, \overline{v}) = d$. Für alle Knoten $v \in V$ markiere nun die erste Kante auf dem $v - \overline{v}$–Pfad in T in Richtung $\overline{v}$ und benutze das Schubfachprinzip] *Konstruiere einen spannenden Baum in* Q_d *mit Durchmesser* $2d - 1$.

1.1.4 Zufällige Graphen

Sei $\mathcal{G}_n$ die Menge aller *markierten* Graphen auf n Knoten, das heißt, daß alle Graphen aus $\mathcal{G}_n$ die gleiche Knotenmenge $V_n := \{v_1, \ldots, v_n\}$ haben und wir isomorphe Graphen, die durch eine nichttriviale Permutation der Knotenmenge ineinander überführt werden können, als verschieden ansehen; $\mathcal{G}_n$ enthält also $2^{\binom{n}{2}}$ Graphen. Wir werden $\mathcal{G}_n$ nun mit verschiedenen Wahrscheinlichkeitsmaßen versehen und damit zu einem Wahrscheinlichkeitsraum machen.

Zunächst betrachten wir das Laplacemaß auf $\mathcal{G}_n$; wir nehmen also alle Graphen aus $\mathcal{G}_n$ als gleichwahrscheinlich an. G_n bezeichne im folgenden ein typisches Element aus $\mathcal{G}_n$, d.h. für alle Graphen $H \in \mathcal{G}_n$ gilt

$$Prob\,(G_n = H) \;=\; 2^{-\binom{n}{2}}.$$

G_n heißt ein *zufälliger Graph* (aus $\mathcal{G}_n$). Da für jede Menge $\{v_i, v_j\} \in \binom{V_n}{2}$ die Anzahl der Graphen aus $\mathcal{G}_n$, die $\{v_i, v_j\}$ als Kante enthalten, gleich der Anzahl der Graphen ist, die $\{v_i, v_j\}$ nicht als Kante enthalten, gilt

$$Prob\,(\{v_i, v_j\} \in G_n) \;=\; \frac{1}{2},$$

und für verschiedene potentielle Kanten sind diese Wahrscheinlichkeiten unabhängig voneinander. Umgekehrt erhalten wir, wenn wir für alle $\{i,j\} \in \binom{V_n}{2}$ die Kante $\{v_i, v_j\}$ unabhängig von allen anderen Kanten mit der Wahrscheinlichkeit $\frac{1}{2}$ auswählen, als Ergebnis jeden Graphen aus $\mathcal{G}_n$ mit Wahrscheinlichkeit $2^{-\binom{n}{2}}$, also wieder die Gleichverteilung auf $\mathcal{G}_n$. Diese Interpretation des Wahrscheinlichkeitsraumes $\mathcal{G}_n$ wird sich als besonders nützlich erweisen.

Wenn wir statt dessen jede Kante mit der Wahrscheinlichkeit p auswählen, wobei $0 \leq p \leq 1$ eine Konstante oder eine Funktion von n ist, erhalten wir, wie wir gleich sehen werden, ein anderes Wahrscheinlichkeitsmaß auf $\mathcal{G}_n$. $\mathcal{G}_n$ mit diesem Maß bezeichnen wir als $\mathcal{G}_{n,p}$. Ein zufälliger Graph $G_{n,p} \in \mathcal{G}_{n,p}$ läßt sich also als das Ergebnis von $\binom{n}{2}$ unabhängigen BERNOULLI-Versuchen betrachten, bei denen jeweils eine Menge $\{u,v\} \in \binom{V_n}{2}$ unabhängig mit Wahrscheinlichkeit p als Kante definiert wird. Ein bestimmter Graph $H \in \mathcal{G}_{n,p}$ mit m Kanten, d.h. ein Elementarereignis, hat also die Wahrscheinlichkeit $Prob\,(H) = p^m(1-p)^{\binom{n}{2}-m}$. Beachte: nach dem Binomischen Lehrsatz gilt

$$\sum_{H \in \mathcal{G}_{n,p}} Prob\,(H) \;=\; \sum_{m=0}^{\binom{n}{2}} \binom{\binom{n}{2}}{m} p^m (1-p)^{\binom{n}{2}-m} \;=\; (p+1-p)^{\binom{n}{2}} \;=\; 1.$$

Für die Wahl $p = 1/2$ erhält man das LAPLACE-Maß auf $\mathcal{G}_n$; man schreibt auch kurz $\mathcal{G}_n$ für $\mathcal{G}_{n,1/2}$.

Statt jede Kante mit einer gewissen Wahrscheinlichkeit auszuwählen, hält man bisweilen die Anzahl der Kanten m fest. Der dazugehörige Wahrscheinlichkeitsraum $\mathcal{G}_{n,m}$ wird mit dem LAPLACEmaß ausgestattet. Für einen bestimmten Graphen $H \in \mathcal{G}_{n,m}$ ist also $Prob\,(G_n = H) = 1/\binom{\binom{n}{2}}{m}$. Da ein zufälliger Graph aus $\mathcal{G}_{n,p}$ im Erwartungswert $\binom{n}{2}p$ Kanten besitzt, hat ein solcher Graph ähnliche Eigenschaften wie ein Graph aus $\mathcal{G}_{n,m}$ mit $m = \binom{n}{2}p$ (vgl. [Bol85, S.34, Theorem 2]).

Eine *Grapheneigenschaft* $\mathcal{A}$ ist eine Untermenge der Menge aller Graphen, so daß für je zwei isomorphe Graphen G_1 und G_2 gilt $G_1 \in \mathcal{A} \Leftrightarrow G_2 \in \mathcal{A}$. Man sagt, daß ein Graph G die Eigenschaft $\mathcal{A}$ hat, falls $G \in \mathcal{A}$. Eine Grapheneigenschaft heißt *monoton* [bzw. *erblich* oder auch *hereditär*], falls für jeden Graphen mit dieser Eigenschaft gilt, daß auch jeder seiner [induzierten] Subgraphen diese Eigenschaft trägt. Jede monotone Eigenschaft ist also auch hereditär, doch ist beispielsweise die Eigenschaft eines Graphen, vollständig zu sein, hereditär, aber nicht monoton.

Wir sagen ferner, daß *fast alle* Graphen die Eigenschaft $\mathcal{A} \subseteq \mathcal{G}_{n,p}$ haben, beziehungsweise die Aussage $\mathcal{A}$ *fast sicher* gilt, falls

$$Prob\,(G_{n,p} \text{ hat } \mathcal{A}) \;=\; 1 - o(1) \qquad \text{für } n \to \infty.$$

Für eine gegebene Grapheneigenschaft $\mathcal{A}$ ist es nun interessant, zu fragen, für welche p gilt, daß fast alle Graphen aus $\mathcal{G}_{n,p}$ die Eigenschaft $\mathcal{A}$ haben, oder für welche p fast kein Graph aus $\mathcal{G}_{n,p}$ die Eigenschaft $\mathcal{A}$ hat. In einer Reihe von bahnbrechenden Arbeiten wurden diese Fragestellungen zuerst von ERDŐS und RÉNYI (vgl. [ER60]) für verschiedene Eigenschaften wie Zusammenhang, Auftreten von Untergraphen und anderen untersucht. Sei beispielsweise $\mathcal{A}_k$ die Eigenschaft eines Graphen, einen Pfad der Länge k zu enthalten. Für kleines $0 \leq p << 1$ werden nur wenige Graphen aus $\mathcal{G}_{n,p}$ einen P_{k+1} enthalten, während für $p \to 1$ fast alle Graphen die Eigenschaft $\mathcal{A}_k$ haben. Interessanterweise springt

die Wahrscheinlichkeit für $\mathcal{A}_k$ ganz plötzlich von 0 auf 1 – man spricht von einem Phasenübergang. Für $p = o(n^{-1-\frac{1}{k-1}})$ hat nämlich fast kein Graph aus $\mathcal{G}_{n,p}$ die Eigenschaft $\mathcal{A}_k$, während für $p = \omega(n^{-1-\frac{1}{k-1}})$ fast alle Graphen aus $\mathcal{G}_{n,p}$ einen Pfad der Länge k enthalten. Die Funktion $s(n) := n^{-1-\frac{1}{k-1}}$ heißt *Schwellenwertfunktion* für die Eigenschaft $\mathcal{A}_k$. Die Schwellenwertfunktion für das Auftreten eines Kreises liegt bei $s(n) = 1/n$. Sobald $p = c\frac{\log n}{n}$, $c > 1$, (d.h. wir erwarten $E(m(G_{n,p})) \sim c\frac{1}{2}n \log n$ viele Kanten in einem zufällig aus $\mathcal{G}_{n,p}$ herausgegriffenen Graphen) ist $G_{n,p}$ fast sicher zusammenhängend, d.h. er besitzt einen spannenden Baum. Die Schwellenwertfunktion dafür, daß $G_{n,p}$ fast sicher eine k-Clique enthält, ist schließlich $n^{-2/(k-1)}$. Offenbar haben insbesondere für festes $p \in (0,1)$ fast alle Graphen in $\mathcal{G}_{n,p}$ die oben genannten Eigenschaften. Man kann zeigen, daß jede nicht-triviale monotone Grapheneigenschaft eine Schwellenwertfunktion hat [BT86].

Wo es sich anbietet, werden wir im folgenden Schwellenwertfunktionen und andere Eigenschaften von zufälligen Graphen (meist allerdings ohne Beweis) zur Illustration angeben. „Benutzen“ werden wir zufällige Graphen lediglich in Abschnitt 5.3 und Kapitel 13. In letzterem werden wir die Cliquenzahl und die chromatische Zahl eines zufälligen Graphen bestimmen und daraus probabilistische Aussagen über Greedy-Algorithmen ableiten.

1.2 Die Komplexität von Algorithmen

1.2.1 Entscheidbarkeit

Zu den bedeutenden Errungenschaften der Mathematik im 20. Jahrhundert aus dem Blickwinkel der Logik und Informatik zählen die Unentscheidbarkeitsaussagen von CHURCH [Chu36], GÖDEL [Göd31], KLEENE [Kle36] und TURING [Tur36] aus den dreißiger Jahren. Der böhmische Mathematiker Kurt GÖDEL hatte die Mathematik in eine tiefe Krise gestürzt, als er mit seinem Unvollständigkeitssatz nachwies, daß sich in jedem nicht ganz trivialen Axiomensystem Aussagen formulieren lassen, deren Richtigkeit innerhalb des Systems weder beweisbar noch widerlegbar ist - zwischen den Begriffen wahr und beweisbar also zu unterscheiden ist.[6] HILBERTs Programm zum Beweis der Widerspruchsfreiheit der reinen Zahlentheorie war damit (zumindest vom streng finiten Standpunkt aus) gescheitert; das Fundament der exaktesten aller Wissenschaften brüchig geworden.

TURING nun nahm der Welt zudem die Hoffnung, man könne, gegeben eine mathematische Aussage, nach Anwendung einer endlichen Zahl von mathematischen Operationen jeweils wenigstens immer entscheiden, ob es sich um ein solches „schwarzes Schaf" handele. Ganz pragmatisch reduzierte er diese Frage nach der Entscheidbarkeit einer Aussage auf die Existenz einer (gedachten) primitiven, mechanischen Maschine, der sogenannten Turing-Maschine, die entweder nach endlich vielen Schritten anhält und die korrekte Antwort, *JA* oder *NEIN*, ausgibt - oder aber nie zum Stillstand kommt, was heißt, daß sich die entsprechende Frage nicht entscheiden läßt. Dieses Modell lieferte erstmals eine präzise Definition von *Berechen-* bzw. *Entscheidbarkeit*, indem es die intuitive Vorstellung formalisierte, wann eine Berechnungsprozedur ein „Algorithmus" ist. Die Mächtigkeit dieser Begriffsbildung zeigt sich darin, daß Turing-Maschinen dieselbe Klasse berechenbarer (*rekursiver*) Funktionen oder entscheidbarer Probleme definiert, wie andere, formal-logische Konzepte, die zur selben Zeit unabhängig in Princeton entwickelt wurden: partiell- oder μ-rekursive Funktionen [Kle36] und der λ-Kalkül [Chu36].[7] Diese Robustheit des Berechen- und Entscheidbarkeitsbegriffs führte zu der heute allgemein anerkannten CHURCH-TURING-*These*, die Turing-Maschine als formales Äquivalent der Idee von Berechenbarkeit schlechthin anzusehen: ein Problem ist berechenbar genau dann, wenn es auf einer Turing-Maschine berechenbar ist.

Darüberhinaus lassen sich mit Turing-Maschinen erheblich komplexere und realistischere Maschinenmodelle (wie Random-Access-Maschinen) (in polynomieller Zeit, s.u.) simulieren (und umgekehrt), so daß man bei der Beschreibung eines Algorithmus vom Maschinenmodell absehen und sich darunter ein in einer höheren Programmiersprache wie PASCAL formuliertes Programm vorstellen darf. Wir benutzen daher im folgenden die Wörter Algorithmus und Turing-Maschine synonym.

Sobald ein Modell für Berechenbarkeit zur Verfügung stand, das berechenbare Funktionen und entscheidbare Probleme charakterisierte, war es auch möglich, bestimmte Probleme als so schwer auszuweisen, daß es für ihre *allgemeine* Lösung keine Berechnungsprozedur geben kann. Das sogenannte HALTEPROBLEM [Tur36]: gegeben eine Turing-Maschine und eine Eingabe, (im voraus) zu entscheiden, ob diese auf die Eingabe hin anhalten wird oder nicht, ist ein solches, *prinzipiell nicht entscheidbares* Problem, und es fanden sich

[6] siehe den sehr lesenswerten Artikel von USPENSKY [Usp94]

[7] Äquivalent sind ebenso verschiedene spätere Formulierungen wie POSTs kanonische Systeme, CHOMSKYs Grammatiken [Cho59] oder MARKOV-Algorithmen [Mar51], siehe [Dav82].

andere (natürliche!) in den verschiedensten Bereichen der Mathematik: ein berühmtes ist HILBERTs zehntes Problem über die Lösbarkeit Diophantischer Gleichungen: gegeben ein Polynom p mit ganzzahligen Koeffizienten in n Variablen - besitzt p Wurzeln in $\mathbb{Z}^n$? (siehe [Dav73]).

Auf der Seite der unentscheidbaren Probleme (Theorie der rekursiven Funktionen, siehe [Rog67]) wie der entscheidbaren (Komplexitätstheorie) wurde bald die Klassifizierung des Schwierigkeitsgrads verfeinert. Bei entscheidbaren Problemen bemißt sich ihr Schwierigkeitsgrad (ihre Komplexität) an der Höhe der *Ressourcen* (dies sind *Zeit*, d.h. die Anzahl Schritte, die ein Algorithmus ausführt, oder Größe des *Speicherplatzes*, den er benutzt), die notwendig bzw. ausreichend sind, um das Problem zu entscheiden bzw. die Funktion zu berechnen (quantitative Komplexitätstheorie). Ziel ist es, für möglichst große Problemklassen möglichst scharfe obere wie untere Schranken anzugeben, die im Idealfall zusammenfallen[8]. Wann die Höhe von Ressourcen, die zur Lösung eines Problems erforderlich sind, als hoch bzw. niedrig einzustufen ist, wird zwar im Einzelfall Ermessenssache sein und zudem von der Leistungsfähigkeit der verfügbaren Rechenkapazitäten abhängen; bedarf es zur Lösung eines Problems jedoch mindestens (bezogen auf die Eingabegröße) exponentiell großer Ressourcen, so wird sich auch der Einsatz der übernächsten Rechnergeneration immer auf (mehr oder weniger) mäßig große Problemeingaben beschränken müssen („combinatorial explosion"). Ein solches Problem wird daher als *unzugänglich*[9] bezeichnet.

Beweisbar unzugängliche, aber entscheidbare Probleme konstruierten erstmals RABIN [Rab60] und HARTMANIS und STEARNS [HS65][10]. Sie zeigten, daß die Komplexität von Entscheidungsproblemen jede rekursive Funktion übertreffen kann. Erst Anfang der siebziger Jahre fanden MEYER und STOCKMEYER [MS72] sowie FISCHER und RABIN [FR74] natürliche Entscheidungsprobleme dieser Art in der Automatentheorie, der Theorie der formalen Sprachen und mathematischen Logik.

Für eine Vielzahl grundlegender Probleme der Mathematik und Informatik (nämlich die NP-vollständigen Probleme) ist der Status allerdings noch offen: sie sind zwar alle entscheidbar und zudem auf triviale Weise, nämlich durch Überprüfen aller Möglichkeiten („brute-force"-Methode oder auch vollständige Enumeration) immer in exponentieller Zeit berechenbar. Doch gelang es bisher weder, sie als unzugänglich zu klassifizieren (d.h. zu zeigen, daß es auch nicht effizienter geht), noch für irgendeines eine intelligente Lösungsstrategie mit subexponentieller Laufzeit anzugeben. Hier hat es sich als sinnvoll erwiesen, Probleme wenigstens implizit zu klassifizieren, indem man sie zu anderen einer ganzen Klasse in Beziehung setzt (qualitative oder strukturelle Komplexitätstheorie).

1.2.2 Die Klasse P oder: zugängliche Probleme

Durch ein Maschinenmodell wie die deterministische Turing Maschine[11] als formales Äquivalent von Berechenbarkeit lassen sich sogenannte Entscheidungsprobleme komplexitätstheoretisch besonders einfach beschreiben. Entscheidungsprobleme haben daher historisch gesehen eine besondere Bedeutung. Auch wir wollen sie zur Definition der Klasse P ver-

[8] Einen Überblick, insbesondere über explizite untere Schranken, bietet hier [Sto87].

[9] von englisch „intractable" = widerspenstig (AIGNER übersetzt mit „unangreifbar")

[10] Der Titel dieser Arbeit gab dem Forschungsgebiet seinen Namen.

[11] Für eine Beschreibung von Maschinenmodellen siehe z.B. [Pap94, GJ79, BDG88, HU79]

wenden. Ein *Entscheidungsproblem* Π, wie z.B. das Problem CLIQUE:

INSTANZ: ein Graph $G = (V, E)$ und ein $k \in \mathbb{N}$,
FRAGE: $\omega(G) \geq k$?

hat nur die zwei möglichen Lösungen: *JA* oder *NEIN*. Π versteht sich dabei als eine ganze Problemklasse, z.B. die Menge aller Tupel $\langle G, k\rangle$, wo G ein Graph und $k \in \mathbb{N}$, mit der Frage nach der Existenz einer solchen Clique. Eine einzelne Problemausprägung $\langle G, k\rangle$ wird dann als eine *Instanz* des Entscheidungsproblems Π bezeichnet. Ein Problem zu lösen heißt somit, einen Algorithmus anzugeben, der für alle Instanzen des Problems korrekt *JA* oder *NEIN* ausgibt.

Um Probleminstanzen eines Entscheidungsproblems in einen Computer (bzw. eine Turing-Maschine) eingeben zu können, werden sie „in vernünftiger Weise" durch eine Zeichenkette, d.h. durch ein Wort $x \in \Sigma^*$ über einem Alphabet Σ,[12] codiert, dem Eingabealphabet (der Turing-Maschine). Die Menge $D_\Pi \subseteq \Sigma^*$ bezeichnet dann die Menge der Codierungen der Instanzen des Entscheidungsproblems Π, die Menge $Y_\Pi \subseteq D_\Pi$ wiederum die Menge der Codierungen seiner *JA*-Instanzen. Dies ist im Beispiel CLIQUE also gerade die Menge der Codierungen der Tupel $\langle G, k\rangle$ mit $\omega(G) \geq k$.

Ein Algorithmus $\mathcal{A}$ *akzeptiert* eine Eingabe $x \in \Sigma^*$, falls er nach endlich vielen Schritten im *JA*-Zustand anhält. Die Menge aller dieser Eingabewörter

$$L_\mathcal{A} := \{x \in \Sigma^* : \mathcal{A} \text{ akzeptiert } x\}$$

heißt *die von $\mathcal{A}$ erkannte Sprache*. Ein Algorithmus $\mathcal{A}$ *löst* nun ein Entscheidungsproblem Π, falls er für *alle* Eingaben $x \in \Sigma^*$ anhält und eine Eingabe x genau dann akzeptiert, wenn die Antwort auf die durch x codierte Instanz von Π *JA* lautet, d.h. wenn $\mathcal{A}$ die Sprache Y_Π erkennt. Damit lassen sich Entscheidungsprobleme als Spracherkennungsprobleme auffassen - die *JA*-Instanzen von Π werden durch die von $\mathcal{A}$ erkannte Sprache $L_\mathcal{A}$ repräsentiert.

Die *Codierungslänge* $|x|$ einer Eingabe $x \in \Sigma^*$ soll die „Problemgröße" der durch x codierten Instanz messen und ist als die Anzahl der Zeichen (Bits) im Wort x definiert. (Bei Verwechslungsgefahr schreiben wir auch $\langle x\rangle$ statt $|x|$.) Beispiele:
(1) *Zahlen:* Die binäre Codierung einer natürlichen Zahl erfordert offenbar $\lfloor \log n \rfloor + 1$ Bits. Eine rationale Zahl $x = p/q \in \mathbb{Q}$, p und q teilerfremd, codieren wir, indem wir p und q codieren, und algebraische Zahlen lassen sich immerhin noch durch Angabe (der ganzzahligen Koeffizienten) eines Polynoms codieren, dessen Nullstelle sie sind.
(2) *Graphen:* Am einfachsten codieren wir einen Graphen $G = (V, E)$ durch seine Adjazenzmatrix. Dies erfordert allerdings einen Speicherplatzaufwand von $\binom{n}{2} = \Theta(n^2)$ vielen Bits. Wenn ein Graph nur wenige Kanten hat, ist es günstiger, ihn durch eine *Adjazenzliste* zu codieren. Dies ist ein Feld $Adj[i]$, $i = 1, \ldots, n$, dessen i-ter Eintrag eine (in der Regel einfach verkettete) Liste mit den Nachbarn des i-ten Knotens in G enthält. Unter Vernachlässigung des Speicheraufwandes $\lceil \log n \rceil$ für die Nummer eines Knotens[13] ist der Speicheraufwand hier $2m$. Offenbar kann man beide Codierungen in Zeit $\Theta(n^2)$ ineinander überführen. Je nach Anwendung kann das eine oder das andere Codierungsschema vorteilhafter sein (siehe unten).

[12] Wir nehmen hier ohne Einschränkung stets $\Sigma = \{0, 1\}$ an

[13] Wir legen hier – wie die Praxis es nahelegt – das sogenannte log-RAM-Modell zugrunde, wo eine Speicherzelle $\mathcal{O}(\log n)$ Bits (n die Größe der Eingabe) enthalten kann und die grundlegenden arithmetischen Operationen auf dieses Speicherzellen in konstanter Zeit ausführbar sind.

Die *Zeitkomplexität* oder *Laufzeitfunktion* $T_{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$ eines Algorithmus $\mathcal{A}$ ist wie folgt definiert: $T_{\mathcal{A}}(n)$ ist gleich der maximalen Anzahl von elementaren Rechenoperationen wie Additionen, Multiplikationen, Vergleiche, u.ä., die der Algorithmus $\mathcal{A}$ durchführt, um bei einer Eingabe $x \in \Sigma^*$ mit Codierungslänge $|x| = n$ den Haltezustand zu erreichen. Da für jedes n über alle Eingaben der Länge n maximiert wird, spricht man hier auch von worst-case-Zeitkomplexität.

Mit einem derartigen Maß für die Güte eines Algorithmus zur Hand kann man verschiedene Algorithmen (und Datenstrukturen) für ein betrachtetes Problem vergleichen. Da es meist recht schwierig ist, die Funktion $T_{\mathcal{A}}(n)$ für einen Algorithmus exakt zu bestimmen, gibt man sich hier mit einer Bestimmung der asymptotischen Wachstumsordnung von $T_{\mathcal{A}}(n)$ zufrieden und sagt, ein Algorithmus $\mathcal{A}$ ist effizienter als ein Algorithmus $\mathcal{B}$, wenn $T_{\mathcal{A}}(n) = o(T_{\mathcal{B}}(n))$ für $n \to \infty$. Ein Algorithmus ist jedoch offensichtlich optimal, wenn seine Laufzeit durch das Lesen der Problemeingabe bestimmt wird, d.h. wenn $T_{\mathcal{A}}(n) = \mathcal{O}(n)$.

Beispiele. Festzustellen, ob $\{u, v\} \in E$ gilt für zwei Knoten u und v eines Graphen, kostet bei Codierung als Adjazenzmatrix Zeit $\mathcal{O}(1)$, bei Codierung als Adjazenzliste Zeit $\mathcal{O}(\Delta(G)) = \mathcal{O}(n)$. Der Algorithmus, der aus der Adjazenzmatrix eines Graphen die Knotengrade berechnet, hat offensichtlich Laufzeit $\Theta(n^2)$. Die Knotengrade aus einer Adjazenzliste zu bestimmen, kostet dagegen „nur“ $\sum_{v \in V} d(v) = 2m$ viele Schritte. Ferner hat die Adjazenzliste den Vorteil, zu einem Knoten $v \in V$ in konstanter Zeit (irgend-) einen Nachbarn $u \in \Gamma(v)$ zurückzuliefern oder $d(v) = 0$ auszugeben (was bei der Adjazenzmatrix $\Theta(n)$ Schritte erfordert). Für die Operation „Löschen einer Kante in G“ ist dagegen wieder die Darstellung eines Graphen G durch seine Adjazenzmatrix günstiger; diese Operation kostet hier nur konstante Zeit, während sie bei der Codierung durch eine einfache Adjazenzliste $\Theta(\Delta(G)) = \Theta(n)$ Schritte erfordert. Auf Kosten höheren Verwaltungsaufwands kann man diese Operation aber auch für Adjazenzlisten in konstanter Zeit implementieren: eine Möglichkeit besteht z.B. darin, die Adjazenzlisten doppelt zu verketten und zusätzlich eine Referenzliste für die Kanten zu unterhalten, die die Positionen einer Kante $\{u, v\}$ in beiden Adjazenzlisten $Adj[u]$ und $Adj[v]$ vermerkt.

Die Komplexität eines Problems läßt sich als die Zeitkomplexität eines „schnellsten“ Algorithmus für dieses Problem definieren. Dies erlaubt eine Klassifikation des Schwierigkeitsgrades eines Problems. Die folgende Klasse enthält alle diejenigen Entscheidungsprobleme, für die es einen Algorithmus (eine deterministische Turing-Maschine) gibt, dessen Laufzeit „nur“ polynomiell mit der Eingabegröße wächst:

Definition 1.2.1 *Die Klasse* **P** *(von engl.* polynomial time*) ist die Menge aller Sprachen $L \subseteq \Sigma^*$, zu denen es einen Algorithmus $\mathcal{A}: \Sigma^* \to \{YES, NO\}$ gibt, so daß gilt:*

(i) $\mathcal{A}$ erkennt L, d.h. $L_{\mathcal{A}} = L$,

(ii) $\mathcal{A}$ hat polynomielle Zeitkomplexität, d.h. es gibt ein Polynom p, so daß

$$T_{\mathcal{A}}(n) \leq p(n) \qquad \forall n \in \mathbb{N}.$$

Ein Entscheidungsproblem Π gehört zu P*, wenn bezüglich einer „vernünftigen“ Codierung seiner Instanzen die Sprache Y_Π der Codierungen seiner JA-Instanzen zu* P *gehört.*[14]

[14] Man beachte, daß bei der Zeitkomplexitätfunktion über alle Eingaben $x \in \Sigma^*$ maximiert wird und nicht über die eigentlich nur relevanten Probleminstanzen $x \in D_\Pi$. Ein polynomieller Algorithmus für ein Problem Π muß also insbesondere die zulässigen Eingaben $x \in D_\Pi$ erkennen können. Dies führt zu

Die Unterscheidung zwischen polynomiellen und exponentiellen Algorithmen zur Lösung eines Problems und damit die (grobe) Einteilung aller Probleme in zwei Klassen: die polynomiell lösbaren Probleme und die Probleme, die exponentielle Ressourcen (hier: Laufzeit) beanspruchen, ist grundlegend für die Komplexitätstheorie. Sie kann bis auf VON NEUMANN [Neu53] zurückverfolgt werden, doch schufen erst die Arbeiten von COBHAM [Cob64] und EDMONDS [Edm65] (die polynomielle Lösung des allgemeinen Matching-Problems) das Bewußtsein für die Bedeutung dieser Unterscheidung.

Grundsätzlich mag natürlich ein exponentieller Algorithmus in der Praxis einem schlechten polynomiellen (etwa mit Laufzeit n^{100} oder $2^{100}n$) überlegen sein, zumindest für kleines n oder möglicherweise auch bei *fast allen* Problemeingaben: schließlich mißt die Laufzeitfunktion T_M lediglich die Anzahl Schritte, die ein Algorithmus im schlechtesten Fall benötigt. So verhält es sich z.B. mit dem Simplex-Algorithmus [Dan49] aus der Linearen Optimierung. Trotz exponentieller (worst-case-) Zeitkomplexität [KM72] ist er in der Praxis gutartig: erfahrungsgemäß ist die Anzahl der Iterationen sogar etwa linear bezogen auf die Eingabegröße, was BORGWARDT durch den Nachweis einer polynomiellen *durchschnittlichen* Laufzeit unter einem naheliegenden Wahrscheinlichkeitsmodell untermauert [Bor87]. (Das Problem LINEAR PROGRAMMING ist allerdings in P, vgl. [Kha79, Kar84], auch wenn man in der Praxis meist dem Simplex-Algorithmus den Vorzug gibt.)

Die Entwicklung eines polynomiellen Algorithmus mit *kleinen* Konstanten *beweist* dagegen die - wie wir sagen wollen - *effiziente* Lösbarkeit eines Problems in *allen* Fällen, so daß die Suche nach effizienten Algorithmen einen aktiven und interessanten Bereich der Theoretischen Informatik und Diskreten Mathematik darstellt. Daß man dabei für (natürliche) Probleme in P in fast allen Fällen sogar Algorithmen gefunden hat, deren Laufzeit durch ein Polynom mit kleinen Konstanten nach oben beschränkt ist, hat wesentlich dazu beigetragen, polynomielle Berechenbarkeit generell mit *effizienter Lösbarkeit* gleichzusetzen. Die Klasse P wird daher als die Klasse der *zugänglichen* (engl. tractable), d.h. in der Praxis entscheidbaren Probleme aufgefaßt - in Abgrenzung zu den schon oben erwähnten unzugänglichen Problemen, die exponentielle Ressourcen erfordern.

Neuere Arbeiten von ROBERTSON und SEYMOUR stellen diese Sichtweise jedoch in Frage; sie weisen Probleme als der Klasse P zugehörig aus, ohne daß sich daraus ein effizientes, allgemeines Lösungsverfahren ableitet, siehe [Joh87, FL94, AFLM91]. Ferner ist es möglich, daß andere, z.B. parallele Rechnerarchitekturen die Bedeutung des sequentiellen Berechnungsmodells zurückgehen lassen und die Menge der praktisch zugänglichen Probleme über die Klasse P hinaus erweitern (siehe dazu die Literaturhinweise im Anhang).

---

folgendem Paradox; betrachte die Probleme:

(1) Gegeben ein Hamiltonscher Graph G - ist G bipartit?

(2) Gegeben ein perfekter Graph G - hat G gerade Ordnung?

Zwar kann man für jeden Graphen in polynomieller Zeit testen, ob er bipartit ist oder gerade Ordnung hat. Im ersten Fall ist aber bereits das Problem, die zulässigen Eingaben zu erkennen, NP-schwer. Wenn man hier verabredet, daß zu einer „vernünftigen Codierung“ einer Eingabe die Codierung eines Hamiltonkreises gehört, so kann der Algorithmus plötzlich sogar die NP-vollständige Sprache der bipartiten Hamiltonschen Graphen erkennen. Im zweiten Fall ist z.Z. nicht einmal bekannt, ob es ein Zertifikat für die Perfektheit eines Graphen gibt. Wollte man mit einer Definition von P also die polynomiell lösbaren Probleme fassen, müßte man dem Algorithmus (der deterministischen Turing-Maschine) wohl noch ein Orakel für das Zugehörigkeitsproblem $x \in D_\Pi$ zur Verfügung stellen. Man könnte dann allerdings nicht mehr über die Komplexität einer Sprache $L = Y_\Pi$ reden, ohne die Bezugssprache D_Π anzugeben.

Die Begriffsbildung der Komplexitätsklasse P hat sich allerdings insofern als äußerst leistungsfähig erwiesen, als daß sie erstaunlich stabil unter Abänderung der Voraussetzungen ist. Dies gilt für die Wahl des Maschinenmodells - man glaubt sogar, daß die Klasse P im wesentlichen unabhängig vom zugrundegelegten (sequentiellen) Maschinenmodell ist (*Invarianz These*, siehe [AHU74, Kapitel 1]) - wie auch des Codierungsschemas. Wie bei Graphen die Darstellung als Adjazenzmatrix oder Adjazenzliste, so scheinen allgemein alle „vernünftigen“ (ohne dies formalisieren zu können) Codierungsschemata eines Problems jeweils in polynomieller Zeit ineinander überführbar zu sein, so daß ein Algorithmus mit polynomieller Laufzeit in einem Codierungsschema auch ein polynomieller Algorithmus bezüglich aller anderen ist. Wir werden daher oftmals auf den (abstrakten) Objekten eines Problems (wie Knoten und Kanten bei Graphen) operieren, ohne auf deren Codierung einzugehen, und von einem linearen, quadratischen, kubischen, u.s.w. Algorithmus sprechen, wenn die Laufzeit des Algorithmus linear, quadratisch, u.s.w. in $n + m$ ist (d.h. wir vergessen den Faktor $\log n$ für die Codierungslänge der involvierten Zahlen).

Schließlich bedeutet die Definition der Klasse P durch Entscheidungsprobleme keine wesentliche Einschränkung. Tatsächlich ist man in der Praxis häufig eher an Funktionen interessiert, wie z.B. der Größe einer größten Clique oder eines größten Matchings in einem vorgelegten Graphen, dem Gewicht eines minimal spannenden Baumes oder der chromatischen Zahl eines Graphen. Solchen Optimierungsproblemen kann man jedoch leicht ein Entscheidungsproblem zuordnen, wie z.B. hier: gegeben ein Graph G und ein $k \in I\!N$, „besitzt G eine Clique / ein Matching der Größe mindestens k?“, „einen spannenden Baum mit Gewicht höchstens k?“ oder „ist G k-färbbar?“. Sind diese Entscheidungsprobleme in polynomieller Zeit lösbar, so auch das Optimierungsproblem: da die Anzahl der möglichen Funktionswerte durch $2^{q(|x|)}$, q ein Polynom, beschränkt ist, lokalisiert man mittels binärer Suche den korrekten Funktionswert durch polynomiell viele Aufrufe des Algorithmus für das Entscheidungsproblem. Beide Problemarten sind also polynomiell äquivalent, d.h. das Optimierungsproblem ist genau dann in polynomieller Zeit lösbar, wenn es auch das zugehörige Entscheidungsproblem ist.

1.3 Algorithmische Prinzipien

1.3.1 Breiten- und Tiefensuche

Wir kommen nun zu einem für Graphenalgorithmen fundamentalen Prinzip, der Breitensuche (engl. Breadth-First-Search oder kurz BFS). Breitensuche ist eine Methode, einen Graphen zu durchsuchen bzw. zu durchlaufen, um Strukturinformation über den Graphen zu gewinnen. Wir sagen Prinzip deshalb, weil viele Graphenalgorithmen nichts anderes sind als eine auf das Problem abgestimmte Variante der Breitensuche.

Definition 1.3.1 *Sei $G = (V, E)$ ein Graph und $u \in V$ ein Knoten. Für $0 \leq i \leq n-1$ heißt dann die Knotenmenge $S_i(u) := \{v \in V : dist(u, v) = i\}$ die i-te Sphäre um u.*

Offensichtlich gilt $S_0(u) = \{u\}$ und $S_1(u) = \Gamma(u)$. Zu einem gegebenen Startknoten $u \in V$ berechnet der Algorithmus BREITENSUCHE beginnend mit $i = 0$ sukzessive aus der Sphäre $S_i(u)$ die Sphäre $S_{i+1}(u)$ durch Untersuchung der Nachbarn der Knoten in S_i - bis schließlich alle Knoten der Zusammenhangskomponente von G, die u enthält, besucht

wurden.[15] Die Knoten eines Graphen G seien wie $V = \{1, \ldots, n\}$ durchnumeriert. Der folgenden Algorithmus berechnet ein Feld $Sphäre : V \to \{0, \ldots, n-1\}$, das für jeden Knoten $v \in V$ angibt, zu welcher Sphäre $S_i(u)$ er gehört, d.h. es gilt $Sphäre[v] = i \Leftrightarrow v \in S_i(u)$.

```
PROCEDURE Breitensuche (G, u);

BEGIN
  FOR v ∈ V − u DO bekannt[v] := FALSE;
  FOR v ∈ V − u DO Sphäre[v] := ∞;
  bekannt[u] := TRUE;
  InitQueue[16] (Q);
  Insert (Q, u);
  Sphäre[u] := 0;
  WHILE Q ≠ ∅ DO BEGIN
    v := ExtractHead (Q);
    FOR w ∈ Γ(v) DO IF NOT bekannt[w] THEN BEGIN
      bekannt[w] := TRUE;
      Insert (Q, w);
      Sphäre[w] := Sphäre[v] + 1;
    END; {for}
  END; {while}
END; {Breitensuche}
```

Proposition 1.3.2

1. *Für das vom Algorithmus* Breitensuche (G, u) *berechnete Feld Sphäre gilt:* $Sphäre[v] = dist(u, v)$ *für alle* $v \in V$.

2. Breitensuche (G, u) *markiert genau diejenigen Knoten eines Graphen* G *als bekannt, die in der Komponente von* G*, die den Knoten* u *enthält, liegen.*

3. *Der Algorithmus* Breitensuche *hat Laufzeit* $\mathcal{O}(n + m)$.

Beweis. Ad 1. Zu Beginn wird die Wurzel u in die Queue (Warteschlange) Q gesteckt. Dann wird nach der Regel first-in-first-out sukzessive jeweils ein Knoten aus der Queue entfernt und wie folgt bearbeitet: alle seine Nachbarn, die noch nicht bekannt sind und deren *Sphäre*-Wert daher noch undefiniert ist, werden zur späteren Verarbeitung in die Queue eingefügt. Somit enthält Q im Verlauf von Breitensuche zunächst den Knoten u, dann alle Nachbarn von u, dann wiederum deren Nachbarn (außer u) - d.h. die zweite

[15] Dieser Aspekt des Algorithmus Breitensuche wird beispielsweise vom Algorithmus von Cuthill-McKee bei der Diskretisierung von partiellen Differentialgleichungen ausgenutzt, wo die zu einer Gebietsunterteilung gehörigen Knotenvariablen so durchnumeriert werden sollen, daß die Bandbreite der entstehenden Matrix möglichst klein wird, siehe [Sch84, Abschnitt 3.2.2].

[16] Eine Warteschlange Q (engl. queue) ist eine Datenstruktur, die es erlaubt, Elemente in Q einzufügen (mittels der Prozedur `Insert`(Q, u), die das Element u „am hinteren Ende“ von Q an Q anhängt) und in der selben Reihenfolge („first-in-first-out“ Prinzip) auch wieder aus Q zu entfernen (mittels der Funktion `ExtractHead`, die das Element am Kopf der Warteschlange entfernt und als Funktionswert zurückliefert). Eine Warteschlange läßt sich implementieren als eine (durch Zeiger) einfach verkettete Liste von Feldelementen, die dynamisch allokiert werden. Ein zusätzlicher Zeiger weist auf das letzte Element der Warteschlange.

Sphäre um u, u.s.w. Ein Wert $k+1$ für *Sphäre* wird daher erst vergeben, wenn alle Knoten der $(k-1)$-ten Sphäre besucht wurden und daher alle Knoten auf der k-ten Sphäre bekannt sind.
Ad 2. Induktiv folgt: alle Knoten, die von BREITENSUCHE markiert werden, sind mit u durch einen Pfad verbunden. Sei umgekehrt v ein Knoten aus der Komponente von G, die u enthält, und $u = u_0, u_1, \ldots, u_{dist(u,v)} = v$ ein kürzester $u - v-$Pfad in G. Dann folgt induktiv $Sphäre[u_i] = i$ für $i = 0, \ldots, dist(u, v)$, d.h. inbesondere wird der Knoten v markiert.
Ad 3. Da nach obigem jeder Knoten der Komponente von G, die u enthält, genau einmal in Q aufgenommen wird, ist die Anzahl Operationen, die mit `Insert`, `ExtractHead` und den Arrays *bekannt* und *Sphäre* verbunden sind, sicher $\mathcal{O}(n)$. Die `IF`-Abfrage wird höchstens $\sum_{v \in V} d(v) = 2m$ mal durchgeführt. □

Wir wollen einige unmittelbare Folgerungen aus dem Algorithmus BREITENSUCHE festhalten. Dazu zunächst die

Definition 1.3.3 *Das* Erkennungsproblem *oder auch* Zugehörigkeitsproblem *für eine Graphenklasse $\mathcal{G}$ lautet: gegeben ein Graph G, ist $G \in \mathcal{G}$? Eine Graphenklasse $\mathcal{G}$ kann in polynomieller Zeit* erkannt *werden, wenn es einen polynomiellen Algorithmus gibt, der das zugehörige Erkennungsproblem löst.*

Proposition 1.3.4

1. *Die Abstände $dist(u, v)$ zwischen allen Knotenpaaren in einem Graphen lassen sich in Zeit $\mathcal{O}(n(n + m))$ bestimmen.*
2. *Jeder Graph kann in linearer Zeit auf Zusammenhang getestet werden;*
3. *Die Zusammenhangskomponenten eines Graphen sind in linearer Zeit berechenbar;*
4. *In einem zusammenhängenden Graphen kann in linearer Zeit ein spannender Baum konstruiert werden;*
5. *Ein Graph läßt sich in Zeit $\mathcal{O}(n)$ auf Kreisfreiheit testen.*
6. *Bäume können in Zeit $\mathcal{O}(n)$ erkannt werden.*

Beweis. Ad 1. Zunächst initialisiert man die $n \times n$ Distanzmatrix d auf $d[u, v] := \infty$, $1 \leq i, j \leq n$. Dann ruft man für jeden Knoten $u \in V$ als Startknoten den Algorithmus BREITENSUCHE auf und trägt für alle Knoten v in der Komponente von G, die u enthält, den Wert $d[u, v] := Sphäre[v] < \infty$ ein.
Ad 2. Der Algorithmus BREITENSUCHE markiert genau dann *alle* Knoten eines Graphen als „bekannt“, wenn die Komponente, die u enthält, alle Knoten enthält, d.h. wenn G zusammenhängt.
Ad 3. Die Zusammenhangskomponenten eines Graphen G partitionieren seine Knotenmenge. Die folgende Prozedur wählt aus jeder Zusammenhangskomponente von G den ersten Knoten, der sie begegnet, als Repräsentant dieser Komponente aus und codiert damit wie folgt die Komponenten von G. ZUSAMMENHANGSKOMPONENTEN berechnet ein Feld $Komponente : V \to \{0, \ldots, n\}$, so daß für alle $v \in V$ $Komponente[v]$ schließlich die Knotennummer des eindeutigen Repräsentanten der Komponente von G, die v enthält,

angibt. In der Prozedur BREITENSUCHE wird dazu die Verwaltung der bekannten Knoten statt mit Hilfe des Feldes $bekannt[\cdot]$ in naheliegender Weise durch das Feld $Komponente[\cdot]$ realisiert.

```
PROCEDURE ZUSAMMENHANGSKOMPONENTEN (G);

BEGIN
  FOR v ∈ V DO  Komponente[v] := 0;
  FOR v ∈ V DO IF Komponente[v] = 0 THEN
    BREITENSUCHE (v);
END;
```

Ad 4. Behauptung: Die Vereinigung aller Kanten $\{v, w\}$, über die BREITENSUCHE einen bislang noch unbekannten Knoten $w \in \Gamma(v)$ von einem bereits bekannten Knoten v aus erreicht, bildet einen spannenden Baum T in G.
Beweis: Da G als zusammenhängend vorausgesetzt ist, markiert BREITENSUCHE nach Proposition 1.3.2(2.) alle Knoten von G als bekannt, so daß T ein spannender Subgraph von G ist.
Nach Satz 1.1.26(3.) ist T auch ein Baum, denn T enthält offenbar genau $n-1$ Kanten und ist kreisfrei, was man wie folgt einsieht. Die Endknoten einer jeden Kante in T aufgenommenen Kante sind stets „bekannt". Da BREITENSUCHE eine Kante $\{v, w\}$ nur in T aufnimmt, wenn der Knoten w noch „unbekannt" ist, nimmt BREITENSUCHE also keine Kante in T auf, die einen Kreis schließen würde, da in diesem Moment bereits alle Kreisknoten bekannt wären.
Ad 5. Nach Korollar 1.1.27 enthält ein Graph G einen Kreis genau dann, wenn $m > n - c(G)$. Man braucht also lediglich die Kanten von G zu zählen, wobei man abbrechen kann, wenn man mehr als $n-1$ Kanten findet, und, falls $m < n$, die Anzahl $c(G)$ der Zusammenhangskomponenten von G wie in (3.) zu bestimmen.
Ad 6. Per Definition ist ein Graph ein Baum genau dann, wenn er zusammenhängend und kreisfrei ist, was man mittels (2.) und (5.) prüft. □

Erkennung bipartiter Graphen. Um einen Graphen $G = (V, E)$ daraufhin zu untersuchen, ob er bipartit ist, könnte man entsprechend der Definition so vorgehen, daß man alle $\frac{1}{2}2^n$ 2-Partitionen von V durchtestet, ob die Kanten von G nur zwischen den Partitionsklassen verlaufen. Dies ist offensichtlich (für große Graphen) keine effiziente Strategie.

Proposition 1.3.5 *Bipartite Graphen können in Zeit $\mathcal{O}(n+m)$ erkannt werden.*

Beweis. Sei $G = (V, E)$ ein Graph. Da ein Graph bipartit ist genau dann, wenn jede seiner Komponenten bipartit ist, können wir G O.B.d.A. als zusammenhängend annehmen. Wir starten den Algorithmus Breitensuche in einem beliebigen Knoten $u \in V$ und „färben" im Verlauf der Suche jeden Knoten v, dem wir begegnen, mit der Farbe $dist(u, v)$ mod 2. Sei $V = U_0 \dot\cup U_1$ die so definierte Partition von V. Wir behaupten:

$$U_0 \dot\cup U_1 \textit{ ist eine Bipartition von } G \textit{ genau dann, wenn } G \textit{ bipartit ist.}$$

„$\Rightarrow$": Dies gilt nach Definition von „bipartit".
„$\Leftarrow$": Sei G bipartit mit Bipartition $V_0 \dot\cup V_1$. Sei o.B.d.A. $u \in V_0$. Dann ist leicht zu sehen, daß $U_0 = V_0$ und $U_1 = V_1$. (Insbesondere besitzen zusammenhängende bipartite Graphen also eine *eindeutige* Bipartition ihrer Knotenmenge.)

Also müssen wir in einem zweiten Schritt lediglich noch testen, ob die im ersten Schritt berechnete Partition $V = U_0 \dot\cup U_1$ tatsächlich eine Bipartition von G ist, indem wir für alle Kanten $\{x, y\} \in E$ prüfen, ob ihre Endknoten x und y verschiedene „Farben“ erhalten haben.
Da beide Schritte jeweils nur $\mathcal{O}(n+m)$ viele Rechenoperationen benötigen, ist die Laufzeit insgesamt linear. □

Tiefensuche. In vielen Fällen ist es unerheblich, in welcher Reihenfolge man die Knoten und Kanten eines Graphen abläuft – entscheidend ist, daß alle Knoten und Kanten (ggf. einer Zusammenhangskomponente) „besucht“ werden. Der Algorithmus TIEFENSUCHE (engl. Depth-First-Search oder kurz DFS), den wir nun kennenlernen werden, unterscheidet sich von BREITENSUCHE lediglich durch die Auswahlstrategie für die zu besuchenden Knoten und ist ein repräsentatives Beispiel für den *sinnvollen* Einsatz rekursiver Programmierung.

Die Knoten des Graphen G seien wie $V = \{1, \ldots, n\}$ durchnumeriert. Die Variable t zählt die Aufrufe der Prozedur TIEFENSUCHE. Alle Knoten in G werden bei ihrer „Entdeckung“ mit $bekannt[w] = TRUE$ markiert. Der Feldeintrag $Vor[w]$ vermerkt für alle bekannten Knoten $w \in V - s$, über welchen Knoten in G der Knoten w „entdeckt“ wurde. Ein ausgewählter Knoten $s \in V$ wird der Startknoten der Tiefensuche. Das Feld $DFSnum[\cdot]$ gibt nach Beendigung des Aufrufs TIEFENSUCHE(G, s) Auskunft darüber, in welcher Reihenfolge Knoten besucht wurden; es wird jedoch erst im Abschnitt 2.3 für uns von Bedeutung sein.

```
PROCEDURE TIEFENSUCHE (G, v);

BEGIN
  t := t + 1;
  DFSnum[v] := t;
  FOR w ∈ Γ(v) DO IF NOT bekannt[w] THEN BEGIN
    bekannt[w] := TRUE;
    Vor[w] := v;
    TIEFENSUCHE (G, w);
  END; {for}
END; {Tiefensuche}

BEGIN {Hauptprogramm}
  FOR v ∈ V DO DFSnum[v] := 0;
  FOR v ∈ V DO bekannt[v] := FALSE;
  FOR v ∈ V DO Vor[v] := v;
  t := 0;
  bekannt[s] := TRUE;
  TIEFENSUCHE (G, s);
END; {Hauptprogramm}
```

Schon bei TARRY (1895) kann man nachlesen, daß man mit Hilfe dieser Strategie wieder aus einem Labyrinth herausfindet, ohne eine Kante (d.h. einen Abschnitt zwischen zwei Kreuzungen) öfter als zweimal abzulaufen (wobei man hierbei natürlich die Suche abbrechen wird, sobald ein Ausgang gefunden ist).
Offenbar besucht Tiefensuche jeden Knoten der Zusammenhangskomponente von G, die

s enthält, denn sonst gäbe es eine Kante von einem bekannten Knoten v zu einem noch unbekannten Knoten w und der Aufruf TIEFENSUCHE (G, v) hätte nicht alle Nachbarn von v untersucht. Wie bei der Breitensuche lassen sich also die Zusammenhangskomponenten eines Graphen auch mittels Tiefensuche bestimmen. Ähnlich wie beim Algorithmus BREITENSUCHE ergibt sich:

Proposition 1.3.6 *Sei G ein zusammenhängender Graph und $s \in V(G)$.*

- *Die von einem Aufruf* TIEFENSUCHE (G, s) *berechnete Kantenmenge* $T := \{\{Vor[v], v\} : v \in V - s\}$ *bildet einen spannenden Baum von G.*
- *Ein Aufruf der Form* TIEFENSUCHE (G, s) *kostet Zeit $\mathcal{O}(n + m)$.* □

Anmerkungen und Übungen

Breitensuche hat ihre Ursprünge in Arbeiten von LEE [Lee61] und MOORE [Moo59], Tiefensuche in einer Arbeit von TARJAN [Tar72]. Man beachte, daß Tiefensuche erheblich weniger aufwendig zu programmieren ist als Breitensuche, da keine Warteschlange benötigt wird und der Compiler die Verwaltung des Rekursionsstacks übernimmt. Aus demselben Grunde ist Tiefensuche in der Praxis auch schneller als Breitensuche.

Übung 1.3.7 *Ein Graph G ist bipartit genau dann, wenn für alle Knoten $u \in V$ alle Sphären $S_i(u)$, $0 \leq i \leq n-1$, um u stabil sind.*

Übung 1.3.8 *Die Taillenweite $g(G)$ und ein kürzester Kreis in einem Graphen G lassen sich in Zeit $\mathcal{O}(nm)$ berechnen.*

Mehr zur Bestimmung der Taillenweite in [IR78].

Übung 1.3.9 a) *Ein bipartiter Graph hat $m \leq n^2/4$ viele Kanten.* **b)** *Ob ein Graph G das Komplement eines bipartiten Graphen ist, läßt sich in Zeit $\mathcal{O}(n+m)$ testen.* [Beachte: das Komplement kann $\Omega(n^2)$ viele Kanten enthalten.]

Übung 1.3.10 *Unter Verwendung von Tiefensuche gebe man $\mathcal{O}(n)$-Algorithmen, um kreisfreie Graphen bzw. Bäume zu erkennen, und einen $\mathcal{O}(n+m)$-Algorithmus, um bipartite Graphen zu erkennen, an.*

1.3.2 Dynamisches Programmieren: das kürzeste-Wege-Problem

Mit Hilfe von BREITENSUCHE ließen sich in einem Graphen G in Zeit $\mathcal{O}(n + m)$ die Abstände $dist(u, v)$, $v \in V$, von einem Knoten $u \in V$ berechnen. Anhand einer Verallgemeinerung dieses Problems auf (Kanten-) gewichtete Graphen soll in diesem Abschnitt die auf BELLMAN [Bel57] zurückgehende Technik der Dynamischen Programmierung (im Englischen steht der Begriff „programming“ in diesem Zusammenhang für „Optimierung“) vorgestellt werden, siehe auch [DL77]. Jeder Kante $e = \{u, v\} \in E$ sei nun ein Gewicht (eine Länge) $c(e) \in \mathbb{Q}_+$ zugeordnet, das sich in vielfältiger Weise als Überführungskosten zwischen den Zuständen u und v interpretieren läßt. Die Länge eines Weges $W = v_0, v_1, \ldots, v_\ell$ ist nun durch $c(W) := \sum_{i=1}^{\ell} c(\{v_{i-1}, v_i\})$ definiert. Das der Dynamischen Programmierung zugrundeliegende Optimalitätsprinzip besagt:

eine optimale Strategie enthält nur optimale Teilstrategien.

In unserem Fall heißt das: ist $u = v_0, v_1, \ldots, v_\ell = v$ ein kürzester $u - v-$Pfad, dann ist jeder Teilpfad $u = v_0, v_1, \ldots, v_i$, $i = 1, \ldots, \ell - 1$, ein kürzester $u - v_i-$Pfad. Natürlich wissen wir nicht, welche Knoten ein kürzester $u - v-$Pfad benutzt. Unsere Strategie heißt daher: konstruiere alle kürzesten Pfade, die in u beginnen! Typisch für die Technik der Dynamischen Programmierung ist nun, daß die gesuchten Strukturen in einer „bottom up"-Manier konstruiert werden, d.h. daß aus kleineren optimalen Lösungen größere aufgebaut werden. Als Datenstruktur werden hierbei charakteristischerweise Tabellen benutzt. Der folgende Algorithmus berechnet zu einem zusammenhängenden Graphen G mit Kantengewichten $c : E \to \mathbb{Q}_+$ und einem Startknoten $s \in V$ als Eingabe die Felder $d[v]$ und $Vor[v]$, $v \in V$, so daß schließlich $d[v] = dist(s, v)$ für alle $v \in V$ und $Vor[v]$ die gefundenen kürzesten $s - v-$Pfade, $v \in V - s$, codiert – indem $Vor[v]$ den Vorgänger von v auf dem gefundenen kürzesten $s - v-$Pfad enthält. Die Menge Γ stellt die „Grenzschicht" zwischen den bereits abgearbeiteten Knoten S und den noch nicht entdeckten Knoten des Graphen dar. Für $v \in \Gamma$ ist $d[v]$ sozusagen ein versuchsweiser Abstand von s (siehe unten).

```
PROCEDURE DIJKSTRA (G, s);

BEGIN
  FOR v ∈ V DO Vor[v] := v;
  FOR v ∈ V DO d[v] := ∞;
  d[s] := 0;
  S := ∅; Γ := {s};
  WHILE Γ ≠ ∅ DO BEGIN
    v := argmin {d[w] : w ∈ Γ};
    S := S + v; Γ := Γ − v;
    FOR w ∈ Γ(v) \ (S ∪ Γ) DO BEGIN
      Γ := Γ + w;
      IF d[v] + c({v, w}) < d[w] THEN BEGIN
        d[w] := d[v] + c({v, w});
        Vor[w] := v;
      END;
    END; {for}
  END; {while}
END;
```

Satz 1.3.11 (DIJKSTRA 1959) *Für einen zusammenhängenden, kantengewichteten Graphen $G = (V, E, c)$ und einen Startknoten $s \in V$ berechnet der Algorithmus* DIJKSTRA(G, s) *kürzeste $s - v-$Pfade für alle $v \in V - s$. Er kann mit Laufzeit $\mathcal{O}(n^2)$ implementiert werden.*

Beweis. Beachte zunächst, daß der erste Durchlauf durch die `WHILE`-Schleife lediglich S auf $\{s\}$, Γ auf $\Gamma(s)$ und $Vor[w] := s$ für alle $w \in \Gamma(s)$ setzt. Aufgrund des Zusammenhangs von G besucht diese Variante der Breitensuche jeden Knoten des Graphen, d.h. jeder Knoten gelangt mindestens einmal in die Menge Γ. Da S all die Knoten enthält, die aus Γ entfernt wurden, gelangt kein Knoten ein weiteres Mal in die Menge Γ. Insbesondere terminiert also die `WHILE`-Schleife.
Wir zeigen: die `WHILE`-Schleife besitzt nach ihrem ersten Durchlauf die folgenden Invarianten (es sei $\Gamma(S) := \bigcup_{s \in S} \Gamma(s)$):

(i) $\Gamma = \Gamma(S) \setminus S$;

(ii) für alle Knoten $v \in S$ ist $d[v]$ die Länge eines kürzesten $s-v$–Pfades in G, und für $v \in S - s$ ist $\{Vor(v), v\}$ die letzte Kante auf einem solchen Pfad;

(iii) für alle Knoten $v \in \Gamma$ ist $d[v]$ die Länge eines kürzesten $s-v$–Pfades in $G[S+v]$ und $Vor[v] \in S$ der Vorgänger von v auf einem solchen Pfad.

Da die Bedingungen (i) – (iii) unmittelbar nach dem ersten Durchlauf durch die `WHILE`-Schleife erfüllt sind, betrachten wir die Situation, daß ein Knoten $v = argmin\{d[w] : w \in \Gamma\}$ in S aufgenommen werden soll. Nach Induktionsannahme (iii) ist $d[v]$ gleich der Länge eines kürzesten $s-v$–Pfades, der ganz in $S+v$ verläuft (und durch die Funktion $Vor[\cdot]$ ist ein solcher codiert). Sei nun P' ein beliebiger $s-v$–Pfad in G, der nicht ganz innerhalb der Menge $S+v$ verläuft. Sei $v' \neq v$ der erste Knoten auf P' in $V-(S+v)$. Dann gilt wegen (i) $v' \in \Gamma$ und somit nach Wahl von v: $d[v'] \geq d[v]$. Aufgrund der Positivität der Kantengewichte und nach Induktionsvoraussetzung (iii) folgt daraus

$$c(P') \;>\; c(P'_{[s,v']}) \;\geq\; d[v'] \;\geq\; d[v].$$

Also verläuft jeder kürzeste $s-v$–Pfad innerhalb von $S+v$ und hat Länge $d[v]$. Damit gilt (ii) ebenso für $S' = S+v$.
Der Update innerhalb der `FOR`-Schleife stellt zudem die Bedingungen (i) und (iii) auch für $S' = S+v$ sicher.

Hinsichtlich der Laufzeit beachte man, daß die `WHILE`-Schleife genau n-mal durchlaufen wird (bis $S = V$ und also $\Gamma = \emptyset$). Die Mengen S und Γ seien als Felder von Wahrheitswerten codiert. Der gesamte Aufwand für die Bestimmung der $argmin$'s ist daher wegen $|\Gamma| \leq n-1$ sicher $\mathcal{O}(n^2)$. Die `FOR`-Schleife hingegen wird insgesamt höchstens $\sum_{v \in V} |\Gamma(v)| = 2m = \mathcal{O}(n^2)$-mal mit jeweils konstantem Aufwand durchlaufen. □

Übungen und Anmerkungen

Übung 1.3.12 *Durchmesser und Zentrum können auf Bäumen in Zeit $\mathcal{O}(n)$ bestimmt werden.*

Übung 1.3.13 *Für einen zusammenhängenden Graphen G und einen Knoten $s \in V$ ist die durch die Ausgabe des Algorithmus* DIJKSTRA *definierte Kantenmenge $T := \{\{Vor(v), v\} : v \in V - s\}$ ein spannender Baum von G mit $dist_T(s,v) = dist_G(s,v)$ für alle $v \in V$.*

Unter Verwendung der Datenstruktur der FIBONACCI-Heaps erreicht der Algorithmus von DIJKSTRA eine Laufzeit von $\mathcal{O}(m + n \log n)$ [FT87] (siehe auch [CLR90]), was in einem bestimmten Modell sogar optimal ist, siehe [Kin90]. In [AMOT90] findet man eine $\mathcal{O}(m + n\sqrt{\log C})$-Implementierung bei nicht-negativen, ganzzahligen Gewichten $c(e)$, wobei $C := \max_{e \in E} c(e)$.

Kürzeste-Wege-Probleme werden erheblich schwieriger in dem allgemeineren Fall, daß die Kantengewichte des Graphen beliebige (insbesondere negative) rationale Zahlen sind. In diesem Fall ist nicht einmal die Existenz kürzester Wege gesichert, weil Kreise mit negativem Gewicht auftreten können. Der $\mathcal{O}(nm)$-Algorithmus nach MOORE, BELLMAN und FORD wird in [CLR90] ausführlich behandelt. Einen Skalierungsalgorithmus mit Laufzeit $\mathcal{O}(\sqrt{n}m \log N)$, wobei N das betragsmäßig größte negative Kantengewicht bezeichnet, gab GOLDBERG [Gol95]. Zur Berechnung der kürzesten Pfade zwischen *allen* Knotenpaaren eines solchen Graphen gibt es einen auf D.B. JOHNSON zurückgehenden $\mathcal{O}(n^2 \log n + nm)$-Algorithmus, siehe wieder [CLR90]. In [FG85, MT87] findet man Algorithmen für dieses Problem, deren erwartete Laufzeit (d.h. gemittelt über alle Graphen der Ordnung n) $\mathcal{O}(n^2 \log n)$ ist. Für die Berechnung kürzester Wege in ungerichteten Graphen mit nicht notwendig positiver Gewichtsfunktion siehe [Jun94, Abschnitt 13.6]. Experimentelle Ergebnisse findet man in [GP88, CGR94].

1.3.3 Greedy: minimale aufspannende Bäume

Der Algorithmus BREITENSUCHE lieferte uns einen aufspannenden Baum in einem zusammenhängenden Graphen. Betrachten wir nun wieder eine kantengewichtete Variante. Die Problemstellung heißt jetzt, einen aufspannenden Baum $T \subseteq E$ minimalen Gewichts $c(T) := \sum_{e \in T} c(e)$ in einem kantengewichteten Graphen $G = (V, E, c)$ zu finden. Anwendungen dieses Problems kann man sich beispielsweise beim Entwurf von kostengünstigen Telefon- oder Straßennetzen zwischen vorgegebenen Orten denken.

Proposition 1.3.14 (KRUSKAL 1956) *Sei $G = (V, E, c)$ ein zusammenhängender, durch $c : E \to \mathbb{Q}^+$ gewichteter Graph. Der folgende Algorithmus* MST-KRUSKAL *findet einen minimal aufspannenden Baum $T \subseteq E$ in G und kann mit Laufzeit $\mathcal{O}(m \log n)$ implementiert werden.*

```
PROCEDURE MST-KRUSKAL;

BEGIN
  sortiere E, so daß c(e_1) ≤ ... ≤ c(e_m);
  T := ∅;
  FOR i := 1 TO m DO
    IF T ∪ {e_i} kreisfrei THEN  T := T ∪ {e_i};
END;
```

Beweis. Der Graph (V, T) ist anfangs der leere Graph mit n Komponenten: die einzelnen Knoten. Durch das Hineinnehmen von Kanten werden dann sukzessive jeweils genau zwei verschiedene Komponenten von (V, T) verschmolzen, denn für eine Kante $e = \{x, y\} \notin E$ ist $T + e$ ist genau dann kreisfrei, wenn x und y in verschiedenen Komponenten von (V, T) liegen. Mithin werden höchstens $n - 1$ Kanten in T aufgenommen. Wir zeigen, daß (V, T) schließlich zusammenhängt: Da G zusammenhängt, ist in G kein Schnitt $\langle A, B \rangle$ leer. Also nimmt der Algorithmus MST-KRUSKAL die (bezüglich der Ordnung $c(e_1) \leq \cdots \leq c(e_m)$) erste Kante aus $\langle A, B \rangle$ in T auf, denn sie kann in (V, T) keinen Kreis schließen. Also ist auch in (V, T) kein Schnitt leer, d.h. der Graph (V, T) hängt schließlich zusammen. Insbesondere enthält T somit mindestens $n - 1$ Kanten. Da der Graph (V, T) nach Konstruktion kreisfrei ist, stellt nach Satz 1.1.26(3) die Ausgabe T des Algorithmus MST-KRUSKAL einen spannenden Baum von G dar.

Angenommen nun, $T = \{e_{i_1}, \ldots, e_{i_{n-1}}\}$ sei kein *minimal* aufspannender Baum. Dann gibt es für jeden minimal spannenden Baum T^* in G ein $k \in \{1, \ldots, n-1\}$, so daß die Kante e_{i_k} die erste Kante in T ist, die nicht auch in T^* enthalten ist. Sei T^* ein minimal spannender Baum in G mit k größtmöglich. Es gilt also $\{e_{i_1}, \ldots, e_{i_{k-1}}\} \subseteq T \cap T^*$ und $e_{i_k} \notin T^*$. Der Graph $T^* + e_{i_k}$ besitzt genau einen Kreis C. Da T kreisfrei ist, gibt es eine Kante $e_{i_C} \in C$, die zu T^* gehört, nicht aber zu T. Wegen $e_{i_C} \in C$ ist der Graph $T^* + e_{i_k} - e_{i_C}$ wieder kreisfrei und zusammenhängend, also ein spannender Baum. Da $\{e_{i_1}, \ldots, e_{i_{k-1}}, e_{i_C}\} \subseteq T^*$ zwar kreisfrei ist, der Algorithmus MST-KRUSKAL aber im k-ten Schritt die Kante e_k und nicht e_{i_C} in T aufgenommen hat, folgt $i_C > i_k$ und daraus $c(e_{i_C}) \geq c(e_{i_k})$. Mithin gilt $c(T^* + e_{i_k} - e_{i_C}) \leq c(T^*)$, d.h. $T^* + e_{i_k} - e_{i_C}$ ist ebenfalls ein minimal spannender Baum. Doch hat $T^* + e_{i_k} - e_{i_C}$ die Kanten $\{e_{i_1}, \ldots, e_{i_k}\}$ mit T gemein, im Widerspruch zur Wahl von k und T^*.

Hinsichtlich der Laufzeit beachte man, daß das Sortieren der m Kanten von G nach Gewichten in Zeit $\mathcal{O}(m \log m) = \mathcal{O}(m \log n)$ geschehen kann, siehe z.B. [Wir83, CLR90,

OW93]. Die Gesamtzahl der Operationen in der `WHILE`-Schleife hingegen ist sicherlich $\mathcal{O}(m)$, wenn wir den Test auf Kreisfreiheit in konstanter Zeit ausführen können. Dies läßt sich z.B. bewerkstelligen, indem man ein Feld $Komp : V \to V$ unterhält, so daß $Komp[v]$ jeweils einen (eindeutigen) Repräsentanten der Komponente, in der v liegt, angibt (dieser Repräsentant dient also als Name für die Komponente). Dann ist nämlich $T \cup \{u, v\}$ kreisfrei genau dann, wenn $Komp[u] \neq Komp[v]$. Das Feld $Komp$ wird zu Beginn mit $Komp[v] := v$ für alle $v \in V$ initialisiert. Die Frage ist, wie dieses Feld zu aktualisieren ist, wenn eine Kante $e_i = \{u, v\}$ keinen Kreis schließt und also in den Baum (V, T) aufgenommen wird und mithin die Komponenten $Komp[u]$ und $Komp[v]$ verschmolzen werden müssen. Die folgende Datenstruktur erlaubt es, die Menge der Komponenten in Gesamtlaufzeit (d.h. über den gesamten Verlauf des Algorithmus MST-Kruskal gerechnet) $\mathcal{O}(n \log n)$ zu verwalten.

Für jede Komponente $Komp[v]$ von (V, T) unterhalten wir eine (einfach verkettete) Liste $KompListe[Komp[v]]$ aller Knoten, die sie enthält. Zu Beginn haben wir also n Listen $Komp[v]$, $v \in V$, die jeweils genau den Knoten v enthalten. Ein weiteres Feld $KompGröße[v]$ gibt die Anzahl der Knoten in der Komponente, deren Repräsentant v ist, an und wird mit 1 für alle $v \in V$ vorbelegt.

Beim Verschmelzen zweier Komponenten $Komp[u]$ und $Komp[v]$ gehen wir nun naheliegenderweise so vor, daß wir allen Knoten in der *kleineren* der beiden Komponenten (sagen wir $Komp[v]$) den Namen der größeren Komponente, also $Komp[u]$, zuweisen sowie die größere Liste $KompListe[Komp[u]]$ an die kleinere Liste $KompListe[Komp[v]]$ anhängen. Hierzu ist lediglich die kleinere Liste $KompListe[Komp[v]]$ einmal zu durchlaufen. Die hierbei durchgeführten Operationen zählen wir nun pro Knoten. Jedesmal, wenn ein Knoten v die Komponente wechselt, verdoppelt sich die Größe der Komponente, zu der er gehört. Nach k Wechseln gilt also:

$$KompGröße[Komp[v]] \geq 2^k,$$

d.h. ein Knoten wechselt nur höchstens $\log n$-mal die Komponente. Die Anzahl der Operationen, die mit dem Umbenennen von $Komp[\cdot]$ verbunden sind, ist damit $\mathcal{O}(n \log n)$. Schließlich wird genau $(n-1)$-mal eine Liste an eine andere angehängt und der Wert $KompGröße[Komp[u]]$ aktualisiert (mit jeweils konstantem Aufwand).

Kritisch für die Gesamtlaufzeit des Algorithmus MST-Kruskal ist also das Sortieren der m Kanten nach ihren Gewichten. □

Die naive Regel, stets die lokal günstigste Möglichkeit zu wählen, nennt man das *Greedy-Prinzip* (engl. greedy = gierig). Natürlich *muß* diese Vorgehensweise nicht notwendig zu einer (global) optimalen Lösung führen, wie es hier der Fall war (vgl. zum Beispiel das Problem Clique); es ist sogar eher die Ausnahme. Die Strukturen, auf denen der Greedy-Algorithmus die Optimallösung liefert, wurden in [HMS93] charakterisiert. Trotz (oder gerade wegen) seiner Einfachheit ist das Greedy-Prinzip jedoch eines der wichtigsten algorithmischen Prinzipien und wird uns insbesondere in den Abschnitten 5.2 und 12.4 und Kapitel 10 wieder begegnen.

Anmerkungen und Übungen

Die Geschichte des Problems des minimum spannenden Baumes ist in [GH85] dargestellt. Ein weiterer verbreiteter Algorithmus für mimimum spannende Bäume geht auf Jarník [Jar30] zurück

und wurde von PRIM [Pri57] und DIJKSTRA [Dij59] wiederentdeckt. Für dichte Graphen (mit $\Omega(n^2)$ vielen Kanten) ist er sogar effizienter als der Algorithmus von KRUSKAL:

Übung 1.3.15 *Die durch die Ausgabe des Algorithmus von* DIJKSTRA *definierte Kantenmenge* $T := \{\{Vor(v), v\} : v \in V - s\}$ *stellt einen minimum spannenden Baum von* G *dar.*

Tatsächlich kann man minimum spannende Bäume asymptotisch in „fast linearer" Zeit konstruieren, siehe [FT87, GGST86, KKT95].

Übung 1.3.16 *Der Algorithmus von* KRUSKAL *konstruiert auf der umgekehrten Reihenfolge der Kanten einen maximum spannenden Baum von* G *(beachte: alle Kantengewichte sind positiv).*

Übung 1.3.17 *Sei* $G = (V, E, c)$ *ein zusammenhängender, kantengewichteter Graph und* T *ein spannender Baum von* G*. Für jede Kante* $e \in E \setminus E(T)$ *bezeichne* $C(e)$ *den (eindeutig bestimmten) Kreis in* $T + e$*. Dann gilt:* T *ist ein minimaler spannender Baum von* G *genau dann, wenn er nur lokal minimal ist, d.h. wenn für alle* $e \in E \setminus E(T)$ *und alle* $f \in C(e) - e$ *gilt:* $w(e) \geq w(f)$*.*

Übung 1.3.18 *Sei* $G = (V, E, c)$ *ein zusammenhängender, kantengewichteter Graph. Der Graph auf der Menge* $\mathcal{T}$ *aller spannenden Bäume von* G*, in dem zwei spannende Bäume* T' *und* T'' *von* G *benachbart sind genau dann, wenn* $|E(T')\Delta E(T'')| = 2$*, heißt der* Spannbaumgraph $\mathcal{T}(G)$*.*
a) $\mathcal{T}(G)$ *ist zusammenhängend mit Diameter* $diam(\mathcal{T}(G)) \leq n - 1$*.*
b) *Die Menge der minimalen spannenden Bäume von* G *induziert einen zusammenhängenden Subgraphen von* $\mathcal{T}(G)$*.*

1.4 Die Klasse NP oder: polynomielle Verifizierbarkeit

Für eine große Klasse wichtiger Probleme, eben die NP-Probleme, ist die Frage, ob sie alle einen polynomiellen Algorithmus besitzen (können), ungeklärt. Immerhin kann man die Probleme dieser Klasse untereinander bezüglich ihres Schwierigkeitsgrades vergleichen und so die schwersten unter ihnen ausfindig machen.
Die Klasse NP steht für die Menge der Sprachen (Entscheidungsprobleme), die unter Annahme eines nichtdeterministischen Berechnungsmodells, eben der sogenannten nichtdeterministischen Turing Maschine (NTM), in polynomieller Zeit erkannt werden können. Wir benutzen zur Definition von NP folgenden auf [Coo71] zurückgehenden Begriff.

Definition 1.4.1 *Eine binäre Relation* $R \subseteq \Sigma^* \times \Sigma^*$ *heißt* p-Relation, *falls sie schnell geprüft werden kann, d.h. falls gilt:*

(i) es existiert ein Polynom q*, so daß* $\langle x, z\rangle \in R \Rightarrow |z| \leq q(|x|)$;

(ii) das Prädikat $\langle x, z\rangle \in R$ *kann in (in* $|x|$*) polynomieller (deterministischer) Zeit getestet werden.*

Wir interpretieren dabei $x \in \Sigma^*$ als Probleminstanz eines Entscheidungsproblems Π und die Menge $R(x) := \{z \in \Sigma^* | \langle x, z\rangle \in R\}$ als die Menge der „Lösungen" zur Instanz x, wobei wir übereinkommen, daß diese wieder über demselben Alphabet Σ codiert sind.

Beispiel. Das Entscheidungsproblem CLIQUE wird durch die Relation R repräsentiert, die einer Codierung x eines Tupels $\langle G, k\rangle$ die Menge aller Codierungen z von Cliquen im Graphen G der Größe mindestens k zuordnet. Die Bedingungen (i) und (ii) sind dann trivialerweise erfüllt: gegeben ein Paar $\langle x, z\rangle \in \Sigma^* \times \Sigma^*$, kann man offenbar (leicht) in

polynomieller Zeit überprüfen, ob x einen Graphen G und eine natürliche Zahl k und z eine Clique C in G codiert mit $|C| \geq k$. □

Eine p-Relation definiert in natürlicher Weise eine Sprache

$$L_R := \{x \in \Sigma^* : \exists z \in \Sigma^*, \text{ so daß } \langle x, z\rangle \in R\}$$

und damit ein Entscheidungsproblem Π_R, nämlich das Problem, zu einer Instanz $x \in \Sigma^*$ zu entscheiden, ob $x \in L_R$. Die Elemente $z \in R(x)$ für eine p-Relation R heißen daher auch *Zertifikate* für x, da anhand eines $z \in R(x)$ wegen (ii) in polynomieller Zeit entschieden werden kann, ob $x \in Y_{\Pi_R} \Leftrightarrow x \in L_R$, d.h. ob die Antwort auf das Entscheidungsproblem Π_R für die Instanz x *JA* lautet.

Mit einer p-Relation $R \subseteq \Sigma^* \times \Sigma^*$ lassen sich folgende Probleme assoziieren (vgl. [JVV86]):

- **Existenz** (auch Entscheidungs- oder Sprachenzugehörigkeitsproblem)
  gegeben ein $x \in \Sigma^*$, entscheide, ob $R(x) \neq \emptyset$.
- **Konstruktion** (auch Suchproblem)
  Ist $R(x) = \emptyset$, gib $\emptyset$, sonst ein (beliebiges) $z \in R(x)$ aus.
- **Erzeugen** (auch Generieren oder Sampling)
  Ist $R(x) = \emptyset$, gib $\emptyset$ aus, sonst ein zufälliges Element $z \in R(x)$ mit Wahrscheinlichkeit $1/|R(x)|$.
- **Zählen**
  Bestimme die Anzahl aller Lösungen $\#R(x) := |R(x)|$.

Offensichtlich reduziert sich das Existenzproblem auf die übrigen. Wir beschäftigen uns hier mit dem Existenz- und dem Suchproblem für p-Relationen.

Definition 1.4.2 *Eine Sprache $L \subseteq \Sigma^*$ gehört zur Klasse* **NP** *(von engl.* n*on-deterministic* p*olynomial time), falls eine p-Relation R existiert, so daß*

$$L = L_R = \{x \in \Sigma^* : \exists z \in \Sigma^*, \text{ so daß } \langle x, z\rangle \in R\}$$

Ein Entscheidungsproblem Π *gehört zu* NP*, falls bezüglich einer („vernünftigen") Codierung die Sprache Y_Π seiner JA-Instanzen zu* NP *gehört.*

Die Klasse NP besteht somit aus mit p-Relationen assoziierten Existenzproblemen. Dies bedeutet, daß NP die Entscheidungsprobleme Π enthält, bei denen im Falle einer Probleminstanz $x \in Y_\Pi$ dafür ein kurzer Beweis $z \in R_\Pi(x)$ existiert, den man durch Nachweis von $\langle x, z\rangle \in R_\Pi$ in polynomieller Zeit prüfen oder verifizieren kann. Daher ist das Konstrukt der NTM weniger Abstraktion eines Algorithmus als ein Hilfsmittel, *polynomielle Verifizierbarkeit* zu beschreiben, und wir stellen uns unter einer NTM einen sogenannten *Zertifikat-Prüfalgorithmus*[17] vor, der einen kurzen Beweis dafür, daß die Antwort *JA* lautet, gewissermaßen „raten" darf, bevor er wie eine gewöhnliche polynomielle DTM fortfährt und ihn verifiziert.[18]

[17] von engl. certificate-checking-algorithm, siehe [PS82]

[18] Die NTM darf dazu zu allen $y \in \Sigma^{p(x)}$ verzweigen (dies ist das nichtdeterministische Element) und parallel testen, ob $\langle x, y\rangle \in R$. Genau die $y \in R_\Pi(x)$ führen zu akzeptierenden Berechnungen der NTM. Die *Laufzeitfunktion* $T_M(n)$ einer NTM M bestimmt sich dabei durch die minimale Anzahl Schritte, die sie für eine akzeptierende Berechnung zur Eingabe x benötigt, maximiert über alle Eingaben $x \in \Sigma^*$ der Länge $|x| = n$ (bzw. $= 1$, falls es keine akzeptierende Berechnung gibt).

Beispiele. Das Entscheidungsproblem CLIQUE ist offensichtlich in NP, denn ein Graph G hat Cliquenzahl $\omega(G) \geq k$ genau dann, wenn es mindestens eine Clique C in G der Größe mindestens k gibt. Gegeben eine solche Clique C, ist es offenbar ein leichtes, $\langle G, C\rangle \in$ CLIQUE in polynomieller Zeit zu verifizieren, d.h. daß zum einen C wirklich eine Clique in G ist und zum anderen $|C| \geq k$.

Ein „klassisches" Entscheidungsproblem ist das *Erfüllbarkeitsproblem*, abgekürzt SAT (für engl. satisfiability). Seien $x_1, \ldots, x_n$ Boolesche Variablen, d.h. Variablen, die die Werte 0 oder 1 annehmen können. Die Booleschen Variablen $x_1, \ldots, x_n$ zusammen mit ihren Komplementen $\overline{x_1}, \ldots, \overline{x_n}$ heißen *Literale*. Eine Klausel ist eine Oder-Verküpfung (Disjunktion) von (nicht notwendig verschiedenen) Literalen. Beim Erfüllbarkeitsproblem SAT ist nun ein Boolescher Ausdruck F in konjunktiver Normalform (CNF) gegeben, d.h. eine Konjunktion (logische Und-Verknüpfung) von $m \in \mathbb{N}$ Klauseln. Hier ein Beispiel mit $n = 4$ und $m = 3$:

$$F \quad = \quad x_4 \wedge (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_4}).$$

Gefragt wird danach, ob F erfüllbar ist, d.h. ob es eine Belegung der Variablen mit den Wahrheitswerten *wahr* (=1) oder *falsch* (=0) gibt, so daß F wahr wird. SAT ist also die Sprache der erfüllbaren Booleschen Ausdrücke in CNF.
Offenbar liegt auch SAT in NP, denn wenn ein Boolescher Ausdruck F erfüllbar ist, so kann dies offenbar anhand eines entsprechenden Zertifikats (eben einer erfüllenden Variablenbelegung) in polynomieller Zeit durch Einsetzen überprüft werden. □

Man beachte, daß Probleme in NP trivialerweise immer in exponentieller Zeit entscheidbar sind: sei p ein Polynom, das die Laufzeit der zugehörigen NTM (des Zertifikat-Prüfalgorithmus) beschränkt. Alle Zertifikate für ein $x \in \Sigma^*$ haben o.B.d.A. genau die Länge $q(|x|)$. Dann kann man offenbar in Zeit $\mathcal{O}(p(n)|\Sigma|^{q(n)})$ alle möglichen Zertifikate einer Instanz x mit $|x| = n$ durchprüfen.

1.4.1 NP-Vollständigkeit

Beim Vergleich der beiden Klassen P und NP spielt der Begriff der NP-Vollständigkeit eine Schlüsselrolle. Ihm liegt das von COOK [Coo71] und KARP [Kar72] entwickelte Konzept der Reduzierbarkeit zugrunde, das aus der Theorie der rekursiven Funktionen entlehnt ist. Für die Definition der NP-Vollständigkeit hat sich die auf KARP zurückgehende Form der polynomiellen m(any-one)-Reduzierbarkeit, auch Transformierbarkeit, durchgesetzt.

Definition 1.4.3 *Ein Entscheidungsproblem* Π *(formal eine Sprache) heißt* polynomiell reduzierbar *auf ein Entscheidungsproblem* Π'*, in Zeichen* $\Pi \leq_m \Pi'$*, falls es eine polynomiell berechenbare Funktion* $f : \Sigma^* \to \Sigma^*$ *gibt, so daß gilt*

$$x \in Y_\Pi \quad \Leftrightarrow \quad f(x) \in Y_{\Pi'} \qquad \forall\, x \in \Sigma^*.$$

$\Pi \leq_m \Pi'$ deutet also an, daß das Problem Π nicht schwerer ist als Π', da eine Instanz $x \in D_\Pi$ anhand der Instanz $f(x)$ des Problems Π' effizient entschieden werden kann, wenn dieses effizient lösbar ist.
Die Relation der *wechselseitigen* Reduzierbarkeit zweier Probleme ist transitiv und induziert auf der Klasse aller Probleme in NP eine Äquivalenzrelation. Die Relation „$\leq_m$" induziert wiederum eine teilweise Ordnung auf den resultierenden Äquivalenzklassen, auch *p-degrees* [Lad75], von Entscheidungsproblemen. Die unterste Äquivalenzklasse ist P, sie

enthält die einfachsten Probleme. Die schwersten Probleme in NP werden beschrieben durch die

Definition 1.4.4 *Ein Entscheidungsproblem* Π *heißt* NP-vollständig *(bezüglich „*$\leq_m$*“), falls* $\Pi \in \mathrm{NP}$ *und*

$$\mathrm{NP} \leq_m \Pi \quad :\Leftrightarrow \quad \Pi' \leq_m \Pi \qquad \forall\, \Pi' \in \mathrm{NP}.$$

Aus der Definition ergibt sich unmittelbar die

Proposition 1.4.5 *Für ein* NP*-vollständiges Problem* Π *gilt:* $\Pi \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$. □

D.h. wenn ein einziges NP-vollständiges Problem in polynomieller Zeit gelöst werden kann, so können es alle Probleme in NP. Ist umgekehrt aber ein beliebiges Problem in NP unzugänglich, also schwer, dann sind auch alle NP-vollständigen Probleme unzugänglich. Entweder gehören also alle NP-vollständigen Probleme zu P oder keines. Ein vollständiges Problem repräsentiert mithin die ganze Klasse, der es angehört; es sind im Sinne der Reduzierbarkeit die schwersten Probleme der Klasse. In seiner epochalen Arbeit [Coo71] zeigte Cook die Existenz von NP-vollständigen Problemen:[19]

Satz 1.4.6 (Cook 1971) Sat *ist* NP*-vollständig.* □

Da wir keine Maschinenmodelle einführen wollen, können wir hier leider nicht auf den Beweis eingehen. Man kann ihn z.B. in [GJ79, Weg93, Pap94] nachlesen.

Mit Hilfe der Kenntnis eines NP-vollständigen Problems als „Referenzproblem“ ist es nun nicht mehr schwer, weitere Probleme in NP als NP-vollständig auszuweisen. Aufgrund der Transitivität der Relation „$\leq_m$“ genügt es offenbar, ein bekanntes NP-vollständiges Problem wie Sat auf das betrachtete, neue Problem zu reduzieren. Dies tat Karp (1972) für eine erste Liste von 20 klassischen Problemen der Kombinatorik. Seitdem konnte die NP-Vollständigkeit einer Vielzahl von Entscheidungsproblemen aus Gebieten wie Graphentheorie, Spieltheorie, Zahlentheorie, Operations Research, Logik und Kombinatorik nachgewiesen werden, siehe [GJ79] und [Joh90a]. Dadurch wird die Bedeutung sowohl des Konzepts der Reduzierbarkeit wie auch einer solchen Klassifikation für ein bestimmtes Problem evident: Da es den verschiedenen Disziplinen der Mathematik und Informatik bisher nicht gelungen ist, für *irgendein* NP-vollständiges Problem einen polynomiellen Algorithmus zu entwerfen, wird man solche Probleme als *hartnäckig* (engl. apparently intractable) ansehen in dem Sinn, daß sie *vermutlich* von einer inhärenten Komplexität sind, die sie einer Lösung durch einen polynomiellen (deterministischen) Algorithmus unzugänglich macht ($\mathrm{P} \neq \mathrm{NP}$ Hypothese). Man beachte aber: genausowenig konnte man bisher beweisen, daß sie tatsächlich unzugänglich sind! Alle entscheidbaren Probleme, für die man Unzugänglichkeit nachweisen konnte, liegen nicht in NP. Darüber hinaus muß ein etwaiger Beweis von P=NP nicht notwendig zu *effizienten* Algorithmen für NP-vollständige Probleme führen. Die Frage $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$: ob also nichtdeterministische Algorithmen tatsächlich mächtiger als ihre deterministischen Pendants sind, oder anders formuliert, ob alle Eigenschaften (Entscheidungsprobleme), die nur ein schnell zu verifizierendes Zertifikat besitzen (d.h. in NP liegen), schon effizient entscheidbar (d.h. polynomiell lösbar) sind, ist eines der zentralen ungelösten Probleme der Theoretischen Informatik.[20]

[19] Ein ähnliches Konzept hat unabhängig Levin [Lev73] entwickelt.

[20] Geschichte und aktueller Stand des $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ Problems werden in [Sip92] diskutiert.

Die Vorgehensweise bei Reduzierbarkeitsbeweisen illustrieren wir an dem wichtigen Entscheidungsproblem 3-SAT, bei dem nach der Erfüllbarkeit einer Booleschen Formel in CNF mit genau drei Literalen pro Klausel gefragt ist; in jeder Klausel trete dabei keine Variable (in positiver oder negierter Form) mehrfach auf.

Satz 1.4.7 (COOK 1971) 3-SAT *ist* NP*-vollständig.*

Beweis nach [Wel93]. Daß schon dieser Spezialfall von SAT NP-schwer ist, zeigen wir durch Reduktion von SAT (auf 3-SAT).[21] Sei $F = \bigwedge_{j=1}^{m} C_j$ also eine Instanz von SAT, d.h. eine Boolesche Formel in CNF über n Booleschen Variablen $x_1, \ldots, x_n$ mit den Klauseln $C_1, \ldots, C_m$. O.B.d.A. enthält keine Klausel einen Literal mehrfach. Wir transformieren F zu einer Instanz $\tilde{F}$ von 3-SAT. Dazu verschaffen wir uns zunächst drei Boolesche Variable z_1, z_2, z_3, die in jeder Wahrheitsbelegung von $\tilde{F}$ *falsch* sind, indem wir $\tilde{F}$ folgende sieben Klauseln voranstellen:

$$\begin{array}{ll} \overline{z_1} \vee \overline{z_2} \vee \overline{z_3} & \\ z_1 \vee \overline{z_2} \vee \overline{z_3} & z_1 \vee z_2 \vee \overline{z_3} \\ z_2 \vee \overline{z_3} \vee \overline{z_1} & z_2 \vee z_3 \vee \overline{z_1} \\ z_3 \vee \overline{z_1} \vee \overline{z_2} & z_3 \vee z_1 \vee \overline{z_2}. \end{array}$$

Ihre Konjunktion ist, wie man sich leicht überzeugt, gerade äquivalent mit $\overline{z_1} \wedge \overline{z_2} \wedge \overline{z_3}$. Damit können wir jede Klausel von F, die weniger als drei Literale enthält, durch Disjunktion mit z.B. z_1 bzw. $z_1 \vee z_2$ zu einer „äquivalenten" Klausel mit genau drei Literalen erweitern.

Angenommen, eine Klausel $C_j = w_1 \vee \ldots \vee w_\ell$ von F enhält $\ell \geq 4$ Literale $w_1, \ldots, w_\ell$ (o.B.d.A. $\ell \leq n$). Dann ersetzen wir den Term $w_{\ell-1} \vee w_\ell$ von C_j in $\tilde{F}$ durch eine neu einzuführende Variable y und fügen dafür zu $\tilde{F}$ (konjunktiv) Klauseln (mit jeweils genau drei Literalen) hinzu, die $y \equiv w_{\ell-1} \vee w_\ell$ sicherstellen. Der erhaltene Boolesche Ausdruck hat dann offenbar dieselben erfüllenden Wahrheitswertebelegungen (wobei der jeweilige Wahrheitswert von y durch $y := w_{\ell-1} \vee w_\ell$ gegeben ist), aber die Klausel C_j ist um einen Literal verkürzt worden. Durch wiederholtes Anwenden (höchstens $m \cdot (n-3)$ mal) wird in polynomieller Zeit eine äquivalente 3-SAT-Formel konstruiert.

Hinsichtlich der in jedem Schritt hinzuzufügenden Klauseln, bemerken wir, daß der folgende Boolesche Ausdruck genau dann wahr ist, wenn $y \equiv w_{\ell-1} \vee w_\ell$ gilt:

$$(w_{\ell-1} \wedge w_\ell \wedge y) \;\vee\; (w_{\ell-1} \wedge \overline{w_\ell} \wedge y) \;\vee\; (\overline{w_{\ell-1}} \wedge w_\ell \wedge y) \;\vee\; (\overline{w_{\ell-1}} \wedge \overline{w_\ell} \wedge \overline{y})$$

Um ihn auf konjunktive Normalform (mit genau drei Literalen pro Klausel) zu bringen, beachte man, daß die Variablen $w_{\ell-1}$, w_ℓ und y genau $2^3 = 8$ verschiedene Wertebelegungen besitzen. Daher ist er äquivalent zu:

$$\begin{array}{ll} \equiv & \overline{(w_{\ell-1} \wedge w_\ell \wedge \overline{y}) \vee (w_{\ell-1} \wedge \overline{w_\ell} \wedge \overline{y}) \vee (\overline{w_{\ell-1}} \wedge w_\ell \wedge \overline{y}) \vee (\overline{w_{\ell-1}} \wedge \overline{w_\ell} \wedge y)} \\ \equiv & \overline{w_{\ell-1} \wedge w_\ell \wedge \overline{y}} \;\wedge\; \overline{w_{\ell-1} \wedge \overline{w_\ell} \wedge \overline{y}} \;\wedge\; \overline{\overline{w_{\ell-1}} \wedge w_\ell \wedge \overline{y}} \;\wedge\; \overline{\overline{w_{\ell-1}} \wedge \overline{w_\ell} \wedge y} \\ \equiv & (\overline{w_{\ell-1}} \vee \overline{w_\ell} \vee y) \wedge (\overline{w_{\ell-1}} \vee w_\ell \vee y) \wedge (w_{\ell-1} \vee \overline{w_\ell} \vee y) \wedge (w_{\ell-1} \vee w_\ell \vee \overline{y}) \end{array}$$

[21] Im Gegensatz zur Reduktion in [GJ79] ist diese Reduktion *bijektiv* (engl. parsimonious), d.h. sie erhält die Anzahl der Zertifikate:

$$|R_\Pi(x)| \;=\; |R_{\Pi'}(f(x))| \qquad \forall x \in \Sigma^*. \tag{1.5}$$

Dies zeigt, daß das zu 3-SAT korrespondierende Zählproblem #3-SAT, die erfüllenden Belegungen einer 3-SAT-Formel zu zählen, #P-vollständig ist, da #SAT #P-vollständig ist (wie eine bijektive Variante der generischen (m-)Reduktion aus COOKs Beweis zeigt [Sim75]). Die Klasse #P [Val79] besteht aus den Problemen, für eine p-Relation die Anzahl der Zertifikate zu zählen, siehe z.B. [Wel93].

□

Wie der Beweis zeigt, ist das analoge Entscheidungsproblem k-SAT für alle $k \geq 3$ NP-vollständig. Wie steht es mit $k = 2$? Tatsächlich ist Satz 1.4.7 hinsichtlich k bestmöglich:

Satz 1.4.8 [DP60, EIS76] *Das Problem 2-SAT ist in polynomieller Zeit entscheidbar.*

Beweis. Sei $F = \bigwedge_{j=1}^{\ell} C_j$ eine Boolesche Formel in CNF über den Booleschen Variablen $x_1, \ldots, x_n$ mit höchstens zwei Literalen pro Klausel C_j, $1 \leq j \leq \ell$. O.B.d.A. enthalte F pro Klausel genau zwei Literale (andernfalls verdopple den Literal in jeder Klausel mit nur einem Literal). Konstruiere zu F einen gerichteten Graphen $D(F) = (V, A)$, $A \subseteq V \times V$, wobei $V = \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$ die Menge der Literale ist und A für jede Klausel $C_j = z_j^{(1)} \vee z_j^{(2)}$ in F die gerichteten Kanten $(\overline{z_j^{(1)}}, z_j^{(2)})$ und $(\overline{z_j^{(2)}}, z_j^{(1)})$ enthält. Eine Kante $(z_1, z_2) \in A$ zwischen zwei beliebigen Literalen z_1 und z_2 in $D(F)$ hat dann die Bedeutung, daß für jede erfüllende Belegung der Booleschen Variablen gilt: ist der Literal z_1 wahr, dann auch z_2. Die *starken Zusammenhangskomponenten* eines gerichteten Graphen D sind die Äquivalenzklassen der Relation $u \sim v \Leftrightarrow$ „D enthält einen gerichteten $u-v$–Pfad und einen gerichteten $v-u$–Pfad". Es folgt, daß für jede erfüllende Belegung von F alle Literale, die zu Knoten einer starken Zusammenhangskomponente von D korrespondieren, denselben Wahrheitswert haben. Da sich die starken Zusammenhangskomponenten eines gerichteten Graphen in Zeit $\mathcal{O}(nm)$ berechnen lassen (Übung)[22], zeigt das folgende Lemma, daß 2-SAT in polynomieller Zeit entschieden werden kann. □

Lemma 1.4.9 *Eine 2-SAT-Formel F besitzt eine erfüllende Belegung genau dann, wenn für alle Booleschen Variablen x_i, $i \in \{1, \ldots, n\}$, die Literale x_i und $\overline{x_i}$ in verschiedenen starken Zusammenhangskomponenten von $D(F)$ liegen.*

Beweis. „⇒": Nach obigem klar.
„⇐": Wir konstruieren wie folgt eine erfüllende Belegung der Booleschen Formel F.

```
FOR i := 1 TO n DO IF (x_i noch nicht gesetzt) DO BEGIN
  a := { x_i    falls es in D einen gerichteten x̄_i − x_i−Pfad gibt,
       { x̄_i    sonst;
  L := a + {b : es gibt einen gerichteten a − b−Pfad in D};
  setze alle Literale in L auf wahr;
  setze die Komplemente aller Literale in L auf falsch;
END;
```

Es ist zu überprüfen, daß die Setzung der Wahrheitswerte konsistent ist. Nach Voraussetzung an D gibt es für kein i zugleich einen gerichteten $x_i - \overline{x_i}-$ und einen gerichteten $\overline{x_i} - x_i$–Pfad. Wenn es jedoch einen (und damit genau einen) der beiden Pfade gibt, so wurde a so gesetzt, daß dieser Pfad der gerichtete $\overline{a} - a$–Pfad ist. Es folgt $\overline{a} \notin L$. Angenommen nun, L enthielte einen Literal b und sein Komplement $\overline{b}$, d.h. es gäbe in D einen gerichteten $a - b-$ und einen gerichteten $a - \overline{b}$–Pfad. Aufgrund der folgenden Symmetrie-Eigenschaft von D:

D enthält einen gerichteten $z_1 - z_2$–Pfad genau dann,
wenn D einen gerichteten $\overline{z_2} - \overline{z_1}$–Pfad enthält.

[22] tatsächlich sogar in linearer Zeit – eine Anwendung von Tiefensuche, siehe [Tar72, CLR90]

gäbe es dann aber auch einen $\overline{b}-\overline{a}$–Pfad und daher einen $a-\overline{a}$–Pfad, Widerspruch. Weiterhin enthält L keine in einem früheren Schritt auf *falsch* gesetzten Literale, denn angenommen, es gäbe einen gerichteten $a-b$–Pfad und b wäre in einem früheren Schritt auf *falsch* gesetzt worden. Dann enthielte D wieder aufgrund der Symmetrie-Eigenschaft auch einen gerichteten $\overline{b}-\overline{a}$–Pfad, was heißt, daß auch $\overline{a}$ schon in diesem früheren Schritt (nämlich auf *wahr*) gesetzt worden wäre, Widerspruch. □

Wie der Beweis zeigt, kann ggf. eine erfüllende Belegung einer 2-SAT-Formel in Zeit $\mathcal{O}(nm)$ konstruiert werden. Betrachten wir nun graphentheoretische Entscheidungsprobleme.

Satz 1.4.10 (KARP 1972) *Das folgende Entscheidungsproblem ist* NP*-vollständig:*

INDEPENDENT SET*:*

INSTANZ: ein Graph $G=(V,E)$ und ein $k\in\mathbb{N}$,
FRAGE: $\alpha(G)\geq k$*?*

Beweis. Wieder ist offensichtlich INDEPENDENT SET $\in$ NP. Wir transformieren SAT auf INDEPENDENT SET. Sei $F=\bigwedge_{j=1}^{\ell} C_j$ eine Instanz von SAT, d.h. eine Boolesche Formel in konjunktiver Normalform über den Booleschen Variablen $x_1,\ldots,x_n$. Zu F konstruiere den Graphen $G=G(F)=(V,E)$ mit

$$\begin{aligned} V &= \{(a,j):\ a \text{ ist Literal in der Klausel } C_j,\ 1\leq j\leq \ell\},\\ E &= \{\{(a_1,j_1),(a_2,j_2)\}:\ a_1=\overline{a_2}\ \vee\ j_1=j_2\}. \end{aligned}$$

G enthält also für jedes Auftreten eines Literals a einen Knoten, wobei die Literale einer Klausel eine Clique bilden und Literale aus verschiedenen Klauseln genau dann miteinander verbunden sind, wenn der eine die Negation des andern ist. Nun gilt:

$$\alpha(G)\geq \ell \ \Leftrightarrow\ F \textit{ erfüllbar.}$$

„$\Rightarrow$“: Da eine stabile Menge aus jeder Clique nur höchstens einen Knoten enthalten kann, gilt $\alpha(G)\leq\ell$. Eine ℓ-Clique in G enthält also aus jeder Klausel C_j, $1\leq j\leq\ell$, genau einen Literal. Je nach dem, ob der zugehörige Literal eine Boolesche Variable oder deren Verneinung ist, weise dieser Booleschen Variablen den Wert *wahr* oder *falsch* zu. Nach Konstruktion von G ist diese Wertzuweisung möglich, d.h. wohldefiniert. Den restlichen Booleschen Variablen weise beliebig einen Booleschen Wert zu. Die erhaltene Belegung der Variablen erfüllt dann offenbar jede Klausel und damit die Boolesche Formel. Also ist F erfüllbar.
„$\Leftarrow$“: Wenn umgekehrt eine Boolesche Formel F erfüllbar ist, dann gibt es eine F erfüllenden Belegung der Booleschen Variablen, so daß in jeder Klausel mindestens ein Literal *wahr* ist. Wähle aus jeder Klausel genau einen solchen Literal mit Wahrheitswert *wahr* aus und bilde die zugehörige Knotenmenge $S\subset V$. Dann ist S ist eine stabile Menge der Größe ℓ, denn die zugrundeliegenden Literale haben alle den Wahrheitswert *wahr*, so daß keiner das Komplement eines anderen sein kann.
Wieder ist die Reduktion offensichtlich polynomiell. □

Definition 1.4.11 *Eine Knotenmenge C in einem Graphen $G=(V,E)$ heißt* Knotenüberdeckung, *wenn jede Kante in G mit mindestens einem Knoten aus C inzidiert.*

Wenn man die Kanten von G als Gänge oder Räume eines Gebäudekomplexes interpretiert und jeden Knoten als Beobachtungspunkt, von dem aus man die inzidierenden Kanten überblicken kann, so sucht man beim Problem NODE COVER nach einer kleinsten Menge von Beobachtungspunkten:

NODE COVER:

INSTANZ: ein Graph $G = (V, E)$ und ein $k \in \mathbb{N}$,
FRAGE: besitzt G eine Knotenüberdeckung $C \subset V$ der Größe $|C| \leq k$?

Satz 1.4.12 (KARP 1972) CLIQUE *und* NODE COVER *sind* NP*-vollständig.*

Beweis. INDEPENDENT SET transformiert sich auf diese Entscheidungsprobleme:
Eine Knotenmenge S in einem Graphen $G = (V, E)$ ist stabil genau dann, wenn sie eine Clique in $\overline{G}$ ist.
Eine Knotenmenge S in einem Graphen $G = (V, E)$ ist stabil genau dann, wenn $V \setminus S$ eine Knotenüberdeckung ist. □

Man beachte, daß jedes dieser Probleme für festes $k \in \mathbb{N}$, d.h. wenn k nicht Teil der Eingabe ist, durch Enumeration aller k-Teilmengen von V in Zeit $\mathcal{O}(n^k m)$, also polynomiell, lösbar ist.

INDEPENDENT SET und NODE COVER sind selbst auf Graphen mit Maximalgrad $\Delta(G) \leq 3$ NP-vollständig:

Satz 1.4.13 [GJS76, GJ77] INDEPENDENT SET *($\Delta \leq 3$) ist* NP*-vollständig.*

Beweis. Wir reduzieren von INDEPENDENT SET. Sei also G ein Graph auf den Knoten $v_1, \ldots, v_n$ und sei $v \in V$ ein Knoten in G vom Grad $d := d(v) \geq 4$ mit den Nachbarn $\Gamma(v) = \{u_1, \ldots, u_d\}$. Wir ersetzen lokal v wie folgt durch einen Subgraphen vom Maximalgrad 3: entferne v aus G und füge stattdessen einen Kreis $C_{2d} = (c_1, c_2, \ldots, c_{2d})$ der Länge $2d$ hinzu, wobei jeder der vorherigen Nachbarn u_j, $1 \leq j \leq d$, von v nun mit dem Knoten c_{2j} durch eine Kante verbunden wird. Ferner hänge an den Knoten c_1 ein Blatt b an. Der so aus G entstandene Graph heiße G'. Dann gilt für jedes $k \in \mathbb{N}$:

$$\alpha(G) \geq k \quad \Leftrightarrow \quad \alpha(G') \geq k + d,$$

„$\Rightarrow$“: Sei S eine stabile Menge in G. Dann ist

$$S' := \begin{cases} S - v \; + \{b, c_2, c_4, \ldots, c_{2d}\} & \text{falls } v \in S, \\ S \quad\quad + \{c_1, c_3, \ldots, c_{2d-1}\} & \text{sonst;} \end{cases}$$

eine stabile Menge in G' der Größe $|S| + d$.
„$\Leftarrow$“: Sei S' ein stabile Menge in G' der Größe mindestens $k + d$. Dann ist $|S' \cap \{b, c_1, c_2, \ldots, c_{2d}\}| \leq d + 1$ mit Gleichheit genau dann, wenn $S' \cap \{b, c_1, c_2, \ldots, c_{2d}\} = \{b, c_2, c_4, \ldots, c_{2d}\}$. Also ist

$$S := \begin{cases} S' + v \; - \{b, c_1, c_2, \ldots, c_{2d}\} & \text{falls } |S' \cap \{b, c_1, c_2, \ldots, c_{2d}\}| = d + 1, \\ S' \quad\quad - \{c_1, c_2, \ldots, c_{2d}\} & \text{sonst} \end{cases}$$

eine stabile Menge in G der Größe mindestens k.
Wenn man nun alle Knoten vom Grad größer als 3 in G auf diese Weise ersetzt, erhält

man einen Graphen G^* vom Maximalgrad höchstens 3, für den gilt:

$$\alpha(G) \geq k \quad \Leftrightarrow \quad \alpha(G^*) \geq k + \sum_{d(v)>3} d(v).$$

□

Satz 1.4.13 ist bestmöglich hinsichtlich des Maximalgrads:

Übung 1.4.14 *In Graphen vom Maximalgrad höchstens zwei gibt es einen linearen Algorithmus für* INDEPENDENT SET.

co-NP. Die Definitin von NP ist asymmetrisch in *JA* und *NEIN*: nur wenn die Antwort auf die Instanz des Entscheidungsproblems *JA* lautet, muß ein Zertifikat hierfür existieren. Im Falle einer *NEIN*-Instanz wird nicht gefordert, daß es ein Zertifikat dafür gibt, daß kein *JA*-Zertifikat existiert. Wie sollte beispielsweise ein Zertifikat dafür aussehen, daß ein Graph keine Clique der Größe $\geq k$ besitzt oder eine Boolesche Formel nicht erfüllbar ist? Das Problem Π^c, das aus Π durch Vertauschen von Y_Π und N_Π entsteht, heißt das Komplement von Π.[23] Die Menge aller Entscheidungsprobleme, die Komplemente von NP-Problemen sind, wird mit co-NP bezeichnet. Tatsächlich weiß man von keinem der bekannten NP-vollständigen Probleme, ob auch sein Komplement in NP liegt – es gilt jedoch:

Proposition 1.4.15 NP *ist bezüglich Komplementbildung abgeschlossen (d.h. es gilt* NP = co-NP*) genau dann, wenn es ein* NP*-vollständiges Problem* Π *gibt, dessen Komplement* Π^c *ebenfalls in* NP *liegt.* □

Man vermutet daher, daß NP $\neq$ co-NP. Diese Vermutung ist allerdings noch stärker als die Vermutung P $\neq$ NP, denn wegen der Abgeschlossenheit von P unter Komplementbildung (P = co-P) gilt NP $\neq$ co-NP $\Rightarrow$ P $\neq$ NP. Probleme in NP $\cap$ co-NP heißen nach EDMONDS [Edm65] *wohlcharakterisiert*. Beispielsweise ist Bipartitheit wohlcharakterisiert: ist ein Graph bipartit, so läßt sich das durch eine Bipartition seiner Knotenmenge belegen, ist er nicht bipartit, so stellt jeder ungerade Kreis ein Zertifikat dafür dar. Viele bekannte Maximum-Minimum-Sätze der Graphentheorie beweisen insbesondere, daß ein Problem wohlcharakterisiert ist. Proposition 1.4.15 besagt in diesem Zusammenhang, daß ein wohlcharakterisiertes Problem nicht NP-vollständig sein kann, außer es ist NP = co-NP. Wegen co-P = P und P $\subseteq$ NP enthält NP $\cap$ co-NP ganz P.

Nur für wenige Probleme in NP $\cap$ co-NP hat man bisher noch *keinen* polynomiellen Algorithmus gefunden. Ein berühmtes Beispiel hierfür ist das Problem PRIMES: gegeben eine natürliche Zahl n, ist n prim? (Man beachte, daß das Sieb des ERATOSTHENES in der Codierungslänge $\langle n \rangle = \Theta(\log n)$ exponentielle Laufzeit hat.) Während das Zertifikat für die Zusammengesetztheit einer Zahl auf der Hand liegt, beruht der nichtdeterministische Algorithmus (das polynomiell prüfbare Zertifikat) für PRIMES auf dem kleinen Satz von FERMAT [Pra75], vergleiche auch [BDG88, Pap94].

Übungen und Anmerkungen

Wenn eine Boolesche Formel nicht erfüllbar ist, so kann man sich fragen, wieviele ihrer Klauseln von einer Wahrheitswertebelegung maximal gleichzeitig erfüllt werden können. Dies führt auf das Problem MAX k-SAT:

[23] Man beachte, daß die zu Π^c gehörige Sprache also i.a. nicht das Komplement bezüglich Σ^* der zu Π gehörenden Sprache ist, da es i.a. Worte in Σ^* gibt, die überhaupt keine Elemente aus D_Π codieren. Hier geht man davon aus, daß sich solche Worte durch eine einfache syntaktische Überprüfung herausfiltern lassen.

INSTANZ: eine k-SAT-Formel F und ein $\ell \in \mathbb{N}$,
FRAGE: gibt es eine Belegung der Booleschen Variablen von F,
die mindestens ℓ Klauseln erfüllt?

Während dieses Problem für $k \geq 3$ trivialerweise NP-vollständig ist, gilt nun auch schon für $k = 2$:

Übung 1.4.16 [GJS76] MAX 2-SAT *ist* NP*-vollständig.*
[*Hinweis:* Reduktion von 3-SAT. Für jede 3-SAT-Klausel $C = a \vee b \vee c$ betrachte den folgenden Satz S von zehn 2-SAT-Klauseln.]

$$\begin{array}{cccc} (a) & (b) & (c) & (w) \\ (\overline{a} \vee \overline{b}) & (\overline{b} \vee \overline{c}) & (\overline{c} \vee \overline{a}) & \\ (a \vee \overline{w}) & (b \vee \overline{w}) & (c \vee \overline{w}). & \end{array}$$

Übung 1.4.17 [Tov84] **a)** *Für* SAT*, eingeschränkt auf Boolesche Formeln, in denen jede Variable nur höchstens zweimal (positiv oder negiert) auftritt, gibt es einen linearen Algorithmus.* [*Hinweis:* eliminiere in jedem Schritt eine Variable]
b) SAT*, eingeschränkt auf Boolesche Formeln, in denen jede Variable nur höchstens dreimal (positiv oder negiert) auftritt, ist* NP*-vollständig.* [*Hinweis:* ersetze jede Variable, die ℓ-mal, $\ell > 3$, auftritt, durch einen Satz von ℓ neuen Variablen]

Übung 1.4.18 a)[ADLRY94] *Das Problem* HALF SIZE CLIQUE *„gegeben ein Graph G, enthält G eine $\lceil n/2 \rceil$-Clique?“ ist* NP*-vollständig.*
b)[FK95] *Wenn es einen polynomiellen Algorithmus gibt, um zu entscheiden, ob ein Graph G eine Clique der Größe $\log n$ enthält, dann gibt es einen $\mathcal{O}(n^{\mathcal{O}(\sqrt{n})})$-Algorithmus für* CLIQUE. [*Hinweis:* konstruiere einen Graphen auf $\binom{V}{\sqrt{k}}$]
c) *Sei $a \in \mathbb{N}$ fest. Dann liegt das Problem* INDEPENDENT SET *eingeschränkt auf die Menge aller Graphen G vom Minimalgrad $\delta(G) \geq n - a$ in* P*.*

Während zudem für die Probleme „enthält G eine Clique (bzw. eine stabile Menge) der Größe mindestens k?“ die Existenz eines $\mathcal{O}(n^c)$-Algorithmus für eine Konstante $c \in \mathbb{N}$ unabhängig von k unwahrscheinlich zu sein scheint [DF95a, DF95b, FK95], gilt für NODE COVER:

Übung 1.4.19 a)[BFM89] *Das Problem* NODE COVER *besitzt für festen Parameter k einen $\mathcal{O}(2^k n)$-Algorithmus.*
b)[GJ77] *Das Problem* CONNECTED NODE COVER *„gegeben ein Graph G und ein $k \in \mathbb{N}$, besitzt G eine Knotenüberdeckung $C \subset V$ der Größe $|C| \leq k$, die einen zusammenhängenden Subgraphen induziert (d.h. $G[C]$ ist zusammenhängend)?“ ist* NP*-vollständig.*
c) NODE COVER *wird in polynomieller Zeit entscheidbar, wenn man nach Knotenüberdeckungen fragt, die zudem eine stabile Knotenmenge darstellen.*

Übung 1.4.20 (GALLAI 1959) *Bezeichne $\tau(G)$ die Größe einer minimum Knotenüberdeckung eines Graphen G. Dann gilt $\tau(G) + \alpha(G) = n$.*

Dominanz. So wie bei dem Problem NODE COVER die Kanten eines Graphen $G = (V, E)$ durch möglichst wenige Knoten zu überdecken sind, so sind bei dem folgenden Problem quasi die Knoten durch Knoten zu überdecken – wobei wir sagen wollen, ein Knoten überdeckt einen anderen, wenn er zu ihm benachbart ist. Eine Knotenmenge $D \subseteq V$ heißt *dominierend* in einem Graphen G, falls jedes $v \in V \setminus D$ zu mindestens einem Knoten aus D adjazent ist. Für einen Knoten $v \in V$ nennen wir die Menge $\Gamma(v) + v$ die *abgeschlossene Nachbarschaft* von v. Es gilt:

Übung 1.4.21 *Für einen Graphen G stimmen die folgenden Größen alle überein:*

$$\begin{array}{rcl} \sigma(G) & = & \min\{|D| : D \subseteq V \text{ dominierend in } G\}; \\ \sigma'(G) & = & \min\{|T| : T \cap (\Gamma(v) + v) \neq \emptyset \text{ für alle } v \in V\}; \\ \sigma''(G) & = & \min\{c(W) : W \text{ ist spannender Wald in } G \text{ vom Durchmesser} \leq 2\}. \end{array}$$

$\sigma(G) = \sigma'(G) = \sigma''(G)$ heißt die *Dominanzzahl* von G, das zugehörige Entscheidungsproblem DOMINATING SET:

INSTANZ: ein Graph $G = (V, E)$ und ein $k \in \mathbb{N}$,
FRAGE: besitzt G eine dominierende Menge D der Größe $\leq k$?

Übung 1.4.22 DOMINATING SET ($\Delta(G) \leq 6$) *ist* NP*-vollständig* [LR79]. [*Hinweis:* formuliere NODE COVER durch eine lokale Transformation als ein Knoten-Knoten-Überdeckungsproblem.]

Übung 1.4.23 a) *Für einen Graphen G ohne isolierte Knoten gilt $\sigma(G) \leq \min\{\tau(G), \alpha(G)\} \leq n/2$. Für disjunkte Vereinigungen von K_2s gilt offenbar Gleichheit in der Ungleichungskette. Man gebe jeweils eine Graphenfamilie an, für die die eine bzw. die andere Ungleichung beliebig schlecht wird.*
b) *Sei $G = (V, E)$ ein Graph ohne isolierte Knoten. Wenn $D \subseteq V$ eine inklusionsminimale dominierende Menge ist, so ist auch $V \setminus D$ dominierend.*
c) *Sei G ein zusammenhängender Graph mit $\sigma(G) = \tau(G)$. Dann gilt $\delta(G) \leq 2$* [Vol96].

1.4.2 NP-schwere Probleme und Optimierung

In den Anwendungen des SATISFIABILITY Problem genügt es in aller Regel nicht, bloß zu entscheiden, ob eine gegebene Boolesche Formel F erfüllbar ist; man möchte in dem Fall, daß F erfüllbar ist, auch eine F erfüllende Belegung der Variablen angeben. Ebenso möchte man beim Entscheidungsproblem CLIQUE zur Instanz (G, k) nicht nur wissen, ob $\omega(G) \geq k$ gilt, sondern man möchte ggf. eine Clique $C \subseteq V$ der Größe $|C| \geq k$ auch finden. Dieses Problem bezeichnet man als *Suchproblem*.

Des weiteren ist oftmals eine Zielfunktion c gegeben, bezüglich derer man optimale Lösungen für eine Instanz I sucht. Dies führt zum Begriff des *kombinatorischen Optimierungsproblems*. Bei einem kombinatorischen Optimierungsproblem Π sind drei Dinge gegeben:

(i) Eine Menge $D_\Pi \subseteq \Sigma^*$ von Instanzen,

(ii) für alle Instanzen $I \in D_\Pi$ eine Menge $Sol(I)$ zulässiger Lösungen von I und

(iii) eine Zielfunktion $c: Sol(I) \to \mathbb{Q}$ über den Lösungen.
Die Zahl $c(\sigma)$ heißt auch der Wert der Lösung $\sigma \in Sol(I)$.

Beim Optimierungsproblem CLIQUE z.B. sind die Instanzen Graphen, die Menge $Sol(G)$ der Lösungen zu einer Instanz G ist die Menge der Cliquen in G, und die Zielfunktion c ist einfach die Größe einer Clique C in G, d.h. es ist $c(C) := |C|$ für alle $C \in Sol(G)$. Gesucht ist nun der *Optimalwert*

$$OPT(I) := \begin{cases} \min\{c(I,\sigma) : \sigma \in S(I)\}, & \text{falls } \Pi \text{ ein Minimierungsproblem,} \\ \max\{c(I,\sigma) : \sigma \in S(I)\}, & \text{falls } \Pi \text{ ein Maximierungsproblem,} \end{cases}$$

bzw. eine *Optimallösung* der Instanz $I \in D_\Pi$, d.h. ein $\sigma^* \in Sol(I)$ mit $c(\sigma^*) = OPT(I)$. Beim Problem CLIQUE sind dies also der Wert $\omega(G)$ bzw. eine maximum Clique in G.

Die künstlich anmutende Entscheidungsproblem-Version eines Optimierungsproblems wie z.B. bei CLIQUE, bei der man zur Eingabe einen Parameter $B \in \mathbb{Q}$ als Schranke

hinzufügt und fragt, ob es eine Lösung $\sigma \in Sol(I)$ mit $c(\sigma) \geq B$ (bzw. $\leq B$ bei Minimierungsproblemen) gibt, dient lediglich dem Zweck, das Optimalwert- und Optimallösungsproblem komplexitätstheoretisch einordnen und klassifizieren zu können – man beachte, daß NP als eine Klasse von Entscheidungsproblemen definiert ist.

Offenbar ist ein Entscheidungsproblem auf das entsprechende Suchproblem reduzierbar. Das Suchproblem ist also mindestens so schwer wie das Entscheidungsproblem, und es scheint sogar schwerer zu sein, denn die Antwort ist nicht mehr ein simples *JA* oder *NEIN*: wenn die Antwort *JA* lautet, ist nun sogar ein Beweis (ein Zertifikat) dafür gesucht. Ebenso ist die Entscheidungsproblem-Version eines jeden Optimierungsproblems auf das Optimalwertproblem und dieses wiederum auf das Optimallösungsproblem polynomiell reduzierbar. Diese Probleme sind daher mindestens so schwer wie das zugehörige Entscheidungsproblem, und die Schwierigkeit der Probleme scheint sogar in dieser Reihenfolge zu wachsen.

Um nun die Komplexität von Optimierungs- und Suchproblemen mit der von Entscheidungsproblemen in Beziehung setzen zu können, empfiehlt es sich, den Reduzierbarkeitsbegriff zu verallgemeinern. Bei der many-one-Reduzierbarkeit wurde jede Instanz des einen Problems auf jeweils genau eine Instanz des anderen zurückgeführt. Bei der polynomielle Turing Reduzibilität im Sinne von [Coo71] hingegen erachtet man ein Problem als effizient auf ein anderes reduziert, wenn es einen polynomiellen Algorithmus gibt, der eine Instanz des einen Problems unter Benutzung beliebig (aber natürlich polynomiell) vieler Lösungen von Instanzen des anderen Problems löst. Dies führt zur

Definition 1.4.24 *Ein* Orakel-Algorithmus *(OTM) mit Orakel für ein Problem* Π *ist ein Algorithmus, der eine hypothetische Subroutine aufrufen kann, die ihm zu einer Instanz des Problems* Π *in einem Zeitschritt eine Lösung liefert.*

Definition 1.4.25 *Ein Problem (eine Sprache)* Π *heißt* polynomiell Turing-reduzierbar *auf ein Problem* Π'*, in Zeichen* $\Pi \leq_T \Pi'$*, falls es einen Orakel-Algorithmus mit Orakel für* Π' *gibt, der* Π *in (deterministischer) polynomieller Zeit berechnet.*

Jede $\leq_m$-Reduktion ist also auch eine Turing-Reduktion. Oberhalb von NP können Turing-Reduktionen leistungsfähiger sein als polynomielle Transformationen [LLS75], vgl. auch [WT92]. Innerhalb von NP können sich die beiden Reduktionsarten nur unterscheiden, falls $\mathrm{P} \neq \mathrm{NP}$.

Definition 1.4.26 *Ein Problem* Π *heißt* NP-schwer*, in Zeichen* $\mathrm{NP} \leq_T \Pi$ *oder* $\mathrm{NP} \subseteq \mathrm{P}^{\Pi}$*, falls* $\Pi' \leq_T \Pi$ *für alle* $\Pi' \in \mathrm{NP}$*.*

Die Klassifikation eines Problems Π als NP-schwer stellt eine strukturelle untere Schranke für die Komplexität von Π dar: Π ist nicht polynomiell lösbar, außer es ist P=NP. Nach Definition sind insbesondere alle NP-vollständigen Probleme NP-schwer. Aus obiger Diskussion ergibt sich weiter:

Lemma 1.4.27 *Sei* Π *ein* NP*-vollständiges Entscheidungsproblem. Dann ist das mit* Π *verknüpfte Suchproblem NP-schwer. Sei* Π' *ein kombinatorisches Optimierungsproblem. Wenn sein zugehöriges Entscheidungsproblem* NP*-vollständig ist, so sind die Probleme, den Optimalwert und eine Optimallösung von* Π' *zu bestimmen, NP-schwer.*

Beispielsweise ist also nach den Sätzen 1.4.10 bzw. 1.4.12 die Bestimmung der Graphenparameter $\alpha(G), \omega(G)$ oder $\tau(G)$ NP-schwer und ebenso die Berechnung einer maximum stabilen Menge, einer maximum Clique oder einer minimum Knotenüberdeckung in einem Graphen.

NP-Äquivalenz. Tatsächlich sind diese drei Optimierungsrobleme aber auch nicht schwerer als ihre Entscheidungsproblem-Varianten in dem Sinne, daß, wenn das zugehörige Entscheidungsproblem in polynomieller Zeit lösbar ist, so auch das Optimalwert- und Optimallösungsproblem (derartige Probleme nennt man auch NP-*leicht*). Wir überlegen uns dies am Beispiel CLIQUE. Zunächst ist das Optimalwertproblem polynomiell Turing-reduzierbar auf das Entscheidungsproblem: nach spätestens n (bzw. $\log n$ bei binärer Suche) vielen Aufrufen eines Algorithmus (eines Orakels) für das Entscheidungsproblem CLIQUE mit verschiedenen Werten für k kennt man auch $\omega(G)$. Das Optimallösungsproblem ist wiederum polynomiell Turing-reduzierbar auf das Optimalwertproblem: Man entfernt solange Knoten $v \in V$ mit $\omega(G - v) = \omega(G)$ aus dem Graphen, bis der verbleibende Graph H keinen solchen Knoten mehr enthält. Dann ist $V(H)$ eine Clique in G mit $|V(H)| = \omega(G)$. D.h. die Parameter $\omega(G), \alpha(G)$ oder $\tau(G)$ sowie eine maximum Clique, eine maximum stabile Menge oder eine minimum Knotenüberdeckung sind für jeden Graphen in polynomieller Zeit berechenbar genau dann, wenn P = NP gilt. Man sagt, diese Optimierungsprobleme sind NP-*äquivalent*.

Es scheint jedoch unwahrscheinlich, daß sich alle NP-Suchprobleme polynomiell auf ihre Entscheidungsprobleme Turing-reduzieren lassen:

Satz 1.4.28 (BORODIN, DEMERS 1976) *Falls* P $\neq$ NP$\cap$co-NP, *so gibt es eine p-Relation* R *mit* $L_R \in$ NP\P, *für die das Suchproblem nicht Turing-reduzierbar auf das Entscheidungsproblem ist.* □

Siehe auch [BG94]. Es gilt jedoch immerhin:

Satz 1.4.29 (PAZ, MORAN 1981) *Sei* L *eine* NP*-vollständige Sprache und* R *eine p-Relation für* L *(d.h. es gilt* $L = L_R$*). Dann gilt: das Suchproblem für* R *ist Turing-reduzierbar auf das Entscheidungsproblem für* R.

Beweis. Offensichtlich genügt es, das Suchproblem auf das Optimalwertproblem zu reduzieren. Sei $c : R(x) \to \mathbb{Q}$ die (o.B.d.A. zu maximierende) Zielfunktion auf der Menge $R(x)$ der Zertifikate (der Lösungen) für x, und sei $p(n)$ das Polynom, durch das die Codierungslänge der Zertifikate $z \in R(x)$ zu einer Instanz $x \in L$ der Länge $|x| = n$ beschränkt ist. O.B.d.A. haben alle Zertifikate $z \in R(x)$ Codierungslänge genau $p(|x|)$. Definiere ein neues Optimalwertproblem für die Relation R durch die injektive Zielfunktion

$$c'(z) \;=\; 2^{p(|x|)+1} c(z) \;+\; Pos(z),$$

wobei $Pos(z)$ die Position von z in der lexikographischen Ordnung aller 0-1-Zeichenketten der Länge $p(n)$ angibt. Beachte, daß es unter der Zielfunktion c' für alle $x \in L$ eine eindeutige Optimallösung $z^* \in R(x)$ gibt, die zudem auch eine Optimallösung bezüglich der Zielfunktion c ist. Es reduziert sich also das Suchproblem für $\langle R, c\rangle$ auf das für $\langle R, c'\rangle$ und dieses wiederum auf das Optimalwertproblem für $\langle R, c'\rangle$. Weil L NP-vollständig ist, reduziert sich aber das Optimalwertproblem für $\langle R, c'\rangle$ auf das Optimalwertproblem für $\langle R, c\rangle$. □

NP-schwere Probleme können selbst in dem Fall unzugänglich sein, daß sich P=NP herausstellen sollte. Sie bilden keine Äquivalenzklasse wie die NP-vollständigen Probleme. Es ist ein offenes Problem, ob alle NP-schweren Probleme in NP auch schon NP-vollständig sind. LADNER [Lad75] konnte immerhin zeigen, daß es, falls P $\neq$ NP, Entscheidungsprobleme gibt, die weder in P noch NP-vollständig sind. Für eine genauere Klassifizierung der Komplexität von Optimalwert- und Optimallösungsproblemen siehe [Kre88, CT91].

Übung 1.4.30 *Man zeige direkt, daß das Suchproblem für* SAT NP-*äquivalent ist.*

1.5 Zum Umgang mit NP-schweren Problemen: Algorithmen für INDEPENDENT SET

Unter der Hypothese P $\neq$ NP macht es keinen Sinn, nach effizienten Algorithmen für NP-schwere Optimierungsprobleme zu suchen, die in allen Fällen eine optimale Lösung finden. Trotzdem müssen in der Praxis auch solche hartnäckige kombinatorische Optimierungsprobleme „gelöst" werden. Wir stellen einige wichtige Herangehensweisen dazu vor, illustiert jeweils am Problem INDEPENDENT SET.

Restriktion. Zunächst ist zu prüfen, ob das vorliegende Problem möglicherweise nur ein Spezialfall eines NP-schweren Problems ist und daher in Wirklichkeit polynomiell lösbar (oftmals sind Forscher Opfer eines unumstößlichen Verallgemeinerungsdranges). So sind z.B. fast alle NP-schweren Probleme auf Bäumen polynomiell lösbar bzw. sogar trivial wie z.B. die Cliquenzahl oder das Hamiltonkreisproblem (die Frage nach der Existenz eines Kreises durch alle Knoten). Etwas interessantere Graphenklassen dieser Art betrachten wir in den Kapiteln 7 und 9. Andererseits werden wir in den Abschnitten 6.3 und 6.5 Probleme kennenlernen, die selbst unter starken Einschränkungen an die Eingaben NP-vollständig bleiben. Eine Übersicht, unter welchen Einschränkungen der Eingaben auf spezielle Graphenklassen wichtige NP-schwere Probleme zugänglich werden, findet man in [Joh85].
Betrachten wir beispielsweise das Problem INDEPENDENT SET auf der Klasse der Bäume.

Proposition 1.5.1 *Das Problem* INDEPENDENT SET *besitzt eingeschränkt auf der Klasse der Bäume selbst in der knotengewichteten Form einen $\mathcal{O}(n+m)$-Algorithmus.*

Beweis. Sei $T=(V,E)$ ein Baum und $c: V \to \mathbb{Q}_+$ eine Gewichtsfunktion auf den Knoten. Gesucht ist also eine stabile Menge $S \subset V$ in G maximalen Gewichts $c(S) = \sum_{v \in S} c(v)$. Ein beliebiger Knoten $w \in V$ sei als Wurzel von T ausgezeichnet. Dann gibt es für jeden Knoten $v \in V$ einen eindeutigen Pfad in T zum Wurzelknoten w. Für jeden Knoten $v \in V$ des Baumes T sei T_v der Teilbaum von T, der von denjenigen Knoten $u \in V$ induziert wird, für die der $u-w-$Pfad den Knoten v enthält. Insbesondere ist also $T_w = T$. Betrachte nun die Funktionen $f^+(v)$ und $f^-(v)$, $v \in V$, die das maximale Gewicht einer stabilen Menge in T_v bezeichnen, die einmal v enthält und das andere Mal nicht. Das maximale Gewicht einer unabhängigen Menge in T ist dann offenbar gegeben durch

$$\alpha(T) \;=\; \max\{f^+(w), f^-(w)\}.$$

Wie kann man $f^+(w)$ und $f^-(w)$ berechnen? Bezeichne

$$\text{Söhne}(v) := \{u \in \Gamma(v) : \; u \in T_v\}$$

die Menge der Söhne von v in T. Die Funktionen f^+ und f^- erfüllen damit für alle Knoten $v \in V$ die folgende Rekursionsformel:

$$\begin{aligned} f^+(v) &= c(v) + \sum_{u \in \text{Söhne}(v)} f^-(u) \\ f^-(v) &= \sum_{u \in \text{Söhne}(v)} \max\{f^+(u), f^-(u)\} \end{aligned}$$

– mit der Interpretation, daß die Werte auf einem Blatt $b \in T$ gerade $c(b)$ bzw. 0 sind. Die Idee ist daher, die Funktionswerte von f^+ und f^- in Dynamischer-Programmierungs-Manier sukzessive von den Blättern her zu berechnen. Hierzu genügt es, zunächst eine Breitensuche in T von w aus durchzuführen, die Knoten von G in der Reihenfolge, wie man ihnen dabei begegnet, anzuordnen und sodann in der umgekehrten Reihenfolge jeweils für jeden Knoten die Funktionswerte von f^+ und f^- zu berechnen. Dann sind an jedem Knoten $v \in V$ alle für die Berechnung von $f^+(v)$ und $f^-(v)$ nach obiger Rekursionsformel benötigten Funktionswerte bereits bekannt. Der Aufwand ist offensichtlich $\mathcal{O}(n+m) = \mathcal{O}(n)$. □

In Kapitel 9 wird dieser Algorithmus auf baumartige Graphen verallgemeinert werden. In Abschnitt 4.2 werden die Parameter $\tau(G)$ und $\alpha(G)$ sowie entsprechende Zertifikate auch in der größeren Klasse der bipartiten Graphen durch Reduktion auf das Matching-Problem in polynomieller Zeit berechnet.

Gutartige exponentielle Algorithmen. Des weiteren ist zu bedenken, daß *für kleine Instanzen* eines NP-schweren Problems durchaus ein (im schlechtesten Fall) exponentieller Algorithmus dienlich sein kann. Insbesondere kann er hier durchaus effizienter sein als ein polynomieller Algorithmus mit großen Konstanten. Durch Ausnutzung der speziellen Struktur des Problems gelingt es zudem oftmals, die Basis und die Konstante (manchmal sogar den Exponenten) der Laufzeitfunktion gegenüber dem trivialen Enumerationsalgorithmus zu verkleinern. Ein Optimierungsparadigma, das ebenfalls in diesen Bereich fällt, das wir hier aber nicht behandeln, ist Branch-and-bound, siehe hierzu z.B. [HS78, PS82, LLRS85, Iba87].
Zur Illustration betrachten wir das Problem INDEPENDENT SET. Durch Enumeration aller Teilmengen der Knotenmenge des Graphen erhält man einen trivialen $\mathcal{O}(m2^n)$-Algorithmus für dieses Graphenproblem. Der folgende, etwas intelligentere Algorithmus inkorporiert zwei Ideen. Zum einen ist es, statt alle $S \in 2^V$ zu generieren und auf Stabilität hin zu überprüfen, sicherlich sparsamer, nur alle maximalen stabilen Knotenmengen in G zu generieren und die größte herauszufischen. Tatsächlich enthält ein Graph nur $\mathcal{O}(3^{n/3}) \approx \Theta(1.44^n)$ viele maximale stabile Knotenmengen [MM65]. Für jeden Knoten v und jede maximale stabile Menge S eines Graphen G gilt nun aber: entweder gilt $v \in S$ und $S - v$ ist eine maximale stabile Knotenmenge in $G - (v + \Gamma(v))$, oder aber $v \notin S$ und S ist eine maximale stabile Knotenmenge von $G - v$. Daraus folgt die rekursive Formel

$$\alpha(G) = \max\{\alpha(G - v),\ 1 + \alpha(G - (v + \Gamma(v)))\}.$$

Offenbar ist es weiterhin sinnvoll, um die Ordnung des Restgraphen $G - (v + \Gamma(v))$ zu minimieren, v als einen Knoten maximalen Grades in G zu wählen. Hat der Restgraph G' schließlich nur noch Maximalgrad höchstens zwei, so läßt sich $\alpha(G')$, wie in Übung 1.4.18 gesehen, in linearer Zeit berechnen. Unser rekursiver Algorithmus zur exakten Bestimmung

von $\alpha(G)$ für einen Graphen G hat damit folgende Gestalt:

```
FUNCTION ALPHA (G) : INTEGER;

BEGIN
  IF Δ(G) ≤ 2 THEN
    ALPHA(G) := Σ_{P Pfad in G} ⌈|P|/2⌉ + Σ_{C Kreis in G} ⌊|C|/2⌋;
  ELSE BEGIN
    wähle einen Knoten v maximalen Grades in G;
    ALPHA(G) := max {ALPHA(G − v), 1+ALPHA(G − (v + Γ(v)))};
  END;
END;
```

Bezeiche $f : \mathbb{N} \to \mathbb{N}$ die Laufzeitfunktion des Algorithmus ALPHA (G) für einen Graph G der Ordnung n. Dann gilt, weil stets $|v + \Gamma(v)| \geq 4$ gilt, für eine Konstante $C \in \mathbb{R}_+$:

$$f(n) \;\leq\; C \cdot n^2 \;+\; f(n-1) \;+\; f(n-4).$$

(Zum Vergleich: für die Funktion $f(n) := 2^n$ gilt $f(n) = 2 \cdot f(n-1)$.) Man kann zeigen, daß für die Lösung dieser Rekurrenzgleichung $f(n) = \mathcal{O}(1.39^n)$ gilt, siehe z.B. [Wil86]. Der Aufwand ist also nur noch etwa die Wurzel der Laufzeit des trivialen Enumerationsalgorithmus (jeweils im schlechtesten Fall).

Proposition 1.5.2 ALPHA *ist ein Algorithmus für* INDEPENDENT SET *mit Laufzeit* $\mathcal{O}(1.39^n)$.

Einen exakten Algorithmus für INDEPENDENT SET mit einer Laufzeit von „nur“ $\mathcal{O}(1.26^n)$ bzw. $\mathcal{O}(1.2108^n)$ findet man bei [TT77] bzw. [Rob86]. Auf planaren Graphen (wo das Problem noch immer NP-vollständig ist) kennt man sogar einen $2^{\mathcal{O}(\sqrt{n})}$-Algorithmus [LT80], vgl. Proposition 8.3.3.

Probabilistische Analyse. Da sich die Klassifizierung eines Problems als NP-schwer auf die worst-case-Komplexität von Algorithmen bezieht, ist es darüberhinaus möglich, daß ein hartnäckiges Problem nur für wenige Instanzen wirklich schwer ist. Man kann also versuchen, Algorithmen zu entwerfen, die immerhin im Mittel (bei einer gegebenen Wahrscheinlichkeitsverteilung über den Instanzen) polynomiell sind („erwartete polynomielle Laufzeit“). Ein solcher Algorithmus ist dann für fast alle Instanzen effizient und hat nur „selten“ exponentielle Laufzeit; er ist probabilistisch in Hinblick auf die Laufzeit. Ein prominentes Beispiel ist der Simplex-Algorithmus der Linearen Optimierung, siehe [Bor87] (allerdings ist dieses Problem auch polynomiell lösbar). So schwierig die Analyse der Komplexität im Mittel für einen bestimmten Algorithmus meist ist, so problematisch ist die Festlegung der in der Anwendung relevanten Verteilung, siehe [Joh84a, Hof87, DF89, Gur91, BCOL92] und auch [OKSW94]. In der Praxis führen oft auch sogenannte „branch and bound“-Techniken [das sind raffinierte Enumerationsmethoden, die im Baum aller Lösungsmöglichkeiten Äste abschneiden, in denen die Optimallösung nicht liegen kann] zu recht schnellen (exakten) Algorithmen, obschon sie im schlechtesten Fall exponentiell sind, siehe z.B. [Iba87, PR88].
Ferner garantieren zuweilen Strukturaussagen über zufällige Graphen (siehe Kapitel 13), daß einfache und für alle Eingaben polynomielle Algorithmen asymptotisch fast sicher eine Lösung finden; der Erfolg solcher Algorithmen ist also probabilistisch in Bezug auf den

Erfolg [Rin87, DF89, Fri89, Fri90]. Wenn zudem Mißerfolge eines solchen Algorithmus selten genug sind, so kann man aus einem solchen Algorithmus sogar einen Algorithmus mit immerhin noch erwartet polynomieller Laufzeit machen, der stets eine Optimallösung berechnet: bei den wenigen Eingaben, bei denen der ursprüngliche Algorithmus keine Lösung finden konnte, enumeriert man vollständig.
Als Beispiel betrachten wir einen einfachen Backtracking-Algorithmus für INDEPENDENT SET. Sei G ein Graph auf der Knotenmenge $\{v_1, \ldots, v_n\}$. Eine stabile Menge S codieren wir durch einen 0-1-Vektor der Länge n mit der Interpretation $S[k] = 1$ genau dann, wenn $v_k \in S$. Die folgende Prozedur erzeugt alle stabilen Mengen in G, die auf den ersten $k-1$ Knoten mit der in S codierten stabilen Menge übereinstimmen.

```
PROCEDURE Erzeuge_stabile_Mengen (k);

BEGIN
  IF k < n THEN
    {erzeuge alle stabilen Erweiterungen von S, v_k ∉ S:}
    Erzeuge_stabile_Mengen (k+1);
  IF (S + v_k stabil) THEN BEGIN
    S[k] := 1;                                          {S := S + v_k}
    IF |S| > alpha THEN    alpha := |S|;
    IF k < n THEN
      {erzeuge alle stabilen Erweiterungen von S, v_k ∈ S:}
      Erzeuge_stabile_Mengen (k+1);
    S[k] := 0;                                          {S := S - v_k}
  END;
END; {Erzeuge_stabile_Mengen}
```

Der Aufruf im Hauptprogramm hat folgende Gestalt:

```
BEGIN {Hauptprogramm}
  alpha := 0;                                   {zukünftiges α(G)}
  FOR k := 1 TO n DO    S[k] := 0;        {aktuelle stabile Menge S := ∅}
  Erzeuge_stabile_Mengen (1);
END; {Hauptprogramm}
```

Wegen der Abfrage „$k < n$“ terminiert jede Rekursion in der Rekursionstiefe $k = n$. Die Menge dieser „Endzustände“ entspricht (via der Funktion S) bijektiv der Menge der stabilen Knotenmengen in G. Jeder Prozeduraufruf erfordert für sich Aufwand $\mathcal{O}(n)$. Im schlechtesten Fall (wie z.B. beim leeren Graphen) erzeugt dieser Algorithmus alle 2^n Knotenmengen und die worst-case-Laufzeit ist $\mathcal{O}(n2^n)$. Für die probabilistische Analyse dieses Algorithmus beachte man, daß jeder Prozeduraufruf *mindestens eine* Folge von Prozeduraufrufen erzeugt, die zu einem Endzustand führt. Bezeichne $I(G)$ die Anzahl der stabilen Knotenmengen in G. Wegen der Rekursionstiefe n gibt es also höchstens $n \cdot I(G)$ Prozeduraufrufe. Wenn wir die durchschnittliche Anzahl unabhängiger Mengen in einem Graphen auf n Knoten mit I_n bezeichnen, hat der Algorithmus ERZEUGE_STABILE_MENGEN also durchschnittlich Laufzeit $\mathcal{O}(n^2 \cdot I_n)$.

Bestimmen wir nun I_n näher. Da es $|\mathcal{G}_n| = 2^{\binom{n}{2}}$ viele verschiedene Graphen auf den

Knoten $\{v_1, \ldots, v_n\}$ gibt, ist

$$I_n \;=\; 2^{-\binom{n}{2}} \sum_{G_n \in \mathcal{G}_n} I(G_n).$$

Die Summe auf der rechten Seite zählt jede Teilmenge von V sooft, wie sie als stabile Menge in den $2^{\binom{n}{2}}$ Graphen G_n auf n Knoten auftritt. Es ist also

$$\sum_{G_n} I(G_n) \;=\; \sum_{S \subseteq V} |\{G_n \in \mathcal{G}_n : \; S \text{ stabil in } G_n\}|.$$

Die Anzahl aller Graphen $G_n \in \mathcal{G}_n$, die eine bestimmte k-elementige Knotenmenge $S \subseteq \{v_1, \ldots, v_n\}$ als stabile Menge enthalten, ist gegeben durch die Anzahl der Möglichkeiten, die $\binom{n}{2} - \binom{k}{2}$ noch möglichen Kanten zu setzen oder nicht zu setzen. Somit wird

$$I_n \;=\; 2^{-\binom{n}{2}} \sum_{k=0}^{n} \binom{n}{k} 2^{\binom{n}{2}-\binom{k}{2}} \;=\; \sum_{k=0}^{n} \binom{n}{k} 2^{-\binom{k}{2}}.$$

Da die Funktion $x \mapsto x[\log n - (x-1)/2]$ an der Stelle $x = \log n + 1/2$ maximal wird, können wir wie folgt abschätzen.

$$\begin{aligned}
n^2 \cdot I_n &= n^2 \cdot \sum_{k=0}^{n} \binom{n}{k} 2^{-\binom{k}{2}} \\
&\le n^2 \cdot \sum_{k=0}^{n} n^k \, 2^{-k(k-1)/2} \\
&= n^2 \cdot \sum_{k=0}^{n} 2^{k\,[\log n - (k-1)/2]} \\
&\le n^3 \, 2^{(\log n + 1/2)\,[\log n - (\log n - 1/2)/2]} \\
&\le n^3 \, 2^{0.5 \log^2 n + 0.5 \log n + 0.125} \\
&= 2^{0.5 \log^2 n + 3.5 \log n + 0.125} \qquad = \mathcal{O}(n^{0.5 \log n + 3.5}).
\end{aligned}$$

Proposition 1.5.3 ERZEUGE_STABILE_MENGEN *ist ein Algorithmus für* INDEPENDENT SET *mit durchschnittlicher Laufzeit* $\mathcal{O}(n^{0.5 \log n + 3.5})$.

Auch die durchschnittliche Laufzeit dieses Algorithmus wächst also superpolynomiell, doch ist sie erheblich kleiner als die worst-case-Laufzeit $\mathcal{O}(2^{n + \log n})$.

Eine naheliegende Idee, den obigen Algorithmus zu verbessern, ist, ihn nur die *maximalen* stabilen Mengen aufzählen zu lassen. Dies ist auch in Zeit $\mathcal{O}(mnK)$ möglich, wobei K die Anzahl der maximalen stabilen Mengen in G bezeichnet [TIAS77] (siehe auch [Law79, JYP88]). Auf zufälligen Graphen (auf bis zu 100 Knoten) ist dieser Ansatz jedoch nicht effizienter, was daran liegt, daß fast alle Graphen schon mindestens $2^{0.5(\log n)^2(1-o(1))}$ viele maximale stabile Mengen (der Größe $\log n$) enthalten; weitaus effizienter ist hier der Algorithmus ALPHA.

Heuristiken und Approximation. Schließlich kann man sich für hartnäckige kombinatorische Optimierungsprobleme unter Abschwächung der Forderung nach einer exakten Lösung i.a. diverse effiziente ad-hoc-Algorithmen, sogenannte *Heuristiken*, ausdenken, die

Lösungen produzieren, die zwar i.a. nicht optimal sind, die aber nach dem intuitiven Gefühl „in der Regel recht gut“ sein sollten. Die Herausforderung heißt hier, die gefundene Lösung zu bewerten. Eine Heuristik, die Lösungen konstruiert, die stets eine gewisse Gütegarantie aufweisen, heißt Approximationsalgorithmus. Wir gehen darauf in Kapitel 10 ein. Für viele Heuristiken (wie z.B. für Methoden der lokalen Verbesserung) kennt man allerdings keine Güteschranken, obwohl sie in der Praxis oft gute Resultate liefern. Als wichtige problemunabhängige Optimierungsparadigmen (sogenannte Metaheuristiken), die globale Optima zu konstruieren versuchen, ohne in schlechten lokalen Optima hängen zu bleiben, sind in diesem Zusammenhang u.a. zu erwähnen:

- *Simulated Annealing* [LA87, AK89, OG89, JAMS89, JAMS91],
- *Genetische Algorithmen* [Gol89, SHF94, Mic96],
- *Tabu Search* [GTW93] und
- *Threshold Accepting* [DS90].

Das Greedy-Prinzip, immer die lokal beste Erweiterung der bisherigen konstruierten Teillösung zu wählen, haben wir in Abschnitt 1.3.3 erfolgreich eingesetzt, um einen minimal aufspannenden Baum in einem Graphen zu finden. Untersuchen wir nun die folgende Greedy-Heuristik für INDEPENDENT SET. Sie entnimmt dem Graphen $G = (V, E)$ in jedem Schritt einen Knoten u, der zu keinem der Knoten der bisher konstruierten stabilen Menge S_{Greedy} benachbart ist und fügt ihn zu S_{Greedy} hinzu. Der Knoten u wird zudem stets so gewählt, daß er möglichst wenige neue Nachbarn von S_{Greedy} produziert. Dadurch erhofft man sich, möglichst viele spätere Kandidaten für u zurückzubehalten und das Verfahren also möglichst lange fortsetzen zu können.

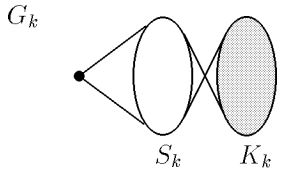
```
FUNCTION GREEDY_MIN_IS : 2^V;
BEGIN
  S_Greedy := ∅;
  U := V;
  WHILE U ≠ ∅ DO BEGIN
    wähle einen Knoten u kleinsten Grades in G[U];
    S_Greedy := S_Greedy + u;
    U := U − (u + Γ(u));
  END;
  GREEDY_MIN_IS:= S_Greedy;
END;
```

Angesichts der Tatsache, daß das Problem INDEPENDENT SET NP-vollständig ist, hat diese Heuristik natürlich den immensen Vorteil, äußerst schnell zu sein. Die Knoten des Graphen werden gerade einmal durchgescannt, zeitkritisch ist lediglich die $|S_{Greedy}|$-malige Bestimmung eines Knotens kleinsten Grades. Eine einfache Implementierung dieses Algorithmus benötigt $\mathcal{O}(n^2)$ Schritte (nämlich wie?). (Wir werden weiter unten eine lineare Implementierung kennenlernen.)

Doch wie gut ist diese Heuristik nun, d.h. was kann man über die Größe der von GREEDY_MIN_IS konstruierten stabile Menge S_{Greedy} aussagen?

Übung 1.5.4 GREEDY_MIN_IS *ist auf vollständigen oder leeren Graphen, Kreisen und Wäldern exakt.*

Die folgende Graphenfamilie $(G_k)_{k\in\mathbb{N}}$ hat jedoch $\alpha(G_k) = \Omega(n(G_k))$, während GREEDY_MIN_IS u.U. nur je eine stabile Menge der Größe 2 findet:



Hierbei bezeichne S_k eine stabile Menge der Kardinalität k, die vollständig zu einem K_k sowie einem weiteren speziellen Knoten verbunden ist. GREEDY_MIN_IS konstruiert also i.a. „beliebig schlechte" stabile Mengen.

Übung 1.5.5 *Man finde eine Folge bipartiter Graphen $(G_k)_{k\in\mathbb{N}}$ mit $\alpha(G_k) = \Omega(n(G_k))$, in denen* GREEDY_MIN_IS *u.U. nur eine stabile Menge der Größe $\mathcal{O}(\sqrt{n})$ findet.*[24]

Es läßt sich jedoch die folgende Aussage machen. GREEDY_MIN_IS produziert eine maximale stabile Menge in G, und für jede maximale stabile Menge S in einem Graphen gilt nach Gleichung (1.3) $|S| \geq n/(\Delta(G)+1) \geq \alpha(G)/(\Delta(G)+1)$. Für die Menge S_{Greedy} läßt sich diese triviale Abschätzung (außer für reguläre Graphen) wie folgt verschärfen.

Proposition 1.5.6 (WEI 1981) *Sei G ein Graph. Die vom Algorithmus* GREEDY_MIN *ausgegebene Menge S_{Greedy} ist eine stabile Knotenmenge in G der Größe*

$$\alpha(G) \geq |S_{Greedy}| \overset{(i)}{\geq} \sum_{v\in V} \frac{1}{d(v)+1} \overset{(ii)}{\geq} \frac{n}{\bar{d}+1} \geq \frac{n}{\Delta(G)+1}, \tag{1.6}$$

wobei $\bar{d} := \frac{1}{n}\sum_{v\in V} d(v) = 2m/n$ den Durchschnittsgrad in G bezeichnet.[25]

Die Abschätzung (i) ist offenbar für alle Graphen, die eine Vereinigung disjunkter Cliquen sind, scharf. Wie jedoch der $K_{r,r}$ zeigt, kann (i) beliebig schlecht werden. Beachte, daß (i) i.a. nicht mehr für jede maximale stabile Menge gilt.

Beweis von Proposition 1.5.6:
Ad (i): Seien $S_{Greedy} = \{u_1, \ldots, u_k\}$ die von GREEDY_MIN in dieser Reihenfolge ausgewählten Knoten. Bezeichne $V_1 := V$, $V_{i+1} = V_i - (u_i + \Gamma(u_i))$ für $i = 1, \ldots, k-1$ sowie $H_i := G[V_i]$ für $i = 1, \ldots, k$. Dann entfernt GREEDY_MIN im i-ten Schritt gerade die Menge $U_i := u_i + \Gamma_{H_i}(u_i)$ aus G. Da die Knotengrade im Restgraphen in jedem Schritt höchstens abnehmen, gilt für jedes $i = 1, \ldots, k$:

$$\sum_{v\in U_i} \frac{1}{d(v)+1} \;\leq\; \sum_{v\in U_i} \frac{1}{d_{H_i}(v)+1} \;\leq\; \sum_{v\in U_i} \frac{1}{d_{H_i}(u_i)+1} \;=\; 1.$$

[24] In Abschnitt 4.2 werden wir sehen, daß es auf bipartiten Graphen einen polynomiellen Algorithmus für INDEPENDENT SET gibt.

[25] Für dreiecksfreie Graphen kann die Schranke (i) vergrößert werden, siehe [AKS80, Gri83a]. Eine völlig andersartige untere Schranke gibt CHUNG [Chu88]:

$$\alpha(G) \cdot \binom{n}{2} \geq \sum_{u\neq v} dist(u,v).$$

Weitere Schranken an $\alpha(G)$ findet man in [Lov82, Ber85, Luz95].

Wegen $V = U_1 \cup \ldots \cup U_k$ und $k = |S_{Greedy}|$ folgt (i).
Ad (ii): Da die Funktion $\frac{1}{1+x}$ konvex ist, verläuft sie oberhalb ihrer Tangente an einer Stelle $\bar{x}$:

$$\frac{1}{1+x} \geq \frac{1}{1+\bar{x}} - \frac{x-\bar{x}}{(1+\bar{x})^2}.$$

Wenn man nun $\bar{x} := \bar{d}$ setzt und für x die Werte $d(v)$, $v \in V$, einsetzt und summiert, findet man (ii). □

Wir wollen am Beispiel des Algorithmus GREEDY_MIN_IS deutlich machen, welchen Einfluß eine geschickte Wahl der Datenstruktur auf die Laufzeit eines Algorithmus hat. Mit nur wenig (programmiertechnischem) Mehraufwand erhält man nämlich:

Proposition 1.5.7 *Die Heuristik* GREEDY_MIN_IS *für* INDEPENDENT SET *kann mit Laufzeit* $\mathcal{O}(n+m)$ *implementiert werden.*

Beweis. Die vielseitige Datenstruktur der sogenannten *Buckets* (engl. Eimer) erlaubt es, einen Knoten von minimalem Grad im Restgraphen $G[U]$ jeweils in konstanter Zeit verfügbar zu machen – bei einem zusätzlichen Aufwand von nur $\mathcal{O}(n+m)$ für Aktualisierungsoperationen. Buckets sind allgemeiner eine Datenstruktur, um Elemente (in hiesigem Beispiel Knoten), denen jeweils eine bestimmte Priorität (hier der Grad im Restgraphen) zugeordnet ist, zu verwalten unter jeweils konstanten Kosten für

- Einfügen eines Elementes,
- Entfernen eines bestimmten Elements und
- Ausgabe eines Elements minimaler Priorität

– vorausgesetzt, die Wertemenge der Prioritätsfunktion ist polynomiell beschränkt.[26] Beachte: Speichert man die Elemente in einem Feld ab, so benötigt die dritte Aufgabe $\Omega(n)$ Schritte, speichert man die Elemente in einer sortierten Warteschlange ab, so benötigen die ersten beiden Aufgaben jeweils Zeit $\Omega(n)$. Die Idee ist daher, beide Darstellungen gewissermaßen zu koppeln. In unserem Fall unterhält man $\Delta(G)+1$ (doppelt verkettete) Listen $Bucket[i]$, $0 \leq i \leq \Delta(G)$, die jeweils alle Knoten in U vom Grad genau i in $G[U]$ enthalten. Eine Variable min vermerkt die erste nicht-leere Liste, d.h. min ist zu Beginn gleich $\delta(G)$. Um Elemente jedoch auch in konstanter Zeit entfernen zu können (um sie ggf. später mit verminderter Priorität woanders wieder einzufügen), unterhalten wir zusätzlich ein Referenzfeld $Position[v]$, $v \in V$, das jeweils auf den zum Element v gehörigen Datensatz in dem Bucket, dem v gerade angehört, verweist.[27]
Dann erfordert die (einmalige) Initialisierung der Listen durch $Bucket[i] := \{v \in V : d(v) = i\}$ für $i = 0, \ldots, \Delta(G)$ zu Beginn des Algorithmus offenbar $\mathcal{O}(n+m)$ Schritte, und, da jeder Knoten höchstens $d(u)$-mal den Bucket wechseln muß, können die Buckets in Zeit $\mathcal{O}(n+m)$ aktualisiert werden. Um eine lineare Laufzeit zu erreichen, kann der Wert min nicht in jedem `WHILE`-Schleifendurchlauf neu berechnet werden – er läßt sich jedoch, wie wir gleich sehen werden, mit einem Gesamtaufwand von $\mathcal{O}(n+m)$ von Schritt zu Schritt

[26] Weitere Anwendungen findet die Datenstruktur der Buckets etwa bei der SMALLEST-LAST Färbungsheuristik oder der MAXIMUM_ADJAZENZ_SUCHE.

[27] Realisieren wird man diese Datenstruktur allerdings durch ein Feld $Knoten[v]$, $v \in V$, dessen Elemente jeweils einen Nachfolger- und einen Vorgänger-Eintrag für die doppelte Verkettung innerhalb der Listen haben, und ein Feld $Bucket[i]$, $i = 0, \ldots, \Delta(G)$, das gerade die Zeiger auf die Listenköpfe enthält.

fortschreiben. Ein Element minimaler Priorität kann damit in Zeit $\mathcal{O}(1)$ bestimmt werden. In dem folgenden Algorithmus BUCKET_GREEDY_MIN_IS (siehe unten) möge $Adj[v]$ für $v \in V$ die Adjazenzliste des Knotens v in G bezeichnen; sie wird während des gesamten Algorithmus nicht verändert. Jeder Knoten von G durchlebt bei Abarbeitung des Algorithmus drei Phasen. Zunächst haben alle Knoten den Status *neu*. In dem Augenblick, in dem u zu S_{Greedy} hinzugefügt wird, werden die Knoten $\Gamma_{G[U]}[u]$ *aktiv*. Ein solcher Knoten erhält den Status *fertig* erst, nachdem er aus dem Graphen entfernt wurde und die Knotengrade aller seiner Nachbarn im Restgraphen um eins vermindert wurden.
Entscheidend ist, daß die Anzahl der Operationen, die *min* involvieren, linear ist. Die Anzahl aller Inkrementierungen von *min* in der inneren `WHILE`-Schleife am Ende ist beschränkt durch $\Delta(G)+1-\delta(G)+k$, wobei k die Anzahl der Dekrementierungen von *min* in der inneren `FOR`-Schleife bezeichnet. Es gilt aber $k = \mathcal{O}(\sum_{v\in V} d(v)) = \mathcal{O}(m)$. □

```
FUNCTION BUCKET_GREEDY_MIN_IS : 2^V;

BEGIN
  FOR v ∈ V DO Status[v] := neu;                                  {U := V}
  FOR v ∈ V DO d[v] := |Adj[v]|;
  FOR i := 0 TO Δ(G) DO InitQueue Bucket[i];
  FOR v ∈ V DO InsertQueue (Bucket[d[v]], v);
  S_Greedy := ∅; min := δ(G);
  WHILE min ≤ Δ(G) DO BEGIN
    u := ExtractHead (Bucket[min]);
    S_Greedy := S_Greedy + u;
    {Entfernen der Knoten u + Adj[u] aus G und...}
    {...Aktualisierung der Knotengrade im Restgraphen:}
    entferne u aus Bucket[d[u]];
    Status[u] := fertig;
    FOR v ∈ Adj[u] DO IF Status[v] = neu THEN
      Status[v] := aktiv;
    FOR v ∈ Adj[u] DO IF Status[v] = aktiv THEN BEGIN
      entferne v aus Bucket[d[v]];
      FOR w ∈ Adj[v] DO IF Status[w] = neu THEN BEGIN
        entferne w aus Bucket[d[w]];
        d[w] := d[w] - 1;
        InsertQueue (Bucket[d[w]], w);
        IF d[w] < min THEN min = d[w];
      END;
      Status[v] := fertig;
    END; {for v}
    WHILE (min ≤ Δ(G)) ∧ (Bucket[min] = ∅) DO min := min + 1;
  END; {äußere while-Schleife}
  BUCKET_GREEDY_MIN_IS:= S_Greedy;
END; {Bucket_Greedy_Min_IS}
```

Anmerkung und Übungen

Verweise auf weitere exakte oder heuristische Algorithmen für INDEPENDENT SET bzw. CLIQUE

findet man in [GSS93c], siehe auch [Jer92].

Übung 1.5.8 *Auf Eingabe eines Graphen G gibt der folgende Algorithmus eine stabile Menge S aus, für die ebenfalls die Abschätzung (1.6)(i) gilt.*

```
FUNCTION GREEDY_MAX_IS : 2^V;

BEGIN
  S := V;
  WHILE Δ(G[S]) > 0 DO BEGIN
    wähle Knoten u ∈ S von maximalem Grad in G[S];
    S := S − u;
  END;
  GREEDY_MAX_IS:= S;
END;
```

Auch für GREEDY_MAX_IS *gibt es eine Folge $(G_k)_{k\in\mathbb{N}}$ von Graphen, so daß $\frac{\alpha(G_k)}{|S|} \to \infty$ für $k \to \infty$.*

Ein Graph G heißt *well-covered* (siehe [Plu93b]), wenn alle maximalen stabilen Mengen in G dieselbe Kardinalität haben. In solchen Graphen liefert der Greedy-Algorithmus trivialerweise eine maximum stabile Menge. Leider sind solche Graphen jedoch schwer zu erkennen:

Übung 1.5.9 [SS92] *Das Problem, zu entscheiden, ob ein Graph well-covered ist, ist* co-NP*-vollständig.* [*Hinweis:* Reduktion von SAT. Zu einer Booleschen Formel F konstruiere einen Graphen $G(F) = (V, E)$ auf der Menge der Klauseln und der Literale von F, der genau dann nicht well-covered ist, wenn F erfüllbar ist.]

Obwohl INDEPENDENT SET auf Bäumen mit Hilfe des Algorithmus GREEDY_MIN_IS in linearer Zeit lösbar ist, bleibt INDEPENDENT SET auf der Menge der Graphen, die lokal, d.h. in der Nähe eines jeden Knotens, aussehen wie ein Baum, NP-vollständig:

Übung 1.5.10 a) *Sei $g \geq 3$ eine feste ganze Zahl. Dann ist das Problem* INDEPENDENT SET *auf der Menge aller Graphen G mit Maximalgrad 3 und Taillenweite $g(G) \geq g$,* NP*-vollständig* [Pol74].
b) *Sei $0 \leq r < 1$ fest. Dann ist* INDEPENDENT SET*, eingeschränkt auf die Menge aller Graphen G mit Maximalgrad 3 und Taillenweite $g(G) \geq n^r$, $n = |V(G)|$,* NP*-vollständig* [Mur92].
[*Hinweis:* Reduktion von INDEPENDENT SET ($\Delta(G) \leq 3$). Füge auf jeder Kante von G $2t$ viele Knoten ein. Berechne die Unabhängigkeitszahl des entstehenden Graphen H und bestimme t wie gewünscht]

Übung 1.5.11 a) *Man entwickle einen linearen Algorithmus, der, gegeben ein Baum T, eine minimum dominierende Knotenmenge in T konstruiert* [CGH75].
b) *Sei $g \geq 3$ eine feste ganze Zahl. Dann ist das Problem* DOMINATING SET *auf der Menge aller Graphen G mit Taillenweite $g(G) \geq g$,* NP*-vollständig.*
c) DOMINATING SET *ist auf bipartiten Graphen* NP*-vollständig* [Ber84, CN82].
d) *Sei $0 \leq r < 1$ fest. Dann ist* DOMINATING SET*, eingeschränkt auf die Menge aller Graphen G mit Taillenweite $g(G) \geq n^r$, $n = |V(G)|$,* NP*-vollständig.*

Kapitel 2

Netzwerkflüsse und Zusammenhang

2.1 Der Markierungsalgorithmus von FORD und FULKERSON

Definitionen. Ein *gerichteter Graph* (engl. directed graph oder kurz digraph) ist ein Tupel $D = (V, A)$, wobei V eine endliche Menge ist und $A \subseteq V \times V$. Wir interpretieren die Elemente aus V als die Knoten des gerichteten Graphen D, die Elemente aus A als die gerichteten Kanten (engl. arcs) oder Bögen des Digraphen D. Beachte, daß ein gerichteter Graph Schleifen, d.h. Bögen der Form (v, v), besitzen darf. Wir betrachten jedoch nur gerichtete Graphen ohne Schleifen. Für eine Knotenmenge $X \subseteq V$ bezeichen

$$\begin{aligned} \Gamma^+(X) &:= \{y \in V \setminus X \mid \exists x \in X : (x, y) \in A\} \\ \Gamma^-(X) &:= \{y \in V \setminus X \mid \exists x \in X : (y, x) \in A\}. \end{aligned}$$

Ein *Netzwerk* $N = (V, A, c, s, t)$ ist ein gerichteter Graph (V, A) mit zwei ausgezeichneten Knoten, der Quelle (engl. source) s und der Senke (engl. target) t (siehe untenstehende Abbildungen), sowie einer nichtnegativen Kapazitätsfunktion $c : A \to \mathbb{R}_0^+$. Eine Abbildung $f : A \to \mathbb{R}_0^+$ heißt $s-t$*-Fluß* in einem Netzwerk N, falls gilt:

1. (Flußerhaltung) $\forall v \in V \setminus \{s, t\} : \quad \sum_{u \in \Gamma^-(v)} f(u, v) = \sum_{w \in \Gamma^+(v)} f(v, w)$,
2. (Zulässigkeit) $\forall a \in A : \quad 0 \leq f(a) \leq c(a)$,

wobei $f((u, v))$ durch $f(u, v)$ abgekürzt wurde. Wir stellen uns dazu vor, daß entlang der Kanten von N ein Gut von s nach t transportiert werden soll, wobei das Gut in keinem Knoten $v \in V \setminus \{s, t\}$ in das Netzwerk eingebracht oder aus ihm austreten darf und über eine Kante $a \in A$ höchstens eine Menge $c(a)$ des Gutes transportiert werden kann.

Ein Fluß f in einem Netzwerk N hat den *Wert*

$$w(f) := \sum_{u \in \Gamma^+(s)} f(s, u) - \sum_{w \in \Gamma^-(s)} f(w, s).$$

Ein Fluß f hat maximalen Wert oder heißt maximum, falls $w(f') \leq w(f)$ für alle Flüsse f' in N. Die Menge aller Flüsse in einem Netzwerk ist als Punktemenge im $\mathbb{R}^{|A|}$ abgeschlossen

und wegen (2.) auch beschränkt. Also nimmt die Funktion $w : f \mapsto w(f)$ auf diesem Kompaktum ihr Maximum an, d.h. in jedem Netzwerk existiert ein maximum Fluß.
Für Mengen $X, Y \subseteq V$ und eine Kantenfunktion $\phi : A \to \mathbb{R}_0^+$ schreiben wir im folgenden abkürzend $\overline{X} := V \setminus X$ und

$$\phi(X,Y) := \sum_{\substack{(x,y)\in A \\ x\in X, y\in Y}} \phi((x,y)).$$

Für eine Knotenmenge $X \subset V$ nennen wir $\langle X, \overline{X}\rangle$ einen $s-t-$Schnitt in einem Netzwerk N, falls $s \in X$ und $t \in \overline{X}$. Den Wert eines Flusses haben wir am $s-t-$Schnitt $\langle \{s\}, V-s\rangle$ definiert. Tatsächlich kann man ihn an jedem $s-t-$Schnitt berechnen:

Lemma 2.1.1 *Sei f ein $s-t-$Fluß und $\langle X, \overline{X}\rangle$ ein $s-t-$Schnitt in einem Netzwerk $N = (V, A, c, s, t)$. Dann gilt:*

$$w(f) \;=\; f(X, \Gamma^+(X)) - f(\Gamma^-(X), X).$$

Beweis.

$$\begin{aligned} w(f) &\overset{\text{def}}{=} f(s,\Gamma^+(s)) \;-\; f(\Gamma^-(s),s) \\ &\overset{(1)}{=} f(s,\Gamma^+(s)) \;-\; f(\Gamma^-(s),s) \;+\; \sum_{x\in X-s} \left(f(x,\Gamma^+(x)) \;-\; f(\Gamma^-(x),x)\right) \\ &= \sum_{x\in X} \left(f(x,\Gamma^+(x)) \;-\; f(\Gamma^-(x),x)\right) \\ &= f(X,X) + f(X,\overline{X}) - f(X,X) - f(\overline{X},X) \\ &= f(X,\overline{X}) - f(\overline{X},X). \end{aligned}$$

□

Sei $\langle X, \overline{X}\rangle$ ein $s-t-$Schnitt in N; dann heißt $c(X, \overline{X})$ die Kapazität des Schnittes $\langle X, \overline{X}\rangle$. Da jedes von s nach t transportierte Gut einen solchen Schnitt „irgendwann" überqueren muß, ist intuitiv klar, daß der Wert eines $s-t-$Flusses nur höchstens so groß sein kann wie die Kapazität des Schnittes $\langle X, \overline{X}\rangle$. Dies läßt sich präzise machen.

Lemma 2.1.2 *Sei f ein $s-t-$Fluß in einem Netzwerk $N = (V, A, c, s, t)$. Dann gilt:*

$$w(f) \;\leq \min_{\langle X,\overline{X}\rangle \; s-t-\text{Schnitt}} c(X,\overline{X}). \tag{2.1}$$

Beweis. Sei $\langle X, \overline{X}\rangle$ ein $s-t-$Schnitt in N. Nach dem vorigen Lemma gilt unter Benutzung der Nichtnegativität einer Flußfunktion f:

$$w(f) \;=\; f(X,\Gamma^+(X)) - f(\Gamma^-(X),X) \;\leq\; f(X,\Gamma^+(X)) \overset{(2)}{\leq} c(X,\Gamma^+(X)) \;=\; c(X,\overline{X}).$$

Da dies für alle $s-t-$Schnitte in N gilt, folgt (2.1). □

Das berühmte Max-Flow-Min-Cut-Theorem besagt nun, daß jedes Netzwerk einen Fluß enthält, dessen Wert die obere Schranke (2.1) erreicht. Bevor wir dazu kommen, zunächst noch eine Definition. Für einen Fluß f in N heißt eine Folge $\langle s, v_1\rangle, \langle v_1, v_2\rangle, \ldots, \langle v_{k-1}, x\rangle$ mit $v_0 := s$, $v_k := x$ und $v_i \in V$ für $i = 0, \ldots, k$ ein *augmentierender $s-x-$Pfad* oder x *erreichbar* von s, falls für alle $i = 0, \ldots, k-1$ eine der folgenden Alternativen gilt:

- $(v_i, v_{i+1}) \in A$ und $\epsilon_i := c(v_i, v_{i+1}) - f(v_i, v_{i+1}) > 0$,
- $(v_{i+1}, v_i) \in A$ und $\epsilon_i := f(v_{i+1}, v_i) > 0$.

Satz 2.1.3 (Max-Flow-Min-Cut-Theorem) (FORD, FULKERSON 1956; ELIAS, FEINSTEIN, SHANNON 1956) *Für einen $s-t-$Fluß f in einem Netzwerk $N = (V, A, c, s, t)$ sind äquivalent:*

(i) f hat maximalen Wert (unter allen $s-t-$Flüssen in N);

(ii) Es gibt keinen augmentierenden $s-t-$Pfad in N (bezüglich f);

(iii) f hat Wert $\min\{c(X, \overline{X}) : \langle X, \overline{X}\rangle$ *$s-t-$Schnitt$\}$.

Beweis. (i) $\Rightarrow$ (ii): gäbe es in N einen augmentierenden $s-t-$Pfad P (der Länge k für ein $k \in \mathbb{N}$), so kann der Fluß f „entlang P" um den Wert $\epsilon := \min_{0 \le i \le k-1} \epsilon_i$ vergrößert werden, indem man f auf den Vorwärtkanten $(v_i, v_{i+1}) \in A$ von P um ϵ vergrößert und auf den Rückwärtskanten $(v_{i+1}, v_i) \in A$ um ϵ verkleinert (beachte: auch der neue Fluß erfüllt das Flußerhaltungsgesetz).
(ii) $\Rightarrow$ (iii): Mit Hilfe des folgenden Markierungsverfahrens läßt sich feststellen, ob es in einem Netzwerk N für einen Fluß f einen augmentierenden $s-t-$Pfad gibt.

- markiere s;
- ist x markiert, $(x, y) \in A$ und $f(x, y) < c(x, y)$, dann markiere y;
- ist x markiert, $(y, x) \in A$ und $f(y, x) > 0$, dann markiere y.

Da V endlich ist, terminiert dieses Verfahren nach endlich vielen Schritten. Das Verfahren hat dann genau die Knoten von N markiert, die von s aus auf augmentierenden Pfaden zu erreichen sind (diese Pfade können zudem rekonstruiert werden, wenn man sich während der Markierungsphase jeweils für jeden Knoten y den Knoten x merkt, von dem aus y markiert wurde). Wenn N nun keinen augmentierenden $s-t-$Pfad enthält, so markiert das Verfahren nur eine Knotenmenge S, die t nicht enthält, und wegen $s \in S$ ist $\langle S, \overline{S}\rangle$ ein $s-t-$Schnitt. Für alle $x \in S$ und $y \in \overline{S}$ gilt dann nach Konstruktion:

$$\begin{aligned} (x,y) \in A &\Rightarrow f(x,y) = c(x,y) \\ (y,x) \in A &\Rightarrow f(y,x) = 0 \end{aligned}$$

Es folgt nach Lemma 2.1.1

$$w(f) \;=\; f(S, \Gamma^+(S)) - f(\Gamma^-(S), S) \;=\; c(S, \Gamma^+(S)) - 0 \;=\; c(S, \overline{S}).$$

Zusammen mit (2.1) folgt

$$\min_{\langle X, \overline{X}\rangle \; s-t-\text{Schnitt}} c(X, \overline{X}) \;\le\; c(S, \overline{S}) \;=\; w(f) \;\le\; \min_{\langle X, \overline{X}\rangle \; s-t-\text{Schnitt}} c(X, \overline{X}).$$

(iii) $\Rightarrow$ (i): Nach (2.1) kann kein Fluß f in N einen Wert größer als $\min\{c(X, \overline{X}) : \langle X, \overline{X}\rangle$ $s-t-$Schnitt$\}$ haben. Also hat f maximalen Wert. □

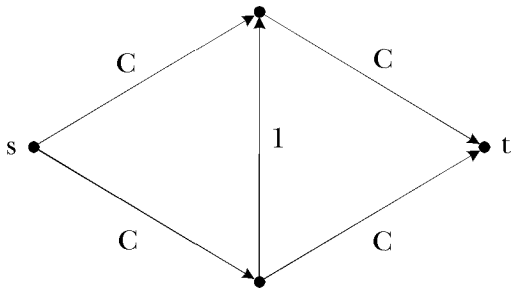
Korollar 2.1.4 (Integral-Flow-Theorem) (FORD, FULKERSON 1956)
In einem Netzwerk $N = (V, A, c, s, t)$ mit ganzzahligen Kapazitäten $c : A \to \mathbb{N}_0$ gibt es stets einen ganzzahligen Fluß $f_{\max} : A \to \mathbb{N}_0$ von maximalem Wert, der in endlich vielen Schritten konstruiert werden kann.

Beweis. Das sogenannte FORD-FULKERSON-Markierungsverfahren erhöht, ausgehend vom Fluß $f \equiv 0$, den Flußwert $w(f)$ sukzessive entlang augmentierender $s-t-$Pfade. Da die Kapazitäten ganzzahlig sind, kann der Wert ϵ, um den $w(f)$ in jedem Augmentierungsschritt erhöht wird, ebenfalls stets ganzzahlig gewählt werden, so daß f stets ein ganzzahliger Fluß ist. Da $w(f)$ in jedem Schritt um mindestens 1 erhöht wird, liefert dieses Verfahren nach höchstens

$$\min_X c(X, \overline{X}) \;=\; w(f_{\max}) \;<\; \infty$$

vielen Augmentierungsschritten einen ganzzahligen maximum Fluß $f_{\max}$. □

Die Schranke $w(f_{\max})$ für die Anzahl der Augmentierungsschritte im FORD-FULKERSON-Markierungsverfahren ist i.a. bestmöglich, vergleiche das folgende Beispiel für $C \in \mathbb{N}$:



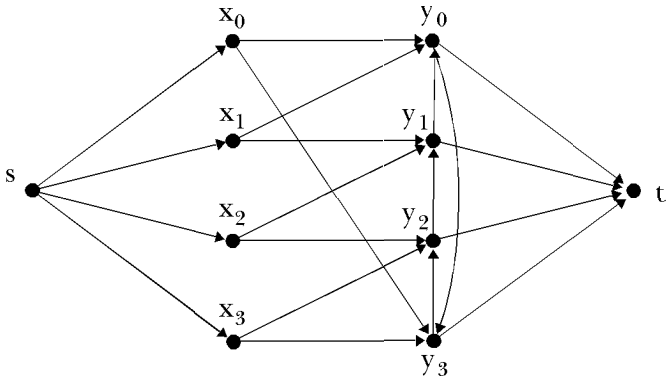
Die Anzahl der Augmentierungsschritte ist also i.a. exponentiell in der Codierungslänge des Netzwerks (beachte: $\langle C \rangle = \mathcal{O}(\log C)$).

In Netzwerken mit irrationalen Kapazitäten terminiert das FORD-FULKERSON-Markierungsverfahren nicht notwendig, ja es gibt sogar Beispiele, in denen der Flußwert nur gegen eine Zahl, die echt kleiner als der Wert eines maximum Flusses ist, konvergiert.

Beispiel: Seien $r := \frac{\sqrt{5}-1}{2} \approx 0.618$ und eine Folge $(c_i)_{i \in \mathbb{N}_0}$ definiert durch $c_i := r^i$. Dann gilt für die Summe der aus den Gliedern c_i gebildeten geometrischen Reihe

$$s := \sum_{i=0}^{\infty} c_i \;=\; \frac{1}{1-r}.$$

Im unten abgebildeten Netzwerk haben die Kanten $e_i := (x_i, y_i)$ die Kapazitäten $c(x_i, y_i) := c_i$, $i = 0, 1, 2, 3$, während alle übrigen Kanten die Kapazität s haben.



Übung 2.1.5 *In obigem Netzwerk läßt sich, ausgehend vom Fluß $f(e) \equiv 0$, der i-te augmentierende $s-t$-Pfad ($i \geq 0$) des* FORD-FULKERSON*-Markierungsverfahrens stets so wählen, daß der Flußwert sich gerade um c_i erhöht* [*Hinweis:* zeige zunächst $c_{i+1} = c_{i-1} - c_i$ für $i \in \mathbb{N}$]. □

Wenn man also die augmentierenden Pfade wie angegeben wählt, so terminiert das Markierungverfahren nicht und der Flußwert $w(f_i)$ konvergiert für $i \to \infty$ gegen s, während der maximum Fluß offenbar den Wert $4s$ hat. Analog läßt sich für jede ganze Zahl $k \geq 4$ ein Netzwerk konstruieren, in dem das FORD-FULKERSON-Markierungsverfahren u.U. nur einen Fluß liefert, dessen Wert nur $1/k$-tel des maximum Flußwertes beträgt.

Dieser Schönheitsfehler des FORD-FULKERSON-Markierungsalgorithmus läßt sich allerdings leicht beheben.

Satz 2.1.6 (EDMONDS, KARP 1972) *Sei $N = (V, A, c, s, t)$ ein beliebiges Netzwerk. Augmentiert man beim* FORD-FULKERSON*-Markierungsverfahren stets entlang eines kürzesten augmentierenden $s-t$-Pfades, so erhält man nach $\mathcal{O}(nm)$ Augmentierungsschritten ein maximum Fluß in N.* □

Da ein augmentierender Pfad mittels einer Breitensuche in linearer Zeit gefunden werden kann, ergibt sich insgesamt ein $\mathcal{O}(nm^2)$-Algorithmus.

Beweis von Satz 2.1.6:
Siehe [LP86, Jun94, TS92]. □

Anmerkungen und Übungen

Kleinste Netzwerke, auf denen das Markierungsverfahren nicht terminiert und nicht gegen den maximum Flußwert konvergiert, hat ZWICK [Zwi95] angegeben.

Effizientere Algorithmen für das Maximum-Fluß-Problem in Netzwerken finden sich in fast allen gängigen Lehrbüchern der algorithmischen Graphentheorie und der kombinatorischen Optimierung. Der schnellste bekannte Maximum-Fluß-Algorithmus hat zur Zeit Komplexität $\mathcal{O}(nm + n^{2+\epsilon})$ für jedes $\epsilon > 0$ [KRT94], siehe auch die Literaturhinweise im Anhang. Das Problem, einen minimum $s-t$-Schnitt $\langle S, T \rangle$ zu finden, wird NP-schwer, sobald man die Kardinalität der Knotenmengen S und T beschränkt [GJS76].

Wir werden in anderem Zusammenhang noch weitere solcher Maximum-Minimum-Aussagen wie die aus dem Max-Flow-Min-Cut-Theorem kennenlernen, wie z.B. den Satz 2.2.1 von MENGER, den Satz 4.2.8 von ORE, den Satz 4.2.15 von KÖNIG oder den Satz 7.3.6 von DILWORTH (einen Überblick hierzu gibt [Sch83]). Tatsächlich sind alle diese Sätze äquivalent zum Max-Flow-Min-Cut-Theorem in dem Sinn, daß sie sich alle leicht mit Hilfe des Max-Flow-Min-Cut-Theorems beweisen lassen und es auch umgekehrt implizieren, siehe z.B. [Jac83, Rei84].

Übung 2.1.7 *Man folgere das Max-Flow-Min-Cut-Theorem aus Korollar 2.1.4*
[*Hinweis:* zunächst auf rationale Kapazitäten verallgemeinern, dann ein Stetigkeitsargument].

Übung 2.1.8 *Der maximale Wert* $w(f) := \sum_{i=1}^{k} f(s_i, \Gamma^+(s_i)) - f(\Gamma^-(s_i), s_i)$ *eines Flusses* f *in einem Netzwerk mit* k *Paaren* $\langle s_i, t_i \rangle$, $i = 1, \ldots, k$, *von Quellen und Senken ist gleich der minimalen Kapazität eines Schnittes, der die Quellen von den Senken trennt*
[*Hinweis:* führe eine „Dummy"-Quelle s und eine „Dummy"-Senke t ein].

2.2 Der Satz von Menger nebst Folgerungen

Der Satz von MENGER bringt den Dualismus zwischen Trennen und Verbinden von Knoten in Graphen zum Ausdruck und ist grundlegend für alle Untersuchungen in Zusammenhangsfragen. Eine Menge von $u-v$−Pfaden heiße *kreuzungsfrei* (engl. internally disjoint), wenn je zwei der Pfade genau die Knoten u und v gemein haben, und *kantendisjunkt*, wenn keine Kante von mehr als einem Pfad benutzt wird.

Satz 2.2.1 (MENGER 1927)
[Kantenversion] *Seien* s *und* t *zwei Knoten in einem Graphen* G*. Dann ist die maximale Anzahl kantendisjunkter* $s-t$*−Pfade in* G *gleich der minimalen Kardinalität einer* $s-t$*−trennenden Kantenmenge.*
[Knotenversion] *Seien* s *und* t *zwei nicht benachbarte Knoten in einem Graphen* G*. Dann ist die maximale Anzahl kreuzungsfreier* $s-t$*−Pfade in* G *gleich der minimalen Kardinalität einer* $s-t$*−trennenden Knotenmenge.*

Beweis. [Kantenversion] Da „$\leq$" trivial ist, beweisen wir nur „$\geq$". Um das Max-Flow-Min-Cut-Theorem 2.1.3 anwenden zu können, transformieren wir den Graphen G wie folgt in ein Netzwerk $N = (V, A, c, s, t)$. Alle mit s inzidierenden Kanten werden von s weg, alle mit t inzidierenden Kanten nach t hin ausgerichtet und jede weitere Kante $\{u, v\}$ in G durch ein Paar entgegengesetzt gerichteter Kanten (u, v) und (v, u) ersetzt. Alle Kanten erhalten die Kapazität $c(e) = 1$. Dann ist jeder $s-t$−Schnitt in diesem Netzwerk N ganzzahlig und definiert daher in naheliegender Weise eine ebenso große $s-t$−trennende Kantenmenge. Die minimale Kardinalität einer $s-t$−trennenden Kantenmenge ist also höchstens so groß wie die Kapazität eines minimum Schnittes oder - nach dem Max-Flow-Min-Cut-Theorem - eines maximum Flusses. Ein maximum Fluß in N andererseits ist nach dem Integral-Fluß-Theorem 2.1.4 o.B.d.A. ganzzahlig und definiert daher ein System kantendisjunkter $s-t$−Wege bzw. o.B.d.A. -Pfade. Da jeder dieser Pfade genau den Wert 1 zum Flußwert beiträgt, ist der Wert eines maximum Flusses höchstens so groß wie die maximale Zahl kantendisjunkter $s-t$−Pfade.
[Knotenversion] Auch hier ist wieder nur „$\geq$" zu zeigen. Wir konstruieren aus G ein Netzwerk $N = (V', A, c, s, t)$, indem wir zunächst alle mit s inzidierenden Kanten von s weg, alle mit t inzidierenden Kanten nach t hin ausrichten und jede andere Kante $\{u, v\}$ durch ein Paar entgegengesetzt gerichteter Kanten (u, v) und (v, u) ersetzen. Um später von Flüssen in N auf knotendisjunkte Pfade in G schließen zu können, wird nun zudem jeder Knoten $v \in V \setminus \{s, t\}$ durch eine gerichtete Kante (v^-, v^+) ersetzt und jede in v hineinlaufende Kante $e \in \Gamma^-(v)$ nun mit v^- und jede aus v hinauslaufende Kante $e \in \Gamma^+(v)$ mit v^+ statt v verbunden. Alle Kanten erhalten die Kapazität $c(e) := 1$, $e \in A$.
Ein minimum Schnitt in N besteht dann o.B.d.A. nur aus Kanten der Form (v^-, v^+),

$v \in V$: denn angenommen, ein minimum $s-t-$Schnitt $\langle X, \overline{X}\rangle$ in N enthält eine Kante (u^+, v^-), $u^+ \in X$ und $v^- \in \overline{X} - t$; dann enthält der Schnitt $\langle X + v^-, \overline{X} - v^-\rangle$ die Kante (u^+, v^-) weniger und, da der Knoten v^- nur die eine ausgehende Kante (v^-, v^+) besitzt, höchstens die Kante (v^-, v^+) mehr, so daß $c(X + v^-, \overline{X} - v^-) \leq c(X, \overline{X})$ (falls $v^- = t$ analog) gilt und man zum Schnitt $\langle X + v^-, \overline{X} - v^-\rangle$ übergehen kann. Ein solcher Schnitt korrespondiert aber zu einer $s-t-$trennenden Knotenmenge in G. Also ist $(*)$ die Kapazität eines minimum Schnittes in N mindestens so groß wie die minimale Kardinalität einer $s-t-$trennenden Knotenmenge in G. Nach dem Max-Flow-Min-Cut-Theorem (nicht-triviale Richtung) wiederum ist $(**)$ die Kapazität eines minimum Schnittes in N höchstens gleich dem Wert eines maximum Flusses in N. Da ein maximum Fluß f in N nach dem Integral-Fluß-Theorem o.B.d.A. ganzzahlig ist und somit nur die Werte 0 oder 1 auf den Kanten $e \in A$ annimmmt, wird durch f in naheliegender Weise ein System von knotendisjunkten $s-t-$Pfaden in G definiert. Der Wert eines maximum Flusses f ist folglich $(***)$ höchstens gleich der maximalen Anzahl knotendisjunkter $s-t-$Pfade in G. Setzt man $(*)-(***)$ zusammen, ergibt sich die gewünschte Ungleichung. □

Der Satz von MENGER motiviert die folgende Definition.

Definition 2.2.2 *Sei G ein Graph auf mindestens $k+1$ Knoten. Dann heißt G k-fach* (kanten-) zusammenhängend, *falls es zwischen je zwei Knoten s und t von G mindestens k kreuzungsfreie (kantendisjunkte) $s-t-$Pfade gibt.*

Ein unzusammenhängender Graph ist 0-zusammenhängend, während Zusammenhang gleichbedeutend ist mit 1-Zusammenhang. Der K_n ist offenbar $(n-1)$-fach (kanten-) zusammenhängend, Kreise sind zweifach (kanten-) zusammenhängend. Ein Graph der Ordnung ≥ 3 ist zweifach (kanten-) zusammenhängend genau dann, wenn er zusammenhängend ist und keine Artikulation (bzw. Brücke) enthält.

Korollar 2.2.3 (WHITNEY 1932) *Sei G ein Graph auf mindestens $k+1$ Knoten, der nicht vollständig ist. Dann sind äquivalent:*

(i) G ist k-fach zusammenhängend,

(ii) jede trennende Knotenmenge hat mindestens k Knoten.

Beweis. „⇒“: Sei $A \subset V$ eine trennende Knotenmenge im Graph G (existiert, da G nicht vollständig); und seien s und t zwei Knoten in verschiedenen Zusammenhangskomponenten von $G - A$. Dann sind s und t also nicht benachbart. Da es nach Voraussetzung zwischen s und t mindestens k kreuzungsfreie $s-t-$Pfade gibt, die alle mindestens einen Knoten aus A benutzen müssen, folgt $|A| \geq k$.
„⇐“: Nach dem Satz von MENGER gibt es nach Voraussetzung zwischen je zwei nicht benachbarten Knoten in G mindestens k kreuzungsfreie Pfade. Es bleibt also zu zeigen, daß es auch zwischen zwei benachbarten Knoten s und t in G mindestens k kreuzungsfreie Pfade gibt. O.B.d.A. sei daher $k \geq 2$. Sei $e := \{s, t\}$. Wir behaupten nun: Im Graph $G - e$ hat jede $s-t-$trennende Knotenmenge mindestens $k-1$ Knoten. Zum Beweis dieser Behauptung nehmen wir an, A sei eine $s-t-$trennende Knotenmenge der Kardinalität $|A| \leq k-2$ in $G-e$. Da A in G nicht trennt, hat $G-e-A$ höchstens zwei Komponenten S und T mit $s \in S$ und $t \in T$. Da $G-e-A$ noch mindestens drei Knoten hat, hat mindestens eine dieser Komponenten noch einen weiteren Knoten, sagen wir $s \neq r \in S$. Dann ist aber $A + s$ eine $r-t-$trennende Knotenmenge der Kardinalitat $\leq k-1$ in G

im Widerspruch zur Voraussetzung. Damit ist unsere Behauptung bewiesen. Wieder nach dem Satz von MENGER gibt es nun in $G-e$ mindestens $k-1$ kreuzungfreie Pfade zwischen s und t. Zusammen mit dem $s-t-$Pfad $\{s,t\}$ haben wir also im Graph G mindestens k kreuzungsfreie $s-t-$Pfade nachgewiesen. □

Aus der Kantenversion des Satzes von MENGER folgt unmittelbar:

Korollar 2.2.4 (WHITNEY 1932) *Ein Graph G ist k-fach kantenzusammenhängend genau dann, wenn jede trennende Kantenmenge mindestens k Kanten hat.* □

Die folgende Charakterisierung von k-Zusammenhang wird sich als ein äußerst handliches Werkzeug bei Zusammenhangsfragen erweisen.

Korollar 2.2.5 (Fächersatz) *Sei G ein Graph auf mindestens $k+1$ Knoten. Dann ist G k-zusammenhängend genau dann, wenn es für alle $A=\{a_1,\ldots,a_k\}\subset V$ und alle $s\in V\setminus A$ ein System von kreuzungsfreien $s-a_i-$Pfaden, $i=1,\ldots,k$, gibt.*

Beweis. „⇐“: Zunächst folgt wegen $|V|\geq k+1$, daß $d(v)\geq k$ für alle $v\in V$. Daß es nun für jedes Knotenpaar s,t in G k kreuzungsfreie $s-t-$Pfade gibt, ersieht man, wenn man $A\subseteq\Gamma(t)$ mit $|A|=k$ setzt.
„⇒“: O.B.d.A. sei G nicht vollständig. Füge G einen neuen Knoten t hinzu, der mit allen Knoten aus A verbunden wird. Wir behaupten, daß der enstandene Graph G' dann ebenfalls k-zusammenhängend ist. Die mindestens k kreuzungsfreien $s-t-$Pfade in G' definieren dann ein $s-A-$ Pfadsystem. Zum Beweis der Behauptung nehmen wir an, es gäbe in G' eine trennende Knotenmenge S der Kardinalität $|S|\leq k-1$.
Fall 1: $t\in S$. Dann ist $S-t$ trennend in G. Jede trennende Knotenmenge in G enthält nach Korollar 2.2.3 aber mindestens k Knoten, Widerspruch.
Fall 2: $t\notin S$. Wegen $|S|\leq k-1$ ist $G-S$ zusammenhängend. Da $\Gamma(t)=A$ und $|A|=k$, ist aber auch t noch mit $G-S$ verbunden, Widerspruch.
Also enthält jede trennende Knotenmenge in G' mindestens k Knoten, und G' ist nach Korollar 2.2.3 k-zusammenhängend. □

Die Nützlichkeit des Fächersatzes kommt im Beweis des nächsten Resultats zum Ausdruck. Ein 2-zusammenhängender Graph besitzt wegen $\delta(G)\geq 2$ keine Blätter. Also ist G nach Lemma 1.1.24 kein Wald und enthält daher einen Kreis (vgl. auch Lemma 1.1.3). Wir können nun sogar zeigen:

Satz 2.2.6 (DIRAC 1960) *Je k Knoten eines k-zusammenhängenden Graphen, $k\geq 2$, liegen auf einem gemeinsamen Kreis.*

Beweis. Da G 2-zusammenhängend, gibt es nach obigem einen Kreis $C=(v_0,\ldots,v_l,v_0)$ in G. Falls C einen Knoten u der k vorgegebenen Knoten nicht enthält, so können wir C wie folgt vergrößern.
Im Fall $|C|\leq k$ gibt es nach dem Fächersatz von u aus $|C|$ knotendisjunkte $u-C-$Wege. Also können wir u beliebig zwischen zwei aufeinanderfolgende Knoten v_i und v_{i+1} auf C einfügen, indem wir die Kreiskante $\{v_i,v_{i+1}\}$ durch den v_i-u-Pfad und den $u-v_{i+1}-$Pfad ersetzen.
Im Fall $|C|>k$ gibt es nach dem Fächersatz von u aus k knotendisjunkte $u-C-$Wege. Sei $S\subseteq C$ mit $|S|=k$ die Menge der jeweils ersten Knoten auf C dieser $u-C-$Pfade. Die Knoten aus S unterteilen C in k Abschnitte. Da C höchstens $k-1$ vorgegebene

Knoten enthält, gibt es einen Abschnitt, in dem keine vorgegebenen Knoten liegen. Dieser Abschnitt läßt sich durch die ihn definierenden $u - C-$Pfade ersetzen, so daß ein Kreis entsteht, der alle bisherigen und auch den Knoten u enthält. □

Hinsichtlich der Länge eines längsten Kreises in einem k-zusammenhängenden Graphen vergleiche Übung 2.2.10 und Satz 3.2.13.

Übungen

Übung 2.2.7 *Ein Graph mit $\delta(G) \geq \lfloor n/2 \rfloor$ ist 2-zusammenhängend oder besteht aus zwei $K_{\frac{n+1}{2}}$, die in einem Knoten zusammengeheftet sind.*

Übung 2.2.8 *Man folgere die Kantenversion aus der Knotenversion des Satzes von* MENGER. [*Hinweis:* hänge je ein Blatt an s und t an und wende die Knotenversion auf den Linegraphen von G an.]

Übung 2.2.9 **a)** *Ein Graph G auf mindestens $k+1$ Knoten ist k-zusammenhängend genau dann, wenn es für alle $X, Y \in \binom{V}{k}$ ein System von k (u.U. trivialen) $X-Y-$Pfaden gibt, die paarweise keine Knoten gemeinsam haben.*
b) *Gibt es in einem Graphen für je zwei k-elementige Mengen $X = \{x_1, \ldots, x_k\} \subset V$ und $Y = \{y_1, \ldots, y_k\} \subset V$ knotendisjunkte $x_i - y_i-$Pfade für $i = 1, \ldots, k$, dann ist G mindestens $(2k-1)$-zusammenhängend.*

Übung 2.2.10 *Sei G ein k-zusammenhängender Graph, $k \geq 2$. Dann besitzt G einen Kreis der Länge* $\min\{n, 2k\}$.

Übung 2.2.11 [NI92a, NP94] *Sei $G = (V, E)$ ein k-kantenzusammenhängender Graph. Dann gibt es eine Folge spannender Bäume $T_i \subseteq E$, $i = 1, \ldots, k$, so daß für $i = 1, \ldots, k$ der Graph $G[T_1 \cup \ldots \cup T_i]$ i-kantenzusammenhängend ist.*

2.3 2-Zusammenhang, Blockstruktur und Tiefensuche

Zweifach zusammenhängende (auch nicht-separabel genannte) Graphen und Subgraphen sind, wie wir noch sehen werden, von besonderer Bedeutung. Sie besitzen eine hübsche Charakterisierung, die gelegentlich sehr von Nutzen sein kann.

Satz 2.3.1 (Ohrenzerlegung) (WHITNEY 1932b; HALIN, JUNG 1963) *Ein Graph G ist zweifach zusammenhängend genau dann, wenn er darstellbar ist als*

$$G = C \cup P_1 \cup \cdots \cup P_k, \qquad k \in \mathbb{N}_0, \tag{2.2}$$

wobei C ein Kreis ist und die P_i, $i = 1, \ldots, k$, Pfade sind, die genau ihre zwei Endpunkte mit $C \cup P_1 \cup \cdots \cup P_{i-1}$ gemeinsam haben.

Alle 2-zusammenhängenden Graphen lassen sich demnach also derart konstruieren, daß man einem Kreis sukzessive Pfade (sogenannte topologische Diagonalen) hinzufügt, die nur ihre beiden Endpunkte mit dem bisherigen Graphen gemeinsam haben!

Beweis von Satz 2.3.1:
„⇐“: durch Induktion nach k. Für $k = 0$ ist $G = C_n$. Sei also $k \geq 1$ und der Graph $G_{k-1} := C \cup P_1 \cup \cdots \cup P_{k-1}$ 2-zusammenhängend. Um zu zeigen, daß dann auch $G_k := C \cup P_1 \cup \cdots \cup P_k$ 2-zusammenhängend ist, beweisen wir, daß es für je zwei Knoten $s \neq t$ in G_k zwei

knotendisjunkte $s-t$−Pfade gibt. Seien a_1 und a_2 die Endknoten von P_k in G_{k-1}. O.B.d.A. sei $s \in G_{k-1}$ und $t \in P_k - a_1 - a_2$. Nach dem Fächersatz 2.2.5 gibt es zwei kreuzungsfreie $s - \{a_1, a_2\}$−Pfade. Diese kombiniert man mit den beiden $t - \{a_1, a_2\}$−Pfaden.
„⇒“: Als 2-zusammenhängender Graph besitzt G einen Kreis C. Setze $G_0 := C$. Sei nun $G_i = C \cup P_1 \cup \cdots \cup P_i$, $i \in \mathbb{N}_0$, bereits konstruiert. Falls $G_i \neq G$, so gibt es, da G zusammenhängt, eine Kante $e \in E \setminus E(G_i)$, die mit einem Knoten $x \in G_i$ inzidiert. Sei $e = \{x, y\}$. Falls auch $y \in G_i$, so setze $P_{i+1} := \{x, y\}$ und $G_{i+1} := G_i \cup P_{i+1}$. Andernfalls ist aufgrund des 2-Zusammenhangs von G auch $G-x$ noch zusammenhängend, und es gibt einen Pfad P in $G - x$, der y mit $V(G_i) - \{x\}$ verbindet. Setze P_{i+1} gleich P verlängert um die Kante e und $G_{i+1} := G_i \cup P_{i+1}$. Iteriere. □

Übung 2.3.2 *Die Anzahl k der Ohren P_i in einer Ohrenzerlegung ist stets gleich $m - n$.*

Auch für 3- und 4-fach zusammenhängende Graphen gibt es solche Konstruktionsverfahren, siehe [Hal89].

Satz 2.3.3 *Sei G ein Graph der Ordnung ≥ 3 ohne isolierte Knoten. Dann sind äquivalent:*

(i) G ist zweifach zusammenhängend;

(ii) Je zwei Kanten von G liegen auf einem gemeinsamen Kreis;

(iii) Je eine Knoten und eine Kante von G liegen auf einem gemeinsamen Kreis;

(iv) Je zwei Knoten von G liegen auf einem gemeinsamen Kreis;

Beweis. (i) ⇒ (ii): Wenn man in einem zweifach zusammenhängend Graphen auf einer Kante $e = \{x, y\}$ einen neuen Knoten z einfügt, d.h. die Kante e durch einen P_3 ersetzt, so ist der entstehende Graph G' ebenfalls 2-zusammenhängend: denn für alle Knotenpaare s, t mit $s \neq z \neq t$ gibt es nach wie vor zwei kreuzungsfreie $s - t$−Pfade und im Fall $t = z$ gibt es nach dem Fächersatz 2.2.5 für alle $s \neq z$ zwei kreuzungsfreie $s - \{x, y\}$−Pfade, die sich zu $s - z$−Pfaden verlängern lassen.
Seien $e_i = \{x_i, y_i\} \in E$, $i = 1, 2$, zwei Kanten in G. Füge auf jeder der Kanten e_i einen neuen Knoten z_i ein, $i = 1, 2$. In dem entstandenen Graphen gibt es also zwei kreuzungsfreie $z_1 - z_2$−Pfade, die einen Kreis in G induzieren, der e_1 und e_2 enthält.
(ii) ⇒ (iii) ⇒ (iv): wähle jeweils eine mit dem Knoten inzidierende Kante.
(iv) ⇒ (i): G ist zusammenhängend und besitzt keine Artikulation. Jede trennende Knotenmenge enthält also mindestens zwei Knoten, und nach Korollar 2.2.3 ist G 2-zusammenhängend. □

Definition 2.3.4 *Ein* Block *in einem Graphen G ist ein (inklusions-)maximaler, zusammenhängender, nicht-trivialer Subgraph von G ohne Artikulation.*

Da eine Kante einen zusammenhängenden Subgraphen ohne Artikulation darstellt, ist jede Kante in einem Block enthalten. Ein Block ist also entweder lediglich eine Kante oder ein (inklusions-) maximaler, zweifach zusammenhängender Subgraph von G (aufgefaßt als Kantenmenge). Nach dem Satz über die Ohrenzerlegung ist ein Block notwendig ein induzierter Subgraph.

Lemma 2.3.5 *Ein Kreis ist stets ganz in einem Block enthalten.*

Beweis. Ein Kreis C in einem Graphen G ist ein zusammenhängender Subgraph von G ohne Artikulation. Also ist C in einem inklusionsmaximalen Subgraphen von G dieser Art enthalten. □

Lemma 2.3.6 *Die Blöcke eines Graphen bilden eine Partition seiner Kantenmenge.*

Beweis. Wir sahen bereits, daß jede Kante in einem Block enthalten ist. Angenommen nun, eine Kante $e = \{u, v\}$ sei in zwei Blöcken $B_1 \neq B_2$ eines Graphen G enthalten. Da Blöcke inklusionsmaximale Subgraphen sind, enthält jeder der beiden Blöcke B_i mindestens eine weitere Kante $f_i \neq e$, die nicht im jeweils anderen Block enthalten ist. Insbesondere sind beide Blöcke also zweifach zusammenhängend. Nach Satz 2.3.3(ii) gibt es mithin einen Kreis C in B_2 durch die Kanten e und f_2 und damit einen Teilpfad P von C, dessen Endknoten $u \neq v$ zwar zu B_1 gehören, nicht aber dessen Kanten. Nach dem Satz 2.3.1 über die Ohrenzerlegung wäre auch $B_1 \cup P$ noch 2-zusammenhängend im Widerspruch zur Inklusionsmaximalität von B_1. □

Lemma 2.3.7 (KÖNIG 1936) *Zwei verschiedene Blöcke eines Graphen haben höchstens einen Knoten gemeinsam. Ein Knoten eines Graphen ist mehreren Blöcken enthalten genau dann, wenn er eine Artikulation ist.*

Beweis. Wären zwei Knoten x und y jeweils beide in den Blöcken $B_1 \neq B_2$ eines Graphen G enthalten, so ließe sich im Widerspruch zu Lemma 2.3.5 ein $u-v-$Pfad in B_1 mit einem $u-v-$Pfad in B_2 zu einem Kreis zusammensetzen, der nach Lemma 2.3.6 nicht ganz in einem Block enthalten wäre.
„⇒“: Sei x ein Knoten, der zu zwei Blöcken $B_1 \neq B_2$ gehört, sowie y_1 und y_2 Nachbarn von x in B_1 bzw. B_2. Wenn es einen $y_1 - y_2-$Weg gäbe, der nicht über x führte, so gäbe es einen Kreis, der Kanten aus B_1 und B_2 enthielte. Also ist x $y_1 - y_2-$trennend.
„⇐“: Sei x eine Artikulation. Dann gibt es Knoten v_1 und v_2 in G, so daß x auf jedem $v_1 - v_2-$Weg liegt. Seien y_1 und y_2 die Nachbarn von x auf einem solchen $v_1 - v_2-$Weg. Dann führt auch jeder $y_1 - y_2-$Weg über x. Also liegen die Kanten $\{x, y_1\}$ und $\{x, y_2\}$ auf keinem Kreis und gehören daher zu verschiedenen Blöcken von G. □

Die Artikulationen bilden also die Schnittstellen zwischen den Blöcken eines Graphen. Der *Block-Artikulations-Graph* (engl. block-cutpoint-graph) $bc(G)$ von G wiederspiegelt diese Struktur eines Graphen: wenn $\mathcal{B}$ die Menge der Blöcke und $\mathcal{A}$ die Menge der Artikulationen in G bezeichnet, so hat $bc(G)$ die Knotenmenge $\mathcal{B} \cup \mathcal{A}$, und zwei Knoten B und a aus $bc(G)$ sind genau dann durch eine Kante verbunden, falls B ein Block und a eine Artikulation von G ist und $a \in B$ gilt.

Satz 2.3.8 (GALLAI 1964; HARARY, PRINS 1966)
Der Block-Artikulations-Graph $bc(G)$ eines zusammenhängenden Graphen G ist ein Baum.

Beweis. Offensichtlich ist $bc(G)$ zusammenhängend. Angenommen, es gäbe einen Kreis C in $bc(G)$. Aus C läßt sich aber ein Kreis C' in G gewinnen, der abwechselnd alle Artikulationsknoten $a \in C$ und aus jedem Block $B \in C$ einen nicht-leeren Pfad enthält – im Widerspruch zu Lemma 2.3.5. □

Übungen

Übung 2.3.9 (ROBBINS 1939) *Ein Graph besitzt eine stark zusammenhängende Orientierung seiner Kanten (so daß für je zwei Knoten u und v ein gerichteter $u - v-$Weg existiert) genau dann, wenn er zweifach kanten-zusammenhängend ist (Einbahnstraßen-Regelung!).*

Übung 2.3.10 *Sei G ein Graph der Ordung ≥ 3 ohne isolierte Knoten. Dann sind äquivalent:*

(i) G ist 2-zusammenhängend;

(ii) Zu je zwei Knoten u und v und einer Kante in G gibt es einen $u-v-$Pfad, der die Kante enthält.

(iii) Zu je drei verschiedenen Knoten in G gibt es einen Pfad, der zwei (beliebig gewählte) unter ihnen verbindet und den dritten enthält.

(iv) Zu je drei verschiedenen Knoten in G gibt es einen Weg, der zwei (beliebig gewählte) unter ihnen verbindet und den dritten nicht enthält.

Übung 2.3.11 *Ein Graph G ist zweifach kanten-zusammenhängend, wenn er eine schwache Ohrenzerlegung besitzt, d.h. wenn er aus einem Kreis C durch Hinzufügen nicht nur von Pfaden P_k, sondern auch von Kreisen C_k, die genau einen Knoten mit dem bisherigen Graphen gemeinsam haben, konstruiert werden kann.*

Übung 2.3.12 ($s-t-$Numerierung) *Sei $G=(V,E)$ ein Graph und $\{s,t\} \in E$. Dann gilt: G ist zweifach zusammenhängend genau dann, wenn es eine Numerierung $\sigma : V \to \{1,\ldots,n\}$ der Knoten von G gibt (d.h. σ ist bijektiv), so daß gilt $\sigma(s)=1$, $\sigma(t)=n$ und jeder andere Knoten $v \in V-s-t$ hat mindestens einen Nachbarn mit kleinerer und mindestens einen Nachbarn mit größerer Nummer als v selbst.*

Zur Berechnung einer $s-t-$Numerierung eines Graphen in linearer Zeit siehe [ET76]. Die $s-t-$Numerierung eines Graphen kommt beispielsweise bei einem Planaritätstest zur Anwendung [LEC67].

Übung 2.3.13 *Die Relation $\sim$ auf den Kanten eines Graphen, die durch $e_1 \sim e_2 :\Leftrightarrow$ „$e_1 = e_2$ oder e_1 und e_2 liegen auf einem gemeinsamen Kreis“ definiert ist, ist eine Äquivalenzrelation. Die Äquivalenzklassen sind gerade die Blöcke von G. Eine Kante bildet für sich allein eine Äquivalenzklasse genau dann, wenn sie eine Brücke in G ist.*

Übung 2.3.14 *Ein Graph G ist genau dann Block-Artikulations-Graph eines Graphen, wenn G ein Baum ist, in dem je zwei Blätter geraden Abstand haben.*

Übung 2.3.15 [HN53] *Das Zentrum eines zusammenhängenden Graphen liegt in einem Block.*

Tiefensuche. Mittels Breiten- oder Tiefensuche konnten wir einen Graphen in linearer Zeit auf Zusammenhang testen. Wir wollen im weiteren einen ebenfalls linearen Algorithmus kennenlernen, der testet, ob ein (o.B.d.A. zusammenhängender) Graph auch 2-zusammenhängend ist. Genauer: wir werden den Algorithmus TIEFENSUCHE aus Abschnitt 1.3.1 dahingehend erweitern, daß sich mit seiner Hilfe die Blöcke und Artikulationen in einem Graphen bestimmen lassen. Dies ist äußerst nützlich bei Problemen, wo sich Problemlösungen auf den Blöcken zu einer Lösung auf dem gesamten Graph zusammensetzen lassen, wie beispielsweise bei der Färbung von Graphen oder beim Planaritätstest.

O.B.d.A. setzen wir im folgenden stets voraus, daß der vorliegende Graph G zusammenhängend ist. Betrachten wir zunächst den durch eine Tiefensuche in G definierten spannenden Baum $T := \{\{Vor[v], v\} : v \in V - s\}$; er hat eine besondere Struktur. Die Kanten $\{Vor[v], v\} \in T$ nennen wir *Baumkanten* und jede andere Kante $\{v,w\} \in E \setminus T$ des Graphen *Vorwärts-* oder *Rückwärtskante*, je nach dem, ob $DFSnum[w]$ größer oder kleiner ist als $DFSnum[v]$, wenn die Tiefensuche die Kante $\{v,w\}$ das erste Mal vom Knoten v aus betrachtet.

Lemma 2.3.16 *Es gibt keine Vorwärtskanten.*

Beweis. Ein Aufruf TIEFENSUCHE (G, v) wird erst beendet (und frühere Aufrufe, die zu diesem Aufruf geführt haben, weiterbearbeitet), wenn alle mit v inzidierenden Kanten betrachtet wurden. Eine Kante $\{v, w\}$, die das erste Mal vom Knoten v aus betrachtet wird, kann also keine Vorwärtskante sein, d.h. $DFSnum[w]$ kann nicht größer sein als $DFSnum[v]$, denn dann hätte der Aufruf TIEFENSUCHE (G, w) später stattgefunden als der Aufruf TIEFENSUCHE (G, v) und der Algorithmus hätte den Aufruf für v erst weiterbearbeitet, nachdem der Aufruf für w komplett abgearbeitet gewesen wäre, in welchem Fall die Kante $\{v, w\}$ das erste Mal vom Knoten w aus betrachtet worden wäre. □

Also zerfällt die Kantenmenge E des Graphen in die Menge T der Baumkanten und die Menge B der Rückwärtskanten. Die Kanten $e \in E = T \dot{\cup} B$ wollen wir nun so orientieren, wie der Algorithmus TIEFENSUCHE sie abläuft, d.h. wenn eine Kante $\{v, w\}$ vom Knoten v aus entdeckt und das erste Mal bearbeitet wird, so wollen wir ihr die Orientierung (v, w) geben, sie also von v nach w hin ausrichten. Die Baumkanten sind damit alle weg von der Wurzel s orientiert; und für jeden Knoten $v \in V - s$ gibt es einen eindeutigen gerichteten Pfad in T von der Wurzel s von T nach v (T ist ein gerichteter Wurzelbaum mit Wurzel s). Die Knoten $\neq v$ dieses $s - v-$Pfades heißen die *Vorfahren* von v in T, und die Knoten $w \in V$, so daß v auf dem gerichteten $s - w-$Pfad in T liegt, die *Nachkommen* von v (beachte den Unterschied zu den Begriffen Vorgänger und Nachfolger). Die Nachkommen eines Knotens $v \in V$ bilden für sich wieder einen gerichteten Wurzelbaum T_v mit Wurzel v, der Teilbaum von T ist.

Lemma 2.3.17 *Für jede Rückwärtskante $(v, w) \in B$ gilt: w ist Vorfahre von v in T.*

Jede Rückwärtskante (v, w) zeigt also auf einen Knoten w, der auf dem $s - v-$Pfad von der Wurzel s von T zum Knoten v liegt; folglich gibt es im Graphen G keine sogenannten Querkanten, die verschiedene Äste des Baumes T verbinden.

Beweis von Lemma 2.3.17:
Wegen $DFSnum[w] < DFSnum[v]$ kann der Knoten w kein Nachkomme von v sein. Angenommen also, w sei weder Vorfahre noch Nachkomme von v. Sei u der letzte gemeinsame Knoten des $s - v-$ und des $s - w-$Pfades. Dann hat die Tiefensuche offenbar den Zweig von T, der w enthält, früher bearbeitet als den Zweig, der v enthält, d.h insbesondere vor dem Aufruf TIEFENSUCHE(G, v). Also wurde die Kante $\{v, w\}$ von w aus entdeckt, Widerspruch. □

Die entscheidende Strukturinformation, um schließlich in den Sätzen 2.3.22 und 2.3.21 algorithmische Charakterisierungen für Artikulationen und Blöcke angeben zu können, liefert uns das folgende durch eine Tiefensuche definierte Feld $LowPoint[v]$, $v \in V$.

$$LowPoint[v] := \min \{\{DFSnum[v]\} \cup \{DFSnum[z] : \text{ es gibt einen Nachfahren } x \text{ von } v \text{ in } T \text{ (möglicherweise } x = v) \text{ mit } (x, z) \in B\}\}.$$

$LowPoint[v]$ ist also die kleinste DFS-Nummer $DFSnum[z]$, so daß es im Graphen $T \cup B$ einen gerichteten $v - z-$Weg gibt, der aus einer (möglicherweise leeren) Folge von Baumkanten besteht gefolgt von genau einer Rückwärtskante. Aus der Definition ergibt sich unmittelbar die folgende rekursive Darstellung von $LowPoint[v]$.

Lemma 2.3.18 *Für alle $v \in V$ ist $LowPoint[v]$ das Minimum von*

$$\{DFSnum[v]\} \cup \{DFSnum[z] : (v, z) \in B\} \cup \{LowPoint[w] : (v, w) \in T\}.$$

Damit läßt sich die Prozedur TIEFENSUCHE leicht so modifizieren, daß sie das Feld $LowPoint$ mitberechnet – in insgesamt weiterhin linearer Laufzeit.

```
PROCEDURE TiefensucheM (G, v);

BEGIN
  t := t + 1;
  DFSnum[v] := t;
  LowPoint[v] := t;
  FOR w ∈ Γ(v) DO BEGIN
    IF NOT bekannt[w] THEN BEGIN
      bekannt[w] := TRUE;
      Vor[w] := v;
      T := T + (v, w);
      TiefensucheM (G, w);
      LowPoint[v] := min {LowPoint[v], LowPoint[w]};
    END
    ELSE IF w ≠ Vor[v] THEN BEGIN
      {w ist ein Vorfahr von v und {w, v} ∉ T}
      B := B + (v, w);
      LowPoint[v] := min {LowPoint[v], DFSnum[w]};
    END; {else if}
  END; {for}
END; {TiefensucheM}

BEGIN {Hauptprogramm}
  FOR v ∈ V DO DFSnum[v] := 0;
  FOR v ∈ V DO bekannt[v] := FALSE;
  FOR v ∈ V DO Vor[v] := v;
  T := ∅;  B := ∅;                          {Baum- bzw. Rückwärtskanten}
  t := 0;
  bekannt[s] := TRUE;                          {s ist die Wurzel von T}
  TiefensucheM (G, s);
END; {Hauptprogramm}
```

Eine Partitionierung der Kantenmenge E in die Blöcke des Graphen G geschieht nun durch Ausweisung eines Stellvertreters, der sogenannten Leitkante, für jede Äquivalenzklasse (jeden Block).

Definition 2.3.19 [Läu91] *Eine Baumkante $(v, w) \in T$ heißt* Leitkante *genau dann, wenn $LowPoint[w] \geq DFSnum[v]$ gilt.*

Lemma 2.3.20 *Sei (v, w) eine Leitkante. Dann führen alle Rückwärtskanten aus dem bei v beginnenden und (v, w) enthaltenden Zweig $T_w + (v, w)$ des Baumes T in denselben Zweig zurück.*

Beweis. Sonst wäre wegen Lemma 2.3.17 $LowPoint[w] < DFSnum[v]$. □

Satz 2.3.21 *Für alle Knoten $v \neq s$ und für jede von v ausgehende Kante (v, w) gilt: (v, w) liegt genau dann in demselben Block wie die nach v hineinführende Baumkante $(u, v) \in T$, wenn (v, w) Rückwärtskante oder aber Baumkante und keine Leitkante ist.*

Beweis. Wir benutzen wieder die Charakterisierung aus Satz 2.3.3(ii), daß zwei Kanten genau dann demselben Block angehören, wenn sie auf einem (gemeinsamen) Kreis liegen. „⇒“: Angenommen, (v,w) wäre eine Leitkante. Da nach obigem Lemma alle Rückwärtskanten aus dem bei v beginnenden und (v,w) enthaltenden Zweig des Baumes T in denselben Zweig zurückführen, läge (u,v) nicht auf einem (gemeinsamen) Kreis mit (v,w), Widerspruch.
„⇐“: Wenn (v,w) eine Rückwärtskante ist und also auf einen Knoten $w \neq v$ im gerichteten $s-v$−Pfad in T zeigt, so liegen (u,v) und (v,w) offensichtlich auf einem (gemeinsamen) Kreis. Wenn (v,w) eine Baumkante, aber keine Leitkante ist, und also $LowPoint[w] < DFSnum[v]$ gilt, so gibt es im Zweig T_w eine Rückwärtskante zu einem Knoten $\neq v$ auf dem gerichteten $s-v$−Pfad in T. Wieder liegen (u,v) und (v,w) also auf einem (gemeinsamen) Kreis. □

Mithin ist die erste von Tiefensuche untersuchte Kante (v,w) eines Blockes eine Leitkante, und auch alle weiteren im Graphen $G[V(T_w)+v]$ von TIEFENSUCHEM (G,w) untersuchten Kanten gehören zu diesem Block, bis die Tiefensuche auf eine andere Leitkante stößt, die den Beginn eines neuen Blocks markiert. Jeder Block in G wird also durch einen mit einer Leitkante (v,w) beginnenden Teilbaum von T aufgespannt, den man erhält, wenn man aus $T_w + (v,w)$ alle Zweige herausschneidet, die mit einer weiteren Leitkante beginnen. Insbesondere enthält jeder Block genau eine Leitkante.

Die Zuordnung der Kanten eines Graphen zu seinen Blöcken kann also folgendermaßen von statten gehen. In einer ersten Phase bestimmt man durch Tiefensuche das Feld $LowPoint$. Die zweite Phase besteht aus einem erneuten Ablaufen der Kanten in derselben Reihenfolge durch eine neue Tiefensuche, während der man die Leitkanten bestimmt und den Kanten Labels zuweist, die angeben, zu welchem Block die entsprechende Kante gehört. Als Label bietet sich beispielsweise die Nummer der Leitkante des Blockes an. Diese Labels verwaltet man vorteilhafterweise durch einen Stack:[1] wenn immer eine Leitkante (v,w) gefunden und das erste Mal abgelaufen wird, wird sie auf den Stack gelegt; kommt man zurück zum Knoten v, wird sie wieder vom Stack entfernt. Das oberste Element des Stacks liefert dann stets die Nummer des aktuellen Blocks, so daß man jeder Kante lediglich die Nummer, die oben auf dem Stack liegt, als Label zuzuweisen braucht. Eine entsprechende Erweiterung der Basis-Prozedur TIEFENSUCHE kostet offenbar nur konstant viel Zeit pro Kante.

Satz 2.3.22

(a) Die Wurzel s von T (mit $DFSnum[s]=1$) ist genau dann Artikulation, wenn von ihr mehr als eine Leitkante ausgeht.

(b) Ein Knoten $v \in T-s$ (mit $DFSnum[s]>1$) ist genau dann Artikulation, wenn von ihm mindestens eine Leitkante ausgeht.

Beweis. **(a)** „⇐“: Wenn von s mehr als eine Leitkante ausgeht, dann gibt es in T unterhalb von s mehrere Teilbäume T_w, $w \in \Gamma(s)$. Zwischen zwei Knoten in verschiedenen

[1] Ein Stack (auch Keller oder Stapel) ist eine Datenstruktur, die es erlaubt, Elemente zum Zwischenspeichern einzufügen („auf den Stack zu legen“, Prozedur `Push`) und nach der Regel „first-in-last-out“ wieder zu entfernen / auszugeben („vom Stack herunterzunehmen“, Funktion `Pop`). Ein Stack läßt sich realisieren durch eine einfach verkettete Liste dynamisch allokierter Elemente, wobei Elemente stets am vorderen Ende eingefügt und auch wieder entfernt werden.

Teilbäumen führt nach obigem Lemma 2.3.20 jeder Weg über s. Also ist s Schnittpunkt.
„⇒“: Trivialerweise ist jede aus der Wurzel $s \in T$ herausführende Baumkante eine Leitkante. Wenn von s nur eine einzige Leitkante (s,w) ausgeht, so ist s entweder ein Blatt in G und damit keine Artikulation, oder alle anderen mit s inzidierenden Kanten sind Rückwärtskanten und als solche zusammen mit (s,w) in einem Kreis enthalten - damit liegen aber alle mit s inzidierenden Kanten in demselben Block von G (wie (s,w)), und s ist nach Satz 2.3.3(ii) keine Artikulation.
(b) „⇐“: Wenn von einem Knoten $v \neq s$ eine Leitkante (v,w) ausgeht, so führt wegen Lemma 2.3.20 jeder Weg von $Vor[v]$ zu w über v; also ist v Schnittpunkt.
„⇒“: Sei $v \neq s$ ein Knoten, von dem keine Leitkante ausgeht. Der Graph $G[V \setminus V(T_v)]$ ist zusammenhängend. Falls T_v also nur aus dem Knoten v besteht, so ist v keine Artikulation. Sei andernfalls w ein Nachfolger von v in T, d.h. (v,w) eine aus v herausführende Baumkante. Da (v,w) keine Leitkante ist, gilt $LowPoint[w] < DFSnum[v]$ und es gibt einen (gerichteten) Pfad von w zu einem Vorgänger von v, der v nicht benutzt. Also hängt der Teilbaum T_w mit $G[V \setminus V(T_v)]$ zusammen. Da dies für alle Teilbäume T_w mit w Nachfolger von v in T gilt, ist $G - v$ noch zusammenhängend und somit v keine Artikulation.
□Zusammenfassend erhalten wir:

Satz 2.3.23 (TARJAN 1972) *Die Blöcke und Artikulationen eines Graphen können in linearer Zeit berechnet werden.* □

2.4 ⋆ Die Zusammenhangszahlen

Wir wollen zwei neue Graphenparameter einführen.

Definition 2.4.1 *Die* Zusammenhangszahl $\kappa(G)$ *und die* Kantenzusammenhangszahl $\lambda(G)$ *eines Graphen G sind wie folgt definiert*

$$\begin{aligned} \kappa(G) &:= \max\{k : G \text{ ist } k\text{-fach zusammenhängend}\}; \\ \lambda(G) &:= \max\{k : G \text{ ist } k\text{-fach kantenzusammenhängend}\}. \end{aligned}$$

Beispiele. Ein unzusammenhängender Graph G hat $\kappa(G) = 0 = \lambda(G)$, ein zusammenhängender Graph mit einer Artikulation $\kappa(G) = 1$ und ein zusammenhängender Graph mit einer Brücke $\lambda(G) = 1$. Ferner gilt

$$\begin{array}{lcccl} \kappa(C_n) & = & 2 & = & \lambda(C_n), \\ \kappa(K_n) & = & n-1 & = & \lambda(K_n), \\ \kappa(K_{r,s}) & = & \min\{r,s\} & = & \lambda(K_{r,s}). \end{array}$$

Nach Korollar 2.2.3 ist für einen Graphen $G \neq K_n$ die Zahl $\kappa(G)$ gerade die minimale Kardinalität einer trennenden Knotenmenge – $\lambda(G)$ wiederum entspricht nach Korollar 2.2.4 der minimalen Kardinalität einer trennenden Kantenmenge. □

Proposition 2.4.2 (WHITNEY 1932a) *Für jeden Graphen G gilt:*

$$\kappa(G) \overset{(i)}{\leq} \lambda(G) \overset{(ii)}{\leq} \delta(G).$$

Beweis. Ad (i): Falls $\lambda(G) = 0$, so ist G nicht zusammenhängend, also $\kappa(G) = 0$. Falls $\lambda(G) = 1$, so ist G zusammenhängend, enthält aber eine Brücke $e \in E$. O.B.d.A. sei $n \geq 3$ (sonst fertig). Dann hat einer der Endknoten von e Grad ≥ 2, d.h. dieser Knoten ist eine Artikulation: $\kappa(G) = 1$.
Sei nun $\lambda(G) \geq 2$ und $F = \{e_1, \ldots, e_{\lambda(G)}\}$ eine minimal trennende Kantenmenge. Es ist zu zeigen, daß es auch eine trennende Knotenmenge mit höchstens $|F|$ Elementen gibt. Beachte: es genügt nicht, aus jeder Kante e_i einen beliebigen Endknoten zu entfernen: wir könnten dadurch eine ganze Komponente von $G - F$ entfernen und der Restgraph wäre u.U. noch zusammenhängend. Entferne aus allen Kanten außer e_1 einen Endknoten, der nicht mit e_1 inzidiert. Der entstehende Graph H ist entweder unzusammenhängend (d.h. $\kappa(G) < \lambda(G)$ und wir sind fertig) oder hat die Brücke e_1. In letzterem Fall gibt es wieder einen Knoten, dessen Entfernen G trennt oder im trivialen Graphen K_1 resultiert.
Ad (ii): Die mit einem Knoten minimalen Grades inzidierenden Kanten bilden eine trennende Kantenmenge. □

Wegen $2m \geq n\delta(G)$ (vgl. (1.1)) erhalten wir aus (ii) unmittelbar eine Aussage über die Abhängigkeit des (Kanten-) Zusammenhangs von der Kantendichte oder vom Durchschnittsgrad eines Graphen:

$$\kappa(G) \leq \lambda(G) \leq \left\lfloor \frac{2m}{n} \right\rfloor . \tag{2.3}$$

Während für die Graphen C_n, K_n und $K_{r,s}$ Gleichheit in (i) und (ii) gilt, können die Graphparameter $\kappa(G)$, $\lambda(G)$ und $\delta(G)$ i.a. beliebig weit auseinanderfallen.

Übung 2.4.3 *Für $k \in \mathbb{N}$ konstruiere man Graphen G_k mit $\kappa(G_k) \equiv 1$ und $\lambda(G_k) = k$, sowie Graphen G'_k mit $\lambda(G'_k) \equiv 1$ und $\delta(G'_k) = k$.*

Während sich die Fragen nach dem Zusammenhang und dem 2-Zusammenhang eines Graphen – wie gesehen – in linearer Zeit beantworten lassen,[2] ist die Bestimmung von $\kappa(G)$ erheblich aufwendiger. Wie der Beweis des Satzes von MENGER zeigte, kann man die maximale Anzahl knotendisjunkter $u - v$–Pfade in einem Graphen durch die Lösung eines Maximum-Fluß-Problems und d.h. in polynomieller Zeit bestimmen. Angewandt auf alle $\binom{n}{2}$ Knotenpaare in einem Graphen erhält man einen polynomiellen Algorithmus, um die Zusammenhangszahl eines Graphen zu berechnen.

Tatsächlich genügen, um $\kappa(G)$ zu bestimmen, linear viele Aufrufe einer solchen Funktion „#PFADE(G, u, v)“, die die maximale Anzahl knotendisjunkter $u - v$–Pfade in einem Graphen G zurückliefert. Betrachte dazu die folgende Funktion.

[2] ebenso für 3-Zusammenhang, siehe [HT73]

```
FUNCTION KAPPA(G(V,E)) : INTEGER;
BEGIN
  k := δ(G);
  A := ∅;
  REPEAT
    wähle u ∈ V \ A;
    A := A + u;
    FOR v ∈ (V \ A) \ Γ(u) DO
      k := min {k, #PFADE(G,u,v)};
  UNTIL |A| > k;
  KAPPA:= k;
END;
```

Satz 2.4.4 *Die Funktion* KAPPA *liefert für einen zusammenhängenden Eingabegraphen G nach höchstens $3m$ Aufrufen der Funktion* #PFADE *den Wert $\kappa(G)$ zurück.*

Basierend auf diesem Ansatz konnten EVEN und TARJAN [ET75] einen $\mathcal{O}(\sqrt{n}m^2)$-Algorithmus zur Berechnung von $\kappa(G)$ entwickeln (siehe auch [Kuc90, TS92]).

Beweis von Satz 2.4.4:
Da k innerhalb der `REPEAT`-Schleife höchstens vekleinert und $|A|$ stets vergrößert wird, terminiert der Algorithmus. Für den vollständigen Graphen K_n benötigt der Algorithmus KAPPA offenbar überhaupt keine Aufrufe der Funktion #PFADE und liefert korrekt $\text{KAPPA}(K_n) = n - 1 = \kappa(K_n)$.
Sei G nun also nicht vollständig. Dann wird aber die `REPEAT`-Schleife höchstens $(\kappa(G) + 1)-$mal durchlaufen, denn: Sei T eine beliebige minimum trennende Knotenmenge in G. Wegen $|T| = \kappa(G)$ enthält A spätestens nach dem $(\kappa(G)+1)-$ten Durchlauf der `REPEAT`-Schleife mindestens einen Knoten, der nicht in T liegt. Sei a der erste solche Knoten, den KAPPA wählt. Da G nach Herausnahme von T in mindestens zwei Komponenten zerfällt, gibt es einen Knoten $b \in V \setminus T$, der nicht zu a benachbart ist. Also gab es den Aufruf $\#\text{PFADE}(G, a, b)$, und es folgt, da T eine $a - b-$trennende Knotenmenge ist, daß

$$\text{KAPPA}(G) \;\leq\; \#\text{PFADE}(G, a, b) \;\leq\; |T| \;=\; \kappa(G).$$

Da wegen $\kappa(G) = \min_{u \neq v} \#\text{PFADE}(G, u, v)$ auch umgekehrt $\kappa(G) \leq \text{KAPPA}(G)$ gilt, liefert der Algorithmus KAPPA also auch in diesem Fall korrekt $\text{KAPPA}(G) = \kappa(G)$.
Jeder Durchlauf der `REPEAT`-Schleife enthält höchstens $n-1$ Aufrufe der Funktion PFADE. Also wird die Funktion PFADE wegen (2.3) höchstens

$$(\kappa(G)+1)\cdot(n-1) \;\leq\; \left(\frac{2m}{n}+1\right)\cdot(n-1) \;\leq\; 2m+(n-1) \;\leq\; 2m+m$$

mal aufgerufen. □

Minimum Cut. Die Berechnung der Kantenzusammenhangszahl eines Graphen G ist äquivalent damit, einen minimum Schnitt in dem Netzwerk zu finden. Offenbar genügt es dafür, zu festem $s \in V$ den kleinsten minimum $s - t-$Schnitt für alle $t \in V - s$ zu berechnen. Ein früher, unbeachteter Netzwerk-Fluß-Algorithmus für dieses Problem mit Laufzeit $\mathcal{O}(nm)$ stammt von PODDERYUGIN (siehe [Gus92]), ein weiterer von MATULA [Mat87].

Auch für das kantengewichtete minimum-Schnitt-Problem war bislang der schnellste Algorithmus ein Netzwerk-Fluß-Algorithmus mit Laufzeit $\mathcal{O}(nm\log(n^2/m))$ [HO94].

Kürzlich wurde ein ebenso eleganter wie effizienter Algorithmus gefunden, der bei der Konstruktion eines minimum Schnittes in einem Graphen ganz ohne die Berechnung von maximum Flüssen auskommt. Diesen möchten wir nun, der Übersichtlichkeit halber für das ungewichtete Problem, d.h. für die Berechnung von $\lambda(G)$, vorstellen. Für das kantengewichtete minimum-Schnitt-Problem ergibt sich analog ein $\mathcal{O}(nm+n^2\log n)$-Algorithmus [NOI94].

Zunächst ist es notwendig, den Begriff der Kantenzusammenhangszahl auf Graphen zu übertragen, die u.U. mehrere „parallele“ Kanten zwischen einem Knotenpaar enthalten; Jeder Kante $\{u,v\}\in\binom{V}{2}$ ist sozusagen eine Vielfachheit zugeordnet. Die Kantenzusammenhangszahl $\lambda(G)$ für solche sogenannte Multigraphen (d.h. Graphen mit „mehrfachen“ Kanten) ist dann ganz analog definiert als die Kardinalität einer kleinsten trennenden Kantenmenge.

Der auf Nagamochi und Ibaraki [NI92a] zurückgehende Algorithmus zur Bestimmung von $\lambda(G)$ basiert auf folgender einfacher Beobachtung: Seien u und v zwei Knoten in einem Graphen $G=(V,E)$ und F ein minimum Schnitt in G (d.h. $|F|=\lambda(G)$). Dann trennt F entweder u und v oder die beiden Knoten liegen in derselben Komponente von $G-F$. Im letzteren Fall aber ist $\lambda(G)=\lambda(G_{uv})$, wobei G_{uv} den Graphen bezeichnet, der durch Identifikation der Knoten u und v in G entsteht, d.h. G_{uv} geht aus $G-v$ hervor, indem man für jede Kante $\{v,y\}\in E(G)$, $y\in V\setminus\{u,v\}$, eine Kante $\{u,y\}$ einfügt (beachte die Symmetrie dieser Definition in u und v). Hier entstehen, falls ein Knoten $y\in V\setminus\{u,v\}$ in G sowohl mit u als auch mit v verbunden ist, in G_{uv} mehrfache Kanten zwischen zwei Knoten. Wenn wir die kleinste Kardinalität einer $u-v-$trennenden Kantenmenge (eines $u-v-$Schnittes) in einem Graphen G mit $c(u,v)$ notieren, erhalten wir also:

$$\lambda(G) \;=\; \min\,\{c(u,v),\lambda(G_{uv})\}\,. \tag{2.4}$$

Damit können wir uns bei der Berechnung von $\lambda(G)$ auf den kleineren Graphen G_{uv} zurückziehen, wenn wir nur für ein Knotenpaar u,v in G die Größe $c(u,v)$ eines minimum Schnittes kennen. Während man hierzu natürlich auch wieder Maximum-Fluß-Algorithmen einsetzen könnte, erlaubt die freie Wahl von u und v einen völlig anderen Zugang.

Für disjunkte Knotenmengen $X,Y\subseteq V$ bezeichne $d(X,Y)$ die Anzahl Kanten zwischen X und Y:

$$d(X,Y):=\;|\{\{x,y\}\in E:\;x\in X\wedge y\in Y\}|$$

(Mehrfachkanten entsprechend ihrer Vielfachheit gezählt). Für eine Ordnung $v_1,\ldots,v_n$ der Knotenmenge von G sei $V_i:=\{v_1,\ldots,v_i\}$, $i=1,\ldots,n$. Eine Ordnung $v_1,\ldots,v_n$ heißt dann *zulässig*, falls

$$d\,(v_i,V_{i-1})\;\geq\;d\,(v_j,V_{i-1})\qquad\text{für alle } 2\leq i<j\leq n. \tag{2.5}$$

Mit diesen Definitionen können wir nun die entscheidende Aussage formulieren:

Lemma 2.4.5 (Frank 1994a; Stoer, Wagner 1994) *Sei $v_1,\ldots,v_n$ eine zulässige Ordnung der Knoten eines Graphen G, $n\geq 2$. G darf mehrfache Kanten besitzen. Dann gilt:*

$$c(v_n,v_{n-1})\;=\;d(v_n,V_{n-1}).$$

Die mit dem Knoten v_n inzidierenden Kanten bilden also schon einen (globalen) minimum $v_n - v_{n-1}$–Schnitt in G.[3]

Beweis durch Induktion nach $n = |V|$. Klar für $n = 2$. O.B.d.A. enthält G keine Kanten zwischen v_n und v_{n-1}, denn solche Kanten leisten auf der linken wie auf der rechten Seite der Gleichung denselben Beitrag; es gilt also $d(v_n, V_{n-1}) = d(v_n, V_{n-2})$. Da trivialerweise $c(v_n, v_{n-1}) \leq d(v_n, V_{n-1})$, bleibt „$\geq$“ zu zeigen. Sei $(U, V \setminus U)$ ein minimum $v_n - v_{n-1}$–Schnitt in G. Es ist also $v_n \in U$ und $v_{n-1} \in V \setminus U$. Wir unterscheiden zwei Fälle.
Fall 1: $v_{n-2} \in V \setminus U$. Da $v_1, \ldots, v_{n-2}, v_n$ eine zulässige Ordnung für $G - v_{n-1}$ darstellt, folgt nach Induktionsvoraussetzung

$$\begin{array}{rcl@{\quad}rcl} c(v_n, v_{n-1}) & = & c(U, V \setminus U) & \geq & c(v_n, v_{n-2}) & & \\ & \geq & c_{G-v_{n-1}}(v_n, v_{n-2}) & \overset{(IV)}{=} & d_{G-v_{n-1}}(v_n, V_{n-2}) & = & d(v_n, V_{n-1}). \end{array}$$

Fall 2: $v_{n-2} \in U$. Da $v_1, \ldots, v_{n-2}, v_{n-1}$ eine zulässige Ordnung für $G - v_n$ darstellt, folgt nach Induktionsvoraussetzung

$$\begin{array}{rcl@{\quad}rcl} c(v_n, v_{n-1}) & = & c(U, V \setminus U) & \geq & c(v_{n-1}, v_{n-2}) & & \\ & \geq & c_{G-v_n}(v_{n-1}, v_{n-2}) & \overset{(IV)}{=} & d_{G-v_n}(v_{n-1}, V_{n-2}) & & \\ & = & d(v_{n-1}, V_{n-2}) & \overset{(2.5)}{\geq} & d(v_n, V_{n-2}) & = & d(v_n, V_{n-1}). \end{array}$$

□

Die Beobachtung (2.4) und Lemma 2.4.5 legen den folgenden Algorithmus nahe, um die Kantenzusammenhangszahl λ eines Graphen G und einen minimum Schnitt $(U, V \setminus U)$ in einem Graphen G zu berechnen:

```
PROCEDURE KANTEN_ZUSAMMENHANG;

BEGIN
  FOR v ∈ V DO X[v] := {v}; {Superknoten}
  λ := δ(G);
  FOR k := n DOWNTO 2 DO BEGIN
    berechne eine zulässige Ordnung v_1, ..., v_k von G;
    IF d(v_k, V_{k-1}) < λ THEN BEGIN
      λ := d(v_k, V_{k-1});
      U := X[v_k];
    END;
    X[v_{k-1}] := X[v_{k-1}] ∪ X[v_k];
    G := G_{v_{k-1} v_k};
  END; {for}
END;
```

Die neu zulässige Ordnung $v_1^{(k)}, \ldots, v_k^{(k)}$ von $G_{v_k^{(k+1)} v_{k+1}^{(k+1)}}$ wurde dabei wieder genauso bezeichnet wie die alte $v_1^{(k+1)}, \ldots, v_{k+1}^{(k+1)}$. Der Update von G ist so zu verstehen, daß jeweils der Knoten v_k aus G entfernt wird und alle mit ihm inzidierenden Kanten (außer ggf. $\{v_k, v_{k-1}\}$) beim Knoten v_{k-1} eingetragen werden. Die Mengen $X[v]$ führen Buch über

[3] Die Existenz zweier Knoten x und y mit $c(x, y) = d(x)$ bewies schon MADER [Mad72].

identifizierte Knotenmengen. Als Queues codiert benötigt ihr Update – wie auch der von G – $\mathcal{O}(n+m)$ Zeit. Entscheidend für die Laufzeit des Algorithmus KANTEN_ZUSAMMENHANG ist mithin der Aufwand zur Berechnung einer zulässigen Ordnung. Dieser Frage wollen wir uns nun zuwenden.

Maximum-Adjazenz-Suche. Der Algorithmus BREITENSUCHE sucht die Knoten eines Graphen in der Reihenfolge ab, in der er sie zum ersten Mal „gesehen" hat: dies wird dadurch realisiert, daß die Menge der bekannten, aber noch nicht abgearbeiteten Knoten in einer First-in-first-out Warteschlange gespeichert wird. Der Algorithmus MAXIMUM_ADJAZENZ_SUCHE (nach [Mat93], bei [TY84] auch „maximum cardinality search") hingegen bearbeitet als nächsten Knoten immer einen bekannten, aber noch nicht abgearbeiteten Knoten, der zu den meisten bereits abgearbeiteten Knoten benachbart ist. Dieser Algorithmus konstruiert offensichtlich eine zulässige Ordnung.[4] Eine naive Implementierung dieses Algorithmus benötigt $\mathcal{O}(n^2)$ Schritte. Durch Verwendung der Datenstruktur „Buckets" erhält man jedoch ähnlich wie beim Algorithmus GREEDY_MIN_IS:

Übung 2.4.6 [TY84, NI92a] *Der Algorithmus* MAXIMUM_ADJAZENZ_SUCHE *kann mit Laufzeit* $\mathcal{O}(n+m)$ *implementiert werden.*

Es folgt

Korollar 2.4.7 *Der Algorithmus* KANTEN_ZUSAMMENHANG *kann mit Laufzeit* $\mathcal{O}(nm)$ *implementiert werden.*

Beweis. Durch Breitensuche testet man zunächst, ob G überhaupt zusammenhängt. Falls ja, ist $m = \Omega(n)$. Die Prozedur MAXIMUM_ADJAZENZ_SUCHE überträgt man leicht auf Graphen mit mehrfachen Kanten. Benutzt man diese Prozedur als Subroutine zur Berechnung einer zulässigen Ordnung beim Algorithmus KANTEN_ZUSAMMENHANG, ergibt sich eine Laufzeit von $(n-1) \cdot \mathcal{O}(n+m) = \mathcal{O}(nm)$. □

Anmerkungen und Übungen

Während $\kappa(G)$ und $\lambda(G)$ in polynomieller Zeit berechenbar sind, sind die verwandten Probleme [EDGE-] DISJOINT PATHS:

INSTANZ: ein Graph $G = (V, E)$, $k \in \mathbb{N}$ und Knotenpaare $(s_1, t_1), \ldots, (s_k, t_k)$,
FRAGE: enthält G [Kanten- bzw.] Knoten-disjunkte $s_i - t_i$−Pfade für $i = 1, \ldots, k$?

NP-vollständig [Kar72], sogar schon für planare Graphen [MP93a].

Sei $G = (V, E)$ ein Graph und $\mathcal{P}$ eine Partition seiner Knotenmenge. Der Graph $G|\mathcal{P}$ geht aus G hervor durch Identifikation der Knoten in jeder Partitionsklasse von $\mathcal{P}$, d.h. $G|\mathcal{P}$ hat die Partitionsklassen von $\mathcal{P}$ als Knotenmenge und zwei Partitionsklassen sind durch eine Kante verbunden sind genau dann, wenn es in G eine Kante zwischen einem Knoten der einen und einem Knoten der anderen Partitionsklasse gibt. Enthält ein Graph k kantendiskunkte spannende Bäume, so gilt offenbar:

$$m(G|\mathcal{P}) \geq k\,(|\mathcal{P}| - 1),$$

wobei $|\mathcal{P}|$ die Anzahl der Partitionsklassen von $\mathcal{P}$ bezeichnet. Ein tiefliegendes Resultat der Graphentheorie besagt, daß diese trivialerweise notwendige Bedingung auch schon hinreichend ist:

[4] Die Prozedur MAXIMUM_ADJAZENZ_SUCHE wird später auch bei der Erkennung chordaler Graphen zur Anwendung kommen, siehe Abschnitt 7.2.

Satz 2.4.8 (TUTTE 1961, NASH-WILLIAMS 1961) *Ein einfacher, zusammenhängender Graph G besitzt eine Menge von k kantendisjunkten spannenden Bäumen genau dann, wenn*

$$m(G|\mathcal{P}) \geq k\,(|\mathcal{P}|-1)$$

für alle Partitionen $\mathcal{P}$ von V.

Für einen Beweis siehe auch [Wel76, Chapter 8.4] und [Bol78c]; einen polynomiellen Algorithmus für dieses Problem findet man in [RT85, GW92]. Für hochgradig kantenzusammenhängende Graphen kann man daher kantendisjunkte $s_i - t_i$-Pfade, $1 \leq i \leq k$, im Voraus berechnen:

Übung 2.4.9 [Gus83] *Sei G ein $2k$-kantenzusammenhängender Graph. Dann enthält G mindestens k kantendisjunkte spannende Bäume.*

Wie der K_{2k+1} zeigt, ist dieses Ergebnis bestmöglich.

Im Gegensatz zu MINIMUM CUT sind auch MAXIMUM CUT (siehe Satz 10.6.1), BISECTION (siehe Übung 10.7.2), MINIMUM k-CUT und MULTITERMINAL CUT NP-vollständig, vgl. Abschnitt 10.6.

Übung 2.4.10 *In kubischen Graphen stimmen Knoten- und Kantenzusammenhangszahl überein.*

Übung 2.4.11 a) (PLESNÍK 1975) *In Graphen mit $diam(G) = 2$ gilt $\lambda(G) = \delta(G)$.*
[*Hinweis:* sei F eine minimum trennende Kantenmenge und H_1 und H_2 die Komponenten von $G - F$. Betrachte einen Knoten in der kleineren der beiden Komponenten.]
b) *Gilt in einem Graphen G der Ordnung mindestens 2 für je zwei nicht benachbarte Knoten u und v, daß $d(u) + d(v) \geq n - 1$, dann ist $\lambda(G) = \delta(G)$.*

Übung 2.4.12 a) *Für den d-dimensionale Würfelgraphen Q_d gilt $\kappa(H) = \delta(H) = d$.*
b) (HALIN 1971) *Ein* HALIN-Graph *entsteht aus einem Baum ohne Knoten vom Grad 2, indem man die Blätter durch einen Kreis verbindet. Für einen* HALIN-*Graphen H gilt: $\kappa(H) = \delta(H) = 3$.*

Übung 2.4.13 (HARARY 1962) Je höher der Zusammenhang bzw. Kantenzusammenhang in einem Kommunikationsnetzwerk (= Graph) ist, desto weniger können Ausfälle von Kommunikationsstationen bzw. Kommunikationsleitungen die Kommunikation in dem System gefährden. Für das Problem, in einem gewichteten Graphen einen k-zusammenhängenden ($k > 1$) spannenden Subgraphen minimalen Gewichts zu finden, gibt es vermutlich keinen effizienten Algorithmus. Wir wollen das Problem hier in dem Spezialfall lösen, daß wir uns bei vorgegebener Knotenzahl die Kanten noch aussuchen dürfen (d.h. $G = K_n$, $w \equiv 1$). Bezeichne $f(k,n)$ ($g(k,n)$) die minimale Anzahl von Kanten, die ein k-fach (kanten-)zusammenhängender Graph $G = (V,E)$ mit $|V| = n$, $1 < k < n$, haben kann. Zeige:
a) $f(k,n) \geq \lceil \frac{kn}{2} \rceil$.
b) *Konstruiere einen Graphen $H_{k,n}$, der a) mit Gleichheit erfüllt.*
c) *Daraus auch $g(k,n) = \lceil \frac{kn}{2} \rceil$.*
[*Hinweis* zu b): Ordne die Knoten auf einem Kreis an und unterscheide die drei Fälle k gerade, k ungerade und n gerade, k und n ungerade.]

Man verallgemeinere den Algorithmus KANTEN_ZUSAMMENHANG auf gewichtete Graphen:

Übung 2.4.14 (NAGAMOCHI, IBARAKI 1992b) *Es gibt einen $\mathcal{O}(n^3)$-Algorithmus für das minimum-Schnitt-Problem in gewichteten Graphen.*

Übung 2.4.15 *Für einen Graphen G berechne man in linearer Zeit einen spannenden Baum im Komplement $\overline{G}$ von G bzw. entscheide, daß $\overline{G}$ nicht zusammenhängt.*

Kapitel 3

Durchlaufbarkeit

3.1 Eulersche Graphen

Die Geburtsstunde der Graphentheorie macht man heute an einer Arbeit von EULER aus dem Jahre 1736 fest, in der er das sogenannte *Königsberger Brückenproblem* löste. Gefragt war hierbei, ob die Brücken über die Pregel, die das damalige Königsberg in vier Stadtteile unterteilte, einen Rundweg erlaubten, der jede der Brücken genau einmal überquerte.

Definition 3.1.1 *Ein Graph G heißt* Eulersch, *wenn es einen geschlossenen Kantenzug in G gibt, der alle Kanten von G genau einmal durchläuft.*

Ein solcher Zykel wird dann auch Eulerscher Kantenzug genannt. EULER schloß, daß in einem Eulerschen Graphen wohl jeder Knoten, da ein Eulerscher Kantenzug genauso oft in ihn hinein- wie herausläuft, einen geraden Grad haben müsse und daß es daher einen solchen Rundweg in Königsberg nicht geben könne, weil die Insel Kneiphof fünf Brücken besaß. HIERHOLZER bewies auch den Umkehrschluß. Eine weitere Charakterisierung durch Kreise geht auf VEBLEN zurück.

Satz 3.1.2 (EULER 1736; HIERHOLZER 1873; VEBLEN 1912)
Für einen zusammenhängenden Graphen G sind äquivalent:

(i) G ist Eulersch;

(ii) Jeder Knoten in G hat geraden Grad;

(iii) Die Kantenmenge von G kann in Kreise zerlegt werden.

Beweis. (ii) $\Rightarrow$ (iii): Da G keine Blätter enthält, ist G kein Wald und enthält daher einen Kreis. Wir induzieren über die Anzahl $k \in \mathbb{N}$ von Kreisen in G. Für $k = 1$ ist die Aussage trivial. Enthalte G nun also mindestens zwei Kreise. Entfernen wir einen Kreis C aus G, so verbleibt offenbar wieder ein Graph mit geraden Knotengraden. Nach Induktionsannahme besitzt dieser eine Kreiszerlegung seiner Kantenmenge. Diese bildet zusammen mit dem Kreis C eine Kreiszerlegung von $E(G)$.
(iii) $\Rightarrow$ (i): Durch Induktion nach der Anzahl k der Kreise in einer Kreiszerlegung von G. Klar für $k = 1$. Für beliebiges $k \geq 2$ und einen Kreis C der Kreiszerlegung liefert die Induktionsannahme einen Eulerzug für jede Komponente des Graphen $(V, E \setminus E(C))$. Da G zusammenhängt, enthält der Eulerzug in jeder solchen Komponente mindestens einen

Kreisknoten. Flicke die Eulerzüge der Komponenten an diesen Stellen in den Kreis C ein. □

Eine unmittelbare Folgerung aus Satz 3.1.2 ist, daß Eulersche Graphen in linearer Zeit anhand der geraden Knotengrade erkannt werden können.

Wir kommen damit zu dem Problem, wie man einen Eulerschen Kantenzug in einem gegebenen Eulerschen Graphen konstruiert.[1] Der im folgenden dargestellte lineare Algorithmus geht auf HIERHOLZER zurück. Wenn man, beginnend in einem beliebigen Startknoten $v \in V$, einfach losläuft und seinen Weg stets mit einer noch nicht benutzten Kante fortsetzt, so kann man, da alle Knoten geraden Grad haben, in keinem Knoten (außer v selbst) „steckenbleiben" und man kehrt schließlich wieder zum Ausgangsknoten v zurück. Offensichtlich ist man einen Zykel in G abgelaufen. Man macht sich leicht an einem Beispiel klar, daß dieses Verfahren nicht notwendig einen Eulerschen Kantenzug produziert. Der Beweis des obigen Satzes 3.1.2 legt das folgende Vorgehen nahe: In einer ersten Phase zerlegt man die Kantenmenge des Graphen in disjunkte Zykeln, indem man obiges Verfahren zur Konstruktion eines Zykels iteriert, wobei die Kanten eines konstruierten Zykels jeweils aus dem Graph entfernt werden – beachte hierzu, daß auch im Restgraphen, d.h. nach Herausnahme eines Zykels, alle Knotengrade gerade sind. In einer zweiten Phase werden die Zykeln ineinander gehängt. Es ist nun leider nicht klar, wie die zweite Phase in linearer Zeit realisiert werden kann.[2]

Eine einfache Lösung besteht jedoch darin, beide Phasen untereinander zu verschränken. Wir konstruieren also, beginnend in einem beliebigen Startknoten v_0, einen ersten Zykel Z in G und entfernen die Kanten von Z aus G. Die Knotengrade im Restgraphen $G - Z$ sind wiederum allesamt gerade. Die entscheidende Beobachtung aus dem Beweisteil (iii) $\Rightarrow$ (i) von Satz 3.1.2 ist:

> *der Restgraph $G - Z$ enthält (aufgrund des Zusammenhangs von G) noch Kanten genau dann, wenn es einen auf Z liegenden Knoten gibt, an dem Kanten aus $G - Z$ anliegen.*

Wir laufen also den bereits konstruierten Zykel Z, beginnend in v_0, zur Kontrolle nochmals ab und prüfen an jedem Knoten z, ob an ihm noch Kanten aus $G - Z$ anliegen. Falls ja, werden vom Knoten z aus im Graphen $G - Z$ solange disjunkte Zykel konstruiert und *hinter* der aktuellen Position im Zykel Z in den Zykel Z eingefügt, bis an z keine Kanten aus $G - Z$ mehr anliegen. Aufgrund der Gradbedingung kann stets ein neuer Zykel konstruiert werden, solange an einem Knoten z noch Kanten aus $G - Z$ anliegen. Dadurch wiederum, daß die neu konstruierten Zykel hinter der aktuellen Position in den bestehenden Kantenzug eingefügt werden, wird gewährleistet, daß der Kontrolldurchlauf später auch die neu eingefügten Zykel abläuft. Weil G zusammenhängend ist, werden während des Kontrolldurchlaufs schließlich alle Kanten von G in den Kantenzug aufgenommen.

Eine PASCAL-Implementierung dieses Algorithmus (siehe nächste Seite) verwendet folgende Variablen. Der zu konstruierende Eulersche Kantenzug $v_0, v_1, \ldots, v_m = v_0$ in G wird durch eine einfach verkettete Knotenliste codiert; Der Zeiger *EulerZug* verweist auf den ersten Listeneintrag. Der Zeiger *Position* weist auf diejenige Stelle des Eulerschen Kantenzuges, an der sich der Kontrolldurchgang gerade befindet. Der Zeiger *Position* wird

[1] Ein verbreitetes Verfahren hierfür ist der brückenvermeidende Algorithmus von FLEURY, siehe [BM76, Ber85, Gib85, LO86, TS92, Aig93, Bry93, CH94, Wes95]. Er ist jedoch nicht linear.

[2] für eine Lösung siehe jedoch [AIS87]

erst wieder `NIL` am Ende der Liste des Eulerschen Kantenzugs, wenn also alle Kanten von G eingehängt wurden. [3]

Proposition 3.1.3 *In einem zusammenhängenden Eulerschen Graphen G konstruiert der Algorithmus* HIERHOLZER *einen Eulerschen Kantenzug in Zeit $\mathcal{O}(n+m)$.*

Beweis. Daß die Liste, auf die $EulerZug$ verweist, einen Eulerschen Kantenzug darstellt, sahen wir bereits. Zur Laufzeit: Jede Kante von G wird genau einmal in einen Zykel aufgenommen und anschließend aus G entfernt. Da die Knotenliste des Eulerzugs genau $m+1$ viele Knoten enthält, ist der Aufwand für den Kontrolldurchlauf sicher $\mathcal{O}(m)$, wenn wir in konstanter Zeit entscheiden können, ob an einem Knoten u noch Kanten aus $G-Z$ anliegen, und gegebenenfalls einen Nachbarn $v \in \Gamma(u)$ auswählen können. Dies ist, codiert man G durch eine Adjazenzliste, gewährleistet. Die Laufzeit des Algorithmus ist also sicherlich linear, wenn es gelingt, das Entfernen von Kanten aus einem durch eine Adjazenzliste gegebenen Graphen G in insgesamt linearer Zeit zu realisieren (beachte: die Position der Kante $\{u,v\}$ ist zunächst nur in der Adjazenzliste des Endknotens u bekannt!). Hierfür bieten sich die folgenden beiden Lösungsmöglichkeiten an.
(1.) Die Adjazenzlisten werden doppelt verkettet (vorwärts und rückwärts), und es wird zusätzlich ein Referenzfeld verwaltet, das für jede Kante auf jeweils *beide* Adjazenzlisten-Einträge verweist. Dann kann eine Kante in konstanter Zeit gelöscht werden. Genausogut kann auch bei jedem Eintrag für einen Nachbarn v in der Adjazenzliste eines Knotens u ein Zeiger auf den entsprechenden Eintrag von u in der Adjazenzliste von v vermerkt werden.
(2.) Man führt ein Boolesches Referenzfeld, das für jede Kante angibt, ob sie noch im Graphen enthalten ist oder ob sie schon in den Eulerschen Kantenzug aufgenommen wurde. Eine Kante $\{u,v\}$ wird dann in dem Augenblick, in dem sie in den Eulerschen Kantenzug aufgenommen wird, aus der Adjazenzliste desjenigen Endknotens gelöscht, von dem aus sie gefunden wurde (das ist der Knoten u in obigem Algorithmus), und aus der Adjazenzliste des anderen Endknotens (d.h. von v) erst, wenn an diesem Knoten irgendwann einmal eine inzidierende Kante bzw. ein Nachbar gesucht wird: bei der Suche nach inzidierenden Kanten (im Kopf der inneren `WHILE`-Schleife) ist bei dieser Variante also stets zu prüfen, ob die Kanten der Adjazenzliste auch tatsächlich noch im Graphen enthalten sind; stößt man auf eine bereits gelöschte Kante, so ist sie unmittelbar aus der Adjazenzliste zu löschen, damit man sie später nicht noch öfter „anfaßt“. Dann faßt die Operation „Löschen einer Kante“ bis zur Vollendung des Eulerschen Kantenzuges jede Kante des Graphen genau zweimal an und benötigt mithin $\mathcal{O}(m)$ Rechenoperationen. □

[3] Um exemplarisch den Umgang mit Zeigervariablen zu demonstrieren, weichen wir hier von unserer sonst eher abstrakten Darstellungsweise von Algorithmen ab.

```
PROCEDURE Konstruiere_Zykel (Position:↑ZykelElement; var G:Graph);

BEGIN
  {von Position↑.Knoten aus einen neuen Zykel konstruieren...}
  {...und hinter Position in den bestehenden Kantenzug einhängen:}
  Fortsetzung := Position↑.next;
  Z := Position;
  u := Position↑.Knoten;
  REPEAT
    wähle v ∈ Γ(u);
    E := E \ {u,v};
    {die Kante {u,v} in den Zykel einfügen:}
    new (Z↑.next);
    Z := Z↑.next;
    Z↑.Knoten := v;
    u := v;
  UNTIL u = Position↑.Knoten;
  {Ende des neu konstruierten Zykels...}
  {...mit dem Rest des bisherigen Eulerzugs verknüpfen:}
  Z↑.next := Fortsetzung;
END; {Konstruiere_Zykel}
```

```
PROCEDURE Hierholzer (G, var EulerZug:↑ZykelElement);

BEGIN
  wähle einen Startknoten v0 ∈ V;
  new (EulerZug);
  EulerZug↑.Knoten := v0;
  EulerZug↑.next := NIL;
  Position := EulerZug;
  REPEAT
    {liegen an Position↑.Knoten Kanten an,...}
    {...die noch nicht in den Kantenzug aufgenommen wurden?}
    WHILE Γ(Position↑.Knoten) ≠ ∅ DO BEGIN
      {an der Stelle Position einen neuen Zykel einhängen:}
      Konstruiere_Zykel (Position,G);
    END;
    {den Kontrolldurchlauf fortsetzen:}
    Position := Position↑.next;
  UNTIL Position↑.next = NIL;
END; {Hierholzer}
```

Anmerkungen und Übungen

Zu dem Problem, in einem nicht notwendig Eulerschen, zweifach kantenzusammenhängenden Graphen G eine Überdeckung der Kanten von G durch Kreise aus möglichst wenigen Kanten zu finden, siehe [Jac93].

Übung 3.1.4 *Der Linegraph eines Eulerschen Graphen ist Eulersch.*

Übung 3.1.5 *Sei G ein zusammenhängender Graph mit $2k$ Knoten ungeraden Grades (vgl. Übung 1.1.1). Dann läßt sich seine Kantenmenge in k (offene) Wege zerlegen und nicht in weniger.*

Übung 3.1.6 *Ein zusammenhängender Graph ist Eulersch genau dann, wenn jeder Schnitt gerade viele Kanten enthält.*

Übung 3.1.7 (Toida 1973; McKee 1984) *Ein zusammenhängender Graph ist Eulersch genau dann, wenn jede Kante in ungerade vielen Kreisen enthalten ist* [*Hinweis:* benutze Satz 3.1.2(ii). Bei „$\Rightarrow$“ bestimme für eine Kante $e = \{u, v\} \in E$ zunächst die Parität der Anzahl aller $u-v-$*Wege* in $G - e$].

Übung 3.1.8 Ein Graph G heiße *willkürlich durchlaufbar* von einem Knoten $v \in V$ aus, wenn man stets einen Eulerschen Kantenzug in G erhält, wie auch immer man die Kanten von G, beginnend in v, abläuft.
a) (Ore 1951) *Ein Graph ist willkürlich durchlaufbar von $v \in V$ aus genau dann, wenn jeder Kreis in G den Knoten v enthält.*
Wenn G von v aus willkürlich durchlaufbar ist, so
b) (Bäbler 1953) *ist $d(v) = \Delta(G)$;*
c) (Harary 1957) *kann nur v ein Artikulationsknoten von G sein.*

De Bruijn-Folgen. Eine 0-1-Folge, deren Ziffern – auf einem Kreis angeordnet – kein k-Tupel $x \in \{0,1\}^k$ doppelt enthalten, ist offenbar höchstens $|\{0,1\}^k| = 2^k$ Bits lang. Eine Folge, die auf einem Kreis angeordnet alle k-Tupel $x \in \{0,1\}^k$ enthält, ist wiederum mindestens 2^k Bits lang. Tatsächlich gilt auch hier ein Max-Min-Satz:

Übung 3.1.9 a) *Formuliere den Satz 3.1.2 für gerichtete Graphen.*
b) (De Rivière, Flye Sainte-Marie 1894) *Für jedes $k \in \mathbb{N}$ gibt es eine 0-1-Folge der Länge 2^k, die – auf einem Kreis angeordnet – jedes k-Tupel $x \in \{0,1\}^k$ genau einmal enthält*
[*Hinweis:* konstruiere einen gerichteten Graphen auf den $(k-1)$-Tupeln über $\{0,1\}$].

3.2 Hamiltonsche Graphen

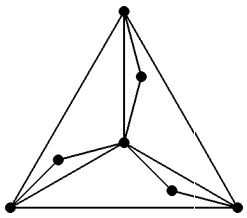
Ein dem Problem des Eulerschen Kantenzuges nur scheinbar verwandtes Problem besteht darin, einen Kreis in einem Graphen zu finden, der jeden Knoten (genau einmal) durchläuft. Ein solcher aufspannender Kreis in einem Graphen heißt dem irischen Mathematiker Sir W.R. HAMILTON [Hamil] zu Ehren *Hamiltonkreis*. Ein *Hamiltonpfad* ist ein Pfad, der alle Knoten trifft. Beachte, daß Kreise wie auch Pfade nach Definition kreuzungsfrei sind. Ein Kreis (ein Pfad) in einem Graphen ist also genau dann ein Hamiltonkreis (ein Hamiltonpfad), wenn er Länge n (bzw. $n-1$) hat. Ein Graph heißt nun *Hamiltonsch*, wenn er einen Hamiltonkreis besitzt. HAMILTON machte die Frage, welche Graphen Hamiltonsch sind, populär durch ein von ihm 1859 herausgegebenes Spiel, bei dem ein Hamiltonkreis im Ecken-Kanten-Graph des Dodekaeders (vgl. Seite 181) gesucht war. Trivialerweise sind z.B. Linegraphen von Eulerschen Graphen oder auch vollständige Graphen Hamiltonsch. Hier gilt sogar

Übung 3.2.1 *Die Kantenmenge des K_n, $n \geq 2$, zerfällt in kantendisjunkte*
a) *Hamiltonpfade genau dann, wenn n gerade ist;*
b) *Hamiltonkreise genau dann, wenn n ungerade ist.*

Ein Hamiltonscher Graph G ist natürlich zweifach zusammenhängend. Genauer gilt, wie man sich an einem Hamiltonschen Kreis in G klar macht:

Lemma 3.2.2 *Sei G ein Hamiltonscher Graph. Dann hat für alle $\emptyset \neq S \subset V$ der Graph $G - S$ höchstens $|S|$ Komponenten.* □

Der nächste Graph zeigt jedoch, daß diese Bedingung nicht hinreichend ist:



Leider lassen sich Hamiltonsche Graphen nicht so einfach charakterisieren wie Eulersche Graphen. Wie der nächste Satz zeigt, gibt es vermutlich auch keinen Algorithmus, einen Hamiltonkreis in einem Graphen zu finden, der wesentlich effizienter ist als der exponentielle Algorithmus, alle n-elementigen Teilmengen von E daraufhin zu überprüfen, ob sie einen Hamiltonkreis darstellen.

Satz 3.2.3 (KARP 1972) *Das Problem* HAMILTONIAN CIRCUIT*:*

INSTANZ: ein Graph $G = (V, E)$,
FRAGE: besitzt G einen Hamiltonkreis?

ist NP*-vollständig.* □

Für einen Beweis siehe [GJ79, PS82, Rei90, Koz92, Weg93, Jun94].

Korollar 3.2.4 *Das Problem* CIRCUMFERENCE*, den* Umfang*, d.h. die Länge eines längsten Kreises, eines Graphen zu bestimmen, ist NP-schwer.*

Man beachte, daß wir im Gegensatz dazu die Länge eines kürzesten Kreises (i.e. die Taillenweite) in polynomieller Zeit bestimmen konnten, siehe Übung 1.3.8.

Aufgrund der Klassifizierung des Hamiltonkreis-Problems als NP-vollständig interessiert man sich für hinreichende Bedingungen dafür, wann ein Graph Hamiltonsch ist. Solche Bedingungen besagen meistens, daß der Graph eine gewisse Dichte (Anzahl der Kanten in Relation zur Anzahl Knoten) haben muß - oft formuliert durch eine Forderung an den Minimalgrad $\delta(G)$. Wie jedoch der Kreis C_n zeigt, müssen Dichte oder Minimalgrad eines Hamiltonschen Graphen nicht notwendig groß sein. Ein Graph mit der Dichte $d = m/n$ besitzt aber beispielsweise immerhin einen Kreis der Länge $\Omega(d)$, vgl. Proposition 1.1.3. Des weiteren kann man Übung 2.2.10 auch so lesen, daß ein Graph auf mindestens drei Knoten mit $\kappa(G) \geq \lceil n/2 \rceil$, der mithin auch Minimalgrad $\delta(G) \geq \lceil n/2 \rceil$ hat, Hamiltonsch ist. Das folgende Lemma ist der Schlüssel, um diese Aussage erheblich zu verallgemeinern und insbesondere die hinreichende Bedingung handhabbarer zu machen.

Lemma 3.2.5 (BONDY, CHVÁTAL 1976) *Seien u und v zwei nichtadjazente Knoten eines Graphen der Ordnung n mit $d(u)+d(v) \geq n$. Dann ist G Hamiltonsch genau dann, wenn $G + \{u,v\}$ Hamiltonsch ist.*

Beweis. Wenn G Hamiltonsch ist, dann natürlich auch $G + \{u,v\}$. Sei also $G + \{u,v\}$ Hamiltonsch mit einem Hamiltonkreis C. Falls C die Kante $\{u,v\}$ nicht benutzt, so ist C offensichtlich auch ein Hamiltonkreis in G. Sei andernfalls $C = (u = v_1, \ldots, v_n = v)$. Es genügt nun, einen Index $i \in \{2, \ldots, n-2\}$ zu finden, so daß $e_R := \{u, v_{i+1}\} \in E$ und $e_L := \{v_i, v\} \in E$, denn wenn man die Kanten $\{u,v\}$ und $\{v_i, v_{i+1}\}$ aus C entfernt und die Kanten e_R und e_L hinzufügt, erhält man einen Hamiltonkreis in G. Genau eine solche „Überkreuzung“ wird aber durch die Bedingung $d(u) + d(v) \geq n$ garantiert: Sei $R := \{2 \leq i \leq n-2 : \{u, v_{i+1}\} \in E\}$ die Menge der Indizes i, so daß es für die Kante $\{v_i, v_{i+1}\}$ die e_R-Kante gibt, und entsprechend $L := \{2 \leq i \leq n-2 : \{v_i, v\} \in E\}$ die Menge der Indizes i, so daß es für die Kante $\{v_i, v_{i+1}\}$ die e_L-Kante gibt. Wegen $L \cup R \subseteq \{2, \ldots, n-2\}$ gilt offenbar $|L \cup R| \leq n-3$. Wegen $R = \{j : v_{j+1} \in \Gamma_G(u)\} \setminus \{1\}$ und $L = \{j : v_j \in \Gamma_G(v)\} \setminus \{n-1\}$ gilt aber

$$|R| + |L| = d(u) - 1 + d(v) - 1 \geq n - 2 > |L \cup R|.$$

Also ist $R \cap L \neq \emptyset$. □

Die *k-te Hamiltonhülle* $\mathcal{H}_k(G)$ eines Graphen G ist wie folgt rekursiv definiert. Falls es in G nichtadjazente Knoten u und v gibt mit $d(u)+d(v) \geq k$, so sei $\mathcal{H}_k(G) := \mathcal{H}_k(G+\{u,v\})$, sonst $\mathcal{H}_k(G) := G$. Überlegen wir uns zunächst:

Lemma 3.2.6 *Die k-te Hamiltonhülle ist für jedes $k \in \mathbb{N}$ wohldefiniert.*

Beweis. Seien $G_1 = (V, E \dot\cup \{f_1, \ldots, f_{l_1}\})$ und $G_2 = (V, E \dot\cup \{g_1, \ldots, g_{l_2}\})$ zwei k-te Hamiltonhüllen des Graphen $G = (V,E)$. o.B.d.A. sei $l_1 \geq l_2$. Angenommen nun, es gälte $G_1 \neq G_2$. Sei dann $f_i = \{x,y\}$ die erste Kante aus G_1, die nicht auch in G_2 vorkommt. Definiere $H = (V, E \cup \{f_1, \ldots, f_{i-1}\})$. Nach Definition von G_1 gilt $d_H(x) + d_H(y) \geq k$. Da H aber auch Subgraph von G_2 ist, folgt $d_{G_2}(x) + d_{G_2}(y) \geq k$, Widerspruch. □

Aus Lemma 3.2.5 schließen wir nun induktiv den

Satz 3.2.7 (BONDY, CHVÁTAL 1976)
Ein Graph ist Hamiltonsch genau dann, wenn seine n-te Hamiltonhülle Hamiltonsch ist.

Da vollständige Graphen Hamiltonsch sind, erhalten wir daraus:

Korollar 3.2.8 (ORE 1960) *Ist in einem Graphen G der Ordnung mindestens 3 die Gradsumme aller Paare nicht benachbarter Knoten $\geq n$, so ist G Hamiltonsch.* □

Korollar 3.2.9 (DIRAC 1952) *Ein Graph G der Ordnung mindestens 3 mit Minimalgrad $\delta(G) \geq n/2$ ist Hamiltonsch.* □

Beachte dazu, daß es Graphen mit $\delta(G) = n/2-1$ gibt, die nicht einmal zusammenhängend sind (Beispiel?)! Der $K_{r,r+1}$ hingegen ist sogar r-zusammenhängend und erfüllt $\delta(K_{r,r+1}) = \frac{n-1}{2}$, ist aber nicht Hamiltonsch (warum?).

Von einer ganz anderen Art ist die folgende Bedingung, wann ein Graph (oder auch die Hamiltonhülle eines Graphen) einen Hamiltonkreis besitzt.

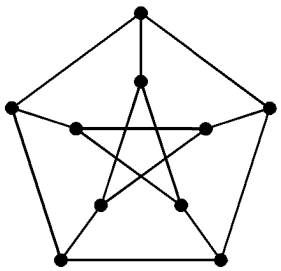
Satz 3.2.10 (CHVÁTAL, ERDŐS 1972) *Für einen Graphen G der Ordnung ≥ 3 gilt:*

$$\kappa(G) \geq \alpha(G) \quad \Rightarrow \quad G \text{ Hamiltonsch.}$$

Beweis. Aus $\alpha(G) = 1$ folgt $G = K_n$. Sei also o.B.d.A. $\kappa \geq \alpha \geq 2$, und C ein Kreis in G. Falls $|C| < n$, so sei S eine Zusammenhangskomponente von $G[V \setminus C]$ und $T = C \cap \bigcup_{s \in S} \Gamma(s)$ die Menge der Knoten auf C, die Nachbarn in S haben. Aufgrund des zweifachen Zusammenhangs von G gilt $|T| \geq 2$. Da $G[S]$ zusammenhängt, sind je zwei Knoten v_1 und v_2 aus T durch einen $v_1 - v_2$−Pfad miteinander verbunden, der nur innere Knoten aus S benutzt. Sei $C = (v_1, \ldots, v_l)$ und bezeichne v_i^* den Nachfolger des Knotens v_i auf C unter einer bestimmten Orientierung des Kreises. Wenn für ein $v_i \in T$ auch $v_i^* \in T$, so können wir offenbar den Kreis C dadurch verlängern, daß wir die Kante $\{v_i, v_i^*\}$ durch den $v_i - v_i^*$−Pfad durch S ersetzen. Dies gilt sogar allgemeiner, wann immer $\{v_i^*, v_j^*\} \in E$ für $\{v_i, v_j\} \subseteq T$: man entfernt die Kanten $\{v_i, v_i^*\}$ und $\{v_j, v_j^*\}$ aus C und vervollständigt wieder durch Einfügen der Kante $\{v_i^*, v_j^*\}$ und des $v_i - v_j$−Pfades durch S zu einem längeren Kreis.
Die Voraussetzung $\kappa \geq \alpha$ garantiert nun, daß stets so vorgegangen werden kann, solange $|C| < n$: Sei $T^* := \{t^* | t \in T\}$. Angenommen, $|C| < n$ und wir könnten den Kreis C nicht mehr wie angegeben verlängern. Dann ist einerseits $\{s, t^*\} \notin E$ für alle $s \in S$ und alle $t \in T$ ($\Rightarrow |T^*| = |T|$) und andererseits T^* stabil. Mithin ist sogar $T^* + s$ für jedes $s \in S$ stabil. Da nun T eine trennende Knotenmenge in G ist, erhalten wir den Widerspruch

$$\kappa(G) \leq |T| = |T^*| < |T^* + s| \leq \alpha(G).$$ □

Der Satz ist bestmöglich in dem Sinn, daß es nicht-Hamiltonsche Graphen mit $\alpha(G) = \kappa(G) + 1$ gibt; Beispiele solcher Graphen sind der PETERSEN-Graph[4]:



[4] Dem PETERSEN-Graphen ist sogar ein ganzes Buch gewidmet worden, siehe [HS93].

und der bereits erwähnte $K_{r,r+1}$. Zwei in einem Knoten verheftete Dreiecke zeigen zudem, daß das Resultat nicht auf Graphen mit $\lambda(G) \geq \alpha(G)$ verallgemeinert.

Während das Problem, einen Hamiltonkreis zu finden, nach Satz 3.2.3 i.a. NP-schwer ist, liefert der obige Beweis für Graphen, die die Bedingung aus Satz 3.2.10 von CHVÁTAL und ERDŐS erfüllen, einen polynomiellen Algorithmus, um diesen auch zu konstruieren: der anfängliche Kreis C kann wie in Proposition 1.1.3 konstruiert werden; dieser muß anschließend offenbar höchstens $(n-3)-$mal wie im Beweis verlängert werden.

Korollar 3.2.11 *Es gibt einen $\mathcal{O}(n \cdot m)$-Algorithmus, der in einem Graphen G entweder einen Hamiltonkreis findet oder aber eine stabile Menge S und eine trennende Menge T in G ausgibt mit $|T| < |S|$.* □

Wie die nächste Proposition zeigt, verallgemeinert die Bedingung von CHVÁTAL und ERDŐS aus Satz 3.2.10 die Bedingung von ORE aus Korollar 3.2.8. Der obige Algorithmus findet also auch in Graphen, die der ORE-Bedingung genügen, einen Hamiltonkreis.

Proposition 3.2.12 (BONDY 1978) *Sei G ein Graph der Ordnung mindestens drei und die Gradsumme nicht benachbarter Knoten von G mindestens n. Dann gilt $\kappa(G) \geq \alpha(G)$.*

Beweis. Wir zeigen $|S| \leq |A|$ für jede stabile Knotenmenge S und jede trennende Knotenmenge A. Im Fall $S \subseteq A$ ist nichts mehr zu zeigen. Sei andernfalls X eine Zusammenhangskomponente von $G - A$ mit $S \cap X \neq \emptyset$. Sei $x \in S \cap X$ und $Y := V \setminus (A \cup X)$. Wir unterscheiden:
Fall 1: $S \cap Y = \emptyset$.
Sei $y \in Y$ beliebig. Da x und y in verschiedenen Komponenten von $G - A$ liegen und daher nicht adjazent sind, gilt nach Voraussetzung

$$\begin{aligned} n &\leq |\Gamma(x)| + |\Gamma(y)| \\ &\leq |(X \cup A) \setminus S| + |Y| + |A| - 1 \\ &= n - |S| + |A| - 1 \end{aligned}$$

und also sogar $|S| < |A|$.
Fall 2: $S \cap Y \neq \emptyset$.
Sei nun $y \in S \cap Y$. Dann gilt ähnlich wie eben

$$\begin{aligned} n &\leq |\Gamma(x)| + |\Gamma(y)| \\ &\leq |X \setminus S| + |A \setminus S| + |Y \setminus S| + |A \setminus S| \\ &\leq |V \setminus S| + |A| \quad = \quad n - |S| + |A|. \end{aligned}$$

□

Graphen, die die Bedingung aus Korollar 3.2.8 erfüllen, haben tatsächlich eine noch erheblich speziellere Struktur: sie besitzen eine Anordnung $v_1, \ldots, v_n$ der Knoten, so daß $G[\{v_1, \ldots, v_k\}]$ einen C_k enthält, $k = 1, \ldots, n$, mit Ausnahme der Fälle, daß G gerade Ordnung hat und isomorph ist entweder zum $K_{n/2,n/2}$ oder zum Graphen $K_{n/2} \times K_2$, der aus zwei Kopien des $K_{n/2}$ besteht, so daß jeder Knoten jeweils mit genau einem Knoten aus der anderen Kopie verbunden ist [Hen90].

Wie gesehen, ist das Problem, die Länge eines längsten Kreises (den Umfang) in einem Graphen zu bestimmen, NP-schwer. Da jeder Kreis in einem Block enthalten ist, ist es

nur natürlich, längste Kreise in 2-zusammenhängenden Graphen zu suchen. Der folgende Satz verbessert Lemma 1.1.3 für 2-zusammenhängende Graphen und zeigt, daß jeder 2-zusammenhängende Graph schon einen Kreis der Länge mindestens $\min\{n, 2\delta(G)\}$ enthält. Dies bedeutet, daß in Übung 2.2.10 der Zusammenhang nicht die entscheidende Rolle spielt (vgl. Proposition 2.4.2 und anschließende Bemerkung).

Satz 3.2.13 (ORE 1960; BERMOND 1976; LINIAL 1976) *Sei G ein 2-zusammenhängender Graph, so daß für je zwei nicht adjazente Knoten u und v gilt: $d(u) + d(v) \geq d$. Dann besitzt G einen Kreis der Länge mindestens* $\min\{n, d\}$.

Der Graph, der aus zwei $K_{\frac{n+1}{2}}$ besteht, die in einem Knoten zusammengeheftet sind, zeigt, daß es hier tatsächlich notwendig ist, 2-Zusammenhang vorauszusetzen. Auch dieser Satz impliziert die Korollare 3.2.8 und 3.2.9, da deren Voraussetzungen implizit den 2-Zusammenhang mit beinhalten (angenommen, ein solcher Graph G enthielte eine Artikulation a; seien X und Y zwei Komponenten in $G - a$ sowie $x \in X$ und $y \in Y$. Dann sind x und y nicht benachbart und hätten daher wegen $d(x) + d(y) \geq n$ mindestens zwei Nachbarn gemeinsam, Widerspruch).

Beweis von Satz 3.2.13 (nach [Lov79]).
Schritt I: Konstruktion eines „langen" Pfades in G. Beginne mit einem beliebigen Pfad $P = (v_0, \ldots, v_\ell)$. In Erweiterung der Beweisidee von Proposition 1.1.3 gibt es (solange $V(P) \neq V$ ist) zwei Möglichkeiten, den Pfad P zu verlängern bzw. aus P einen längeren Pfad zu konstruieren:

(1.) Man kann den Pfad über beide Endknoten hinweg solange verlängern, bis $\Gamma(v_0) \subset P \supset \Gamma(v_\ell)$;

(2.) Falls es einen Index $i \in \{1, \ldots, \ell - 2\}$ gibt, so daß $\{v_0, v_{i+1}\} \in E$ und $\{v_i, v_\ell\} \in E$, so enthält der Kreis $C := (v_0, \ldots, v_i, v_\ell, \ldots, v_{i+1})$ alle Pfadknoten. Falls $\{v_0, v_\ell\} \in E$, so enthält $C := (v_0, \ldots, v_\ell)$ alle Pfadknoten. Wegen $V(P) \neq V$ gibt es einen Knoten $x \in V \setminus V(P)$, der aufgrund des Zusammenhangs von G durch einen $x - C$–Pfad mit C verbunden ist. Sei v_k, $0 \leq k \leq \ell$, der erste Knoten dieses Pfades, der auf C liegt. Dann ist $(x, \ldots, v_k, \ldots, v_\ell, v_0, \ldots, v_{k-1})$ ein längerer Pfad als P.

Den jeweils erhaltenen Pfad wollen wir dann wieder P nennen. Man führe nun beide Operationen im Wechsel solange durch, bis $V(P) = V$ oder sich keine der beiden Operationen mehr anwenden läßt.

Schritt II: Konstruktion eines Kreises C auf Grundlage des in Schritt I erhaltenen Pfades $P = (v_0, \ldots, v_\ell)$. Falls $\{v_0, v_\ell\} \in E$, so muß $(v_0, \ldots, v_\ell, v_0)$ wegen (2.) schon ein Hamiltonkreis sein. Sei also o.B.d.A. $\{v_0, v_\ell\} \notin E$. Wegen (1.) liegen die Nachbarn der Endknoten des Pfades P alle auf P. Wir unterscheiden:

Fall 1: $\exists i, j: \quad 1 \leq i < j \leq \ell - 1 \ \wedge \ \{v_0, v_j\} \in E \ \wedge \ \{v_i, v_\ell\} \in E$

Seien solche i und j mit $|j - i| \stackrel{!}{=} \min$ gewählt. Aus (2.) folgt, daß $dist := |j - i| > 1$. Nach Wahl von i und j ist keiner der Knoten $v_{i+1}, \ldots, v_{j-1}$ auf dem Pfadabschnitt zwischen v_i und v_j mit v_0 oder v_ℓ benachbart. Der Kreis $C := (v_0, \ldots, v_i, v_\ell, \ldots, v_j, v_0)$ enthält also mindestens die Knoten v_ℓ und $\Gamma(v_\ell)$ sowie alle Knoten auf P bis auf v_{j-1} ($\neq v_i$), die Vorgänger auf P von Nachbarn von v_0 sind. Wegen $dist > 1$ sind diese $1 + d(v_\ell) + (d(v_0) - 1) \geq d$ Knoten alle verschieden.

Fall 2: $i := \max\{k : v_k \in \Gamma(v_0)\} \leq \min\{k : v_k \in \Gamma(v_\ell)\} =: j$
Betrachte die Kreise $C_i := (v_0, \ldots, v_i, v_0)$ und $C_j := (v_j, \ldots, v_\ell, v_j)$. Aufgrund des 2-Zusammenhangs von G gibt es zwei knotendisjunkte $C_i - C_j$−Pfade, die jeweils nur ihre Endknoten mit den Kreisen C_i und C_j gemeinsam haben (um dies einzusehen, füge man auf einer Kante e_i auf C_i und einer Kante e_j auf C_j je einen Knoten z_i bzw. z_j ein. Der entstehende Graph ist wiederum 2-zusammenhängend, besitzt also zwei knotendisjunkte $z_i - z_j$−Pfade, aus denen sich die gewünschten $C_i - C_j$−Pfade konstruieren lassen). O.B.d.A. beginnt einer beiden $C_i - C_j$−Pfade in v_i (falls nicht, so laufe man von v_i auf P in Richtung v_j, bis man auf v_j oder einen der Pfade P_1 oder P_2 trifft; wenn man auf einen Pfad gestoßen ist, ersetzt man dessen Anfangsstück, ansonsten komplett einen der Pfade P_1 oder P_2). Analog endet o.B.d.A. einer der Pfade in v_j. Ob einer der beiden Pfade P_1 und P_2 nun ein $v_i - v_j$−Pfad ist oder nicht, in jedem einzelnen dieser Fälle konstruiert man leicht einen Kreis C, der die Pfade P_1 und P_2 enthält und alle der Knoten $v_0, \Gamma(v_0), v_\ell, \Gamma(v_\ell)$. Da $\Gamma(v_0)$ und $\Gamma(v_\ell)$ höchstens einen Knoten gemeinsam haben (falls nämlich $i = j$), enthält der Kreis C also mindestens $d(v_0) + 1 + d(v_\ell) + 1 - 1 \geq d + 1$ viele Knoten. □

Der Beweis ergibt darüberhinaus die folgende Verschärfung von Proposition 1.1.3 für Pfade.

Korollar 3.2.14 *Sei G ein Graph und $d := \min\{d(u)+d(v) : u \text{ und } v \text{ nicht benachbart}\}$. Dann gilt $d \geq 2\delta(G)$, und G enthält einen Pfad auf mindestens $\min\{n, 1+d\}$ vielen Knoten.*

Übungen und Anmerkungen

Übung 3.2.15 *Man gebe vier Graphen an, die jeweils Eulersch bzw. nicht Eulersch und zugleich Hamiltonsch bzw. nicht Hamiltonsch sind.*

Übung 3.2.16 a) *Für einen bipartiten Hamiltonschen Graphen $G = (V_1, V_2, E)$ gilt: $|V_1| = |V_2|$.*
b) *Der zweidimensionale Gittergraph $P_r \times P_s$ ist Hamiltonsch genau dann, wenn r oder s gerade ist.*

Übung 3.2.17 [Gil58] *Der d-dimensionale Würfel Q_d, $d \geq 2$, ist Hamiltonsch.*

Das Problem HAMILTONIAN CIRCUIT bleibt NP-vollständig für Linegraphen [Ber81]. Es gilt jedoch:

Übung 3.2.18 *Der Linegraph eines Hamiltonschen Graphen ist Hamiltonsch.*

Übung 3.2.19 a) [Kri75] HAMILTONIAN CIRCUIT *ist* NP*-vollständig für bipartite Graphen.*
b) *Das Entscheidungsproblem* HAMILTONIAN PATH*: „gegeben ein Graph - enthält er einen Hamiltonpfad?“ ist* NP*-vollständig.* [*Hinweis:* Für einen Knoten $u \in V$ durchläuft ein Hamiltonkreis in G genau zwei Kanten der Form $\{u, v_i\}$ mit $v_i \in \Gamma(u)$. Modifiziere G so, daß jeder Hamiltonpfad in G' genau eine dieser Kanten benutzt.]

HAMILTONIAN CIRCUIT bleibt selbst für 3-zusammenhängende, kubische, planare [GJT76] oder 3-zusammenhängende, kubische, bipartite [ANS80] Graphen NP-vollständig.

Übung 3.2.20 *Das Problem, in einem Graphen G einen spannenden Baum von minimalem Maximalgrad zu konstruieren, ist* NP*-schwer.*

Überraschenderweise gibt es jedoch einen polynomiellen Algorithmus, der einen spannenden Baum in G konstruiert, dessen Maximalgrad den minimal möglichen Maximalgrad nur um höchstens 1 überschreitet [FR94].

Übung 3.2.21 (NASH-WILLIAMS 1971)
Ein r-regulärer Graph der Ordnung $2r+1$ ist Hamiltonsch.

Beachte, daß der Satz 3.2.7 hier nicht greift. Außerdem ist die Aussage bestmöglich hinsichtlich der Ordnung: der Graph $K_{r+1} \cup K_{r+1}$ ist trivialerweise nicht Hamiltonsch. Für Graphen allerdings, die die ohnehin notwendige Bedingung des 2-Zusammenhangs erfüllen, wurde die Aussage sukzessive verschärft. Das bestmögliche Resultat ist, siehe [Vos91]: Ein 2-zusammenhängender r-regulärer Graph G mit $n \leq 3r+3$, $r \geq 3$, ist Hamiltonsch, außer G ist der PETERSEN-Graph oder der PETERSEN-Graph, bei dem ein Knoten durch ein Dreieck ersetzt wurde.

Übung 3.2.22 *Man konstruiere* **a)** *einen zusammenhängenden 4-regulären Graphen auf 11 Knoten* **b)** *einen 2-zusammenhängenden, 4-regulären Graphen jeweils ohne Hamiltonkreis.*
[*Hinweis:* evtl. Lemma 3.2.2]

Übung 3.2.23 [Réd34] *Ein Turnier ist ein vollständiger Graph $K_n = (V, \binom{V}{2})$, wobei jeder Kante eine Richtung zugewiesen ist, d.h. für jede Kante $e \in \binom{V}{2}$ ist einer seiner Endknoten als Kopf, der andere als Fuß ausgezeichnet und die Kante vom Fuß zum Kopf gerichtet. Jedes Turnier enthält einen (gerichteten) Hamiltonpfad, d.h. eine Anordnung der Knoten $v_1, \ldots, v_n$, so daß für $i = 1, \ldots, n-1$ die Kante $\{v_i, v_{i+1}\}$ von v_i nach v_{i+1} gerichtet ist.*

Für einen Graphen G bezeichne

$$\beta(G) := \min \left\{ \frac{\langle A, \bar{A} \rangle}{|A||\bar{A}|} : A \subset V, \emptyset \neq A \neq V \right\} \in [0,1]$$

die *Schnittdichte* (engl. bisection width) von G; ihre Berechnung ist NP-schwer [MS90].

Übung 3.2.24 (LU 1992) **a)** $\frac{4\kappa(G)}{n^2} \leq \beta(G) \leq \frac{2\kappa(G)}{n+\kappa(G)}$.
b) $\beta(G) \geq \frac{(2-\beta(G))\alpha(G)}{n} \quad \Rightarrow \quad G$ *Hamiltonsch.*

Tatsächlich ist ein Graph G der Ordnung mindestens 3 schon Hamiltonsch, wenn er nur $\beta(G) \geq \alpha(G)/n$ erfüllt [Lu92, Lu94]. Dieses Ergebnis ist aufgrund des $K_{r,r+1}$ scharf. Der Graph $G := (K_{2r,2r} \cup K_{2r,2r}) * K_{3r}$ (wobei die Operation $*$ vollständiges Verbinden meint) beispielsweise mit $\alpha(G) = 4r$, $\kappa(G) = 3r$ und $\beta(G) = 3/7$ ist nach diesem Kriterium, aber nicht nach Satz 3.2.10 Hamiltonsch.

Übung 3.2.25 (ORE 1960) *Ist in einem Graphen G die Gradsumme nicht benachbarter Knoten mindestens $n-1$, so besitzt G einen Hamiltonpfad.*

Ein Graph heißt *Hamiltonsch zusammenhängend*, wenn es zwischen je zwei Knoten u und v einen Hamiltonschen $u-v$–Pfad gibt. ORE (1963) zeigte, daß Graphen, in denen die Gradsumme nicht benachbarter Knoten $\geq n+1$ ist, Hamiltonsch zusammenhängend sind. Man leite dies sowohl aus Satz 3.2.10 wie auch aus Satz 3.2.7 ab:

Übung 3.2.26 **a)** *Sei in einem Graphen G die Gradsumme nicht benachbarter Knoten $\geq n+1$. Dann gilt: $\kappa(G) \geq \alpha(G)+1$.*
b) *Ein Graph G mit $\kappa(G) \geq \alpha(G)+1$ ist Hamiltonsch zusammenhängend.*
c) *Ist die $(n+1)$–te Hamiltonhülle $\mathcal{H}_{n+1}(G) \cong K_n$, so ist G Hamiltonsch zusammenhängend.*

In letzter Zeit hat man insbesondere nach hinreichenden Bedingungen gesucht, die mehr als zwei unabhängige Knoten involvieren. In [BBRV89] beispielsweise wird gezeigt: Ein Graph G mit $\kappa(G) \geq 2$, so daß $d(u)+d(v)+d(w) \geq n+\kappa(G)$ für alle unabhängigen Tripel $\{u,v,w\} \in \binom{V}{3}$ von Knoten, ist Hamiltonsch.

Auch die Kardinalität der Vereinigung der Nachbarschaften unabhängiger Knotenmengen wird als Maßzahl untersucht [Fra86]: Ist G k-zusammenhängend und

$$\left|\bigcup_{s\in S}\Gamma(s)\right| \;>\; \frac{k}{k+1}(n-1)$$

für alle stabilen Mengen $S \in \binom{V}{k}$ der Kardinalität k, so ist G Hamiltonsch. (Für einen Beweis siehe auch [Gou88].) Verallgemeinerungen hiervon werden in [CS94] diskutiert bzw. hergeleitet.

In [AC81] findet sich eine Verallgemeinerung des Satzes 3.2.7. Für nicht benachbarte Knoten u und v in einem Graphen G bezeichne α_{uv} die Kardinalität einer größten, u und v enthaltenden stabilen Menge und $\Gamma_{uv} := |\Gamma(u)\cap\Gamma(v)|$. Dann gilt: Sei G ein Graph mit $\alpha_{uv}(G) \le \Gamma_{uv}(G)$ für alle nicht benachbarten Knoten u und v in G. Dann ist G Hamiltonsch genau dann, wenn $G+\{u,v\}$ Hamiltonsch ist.

Ein frühes Ergebnis von Erdős und Gallai besagt: ein Graph mit mindestens $m \ge n$ Kanten enthält einen Kreis der Länge mindestens $\frac{2m}{n-1} > \frac{1}{n}\sum_{v\in V} d(v)$, siehe [EG59, FMR92]. Fan [Fan84] konnte die Voraussetzung von Satz 3.2.13 wie folgt abschwächen: Ein 2-zusammenhängender Graph G mit $\max\{d(u),d(v)\} \ge d/2$ für alle $\{u,v\} \in \binom{V}{2}$ mit $dist(u,v) = 2$ besitzt einen Kreis der Länge mindestens d. Fournier und Fraisse [FF85] verallgemeinerten Satz 3.2.13 auf stärker zusammenhängende Graphen: Ein k-zusammenhängender Graph, $k \ge 2$, in dem die Gradsumme von je $k+1$ unabhängigen Knoten $\ge d$ ist, enthält einen Kreis der Länge mindestens $\min\{n, \frac{2d}{k+1}\}$. Dies beinhaltet auch Satz 3.2.10.

Im Hinblick auf Satz 2.2.6 wird in [EGL91] gezeigt, daß ein k-zusammenhängender Graph der Ordnung $\ge 2\delta(G)$ für jede Menge $S \in \binom{V}{k}$ einen Kreis der Länge $\ge 2\delta(G)$ besitzt, der die Knoten aus S enthält.

Wir bemerken schließlich, daß keines der bisherigen Kriterien einen der 33.439.123.484.294 [LW95b] verschiedenen Hamiltonkreise des Springers im Rösselsprung über das 8×8 Schachbrett, ja nicht einmal einen Hamiltonkreis im C_n garantiert.

Nichtsdestoweniger kann man sehr genau sagen, wann ein zufälliger Graph $G_{n,p} \in \mathcal{G}_{n,p}$ Hamiltonsch ist.

Satz 3.2.27 (Komlós, Szemerédi 1983) *Sei*

$$p \;=\; \frac{\log n + \log\log n + 2c_n}{n}.$$

Dann ist die Wahrscheinlichkeit für einen zufälligen Graphen aus $\mathcal{G}_{n,p}$, Hamiltonsch zu sein, asymptotisch gleich

$$\lim_{n\to\infty} \text{Prob}\,(G_{n,p} \textit{ ist Hamiltonsch}) \;=\; \begin{cases} 0 & \textit{falls } c_n \to \infty, \\ e^{-e^{-2c}} & \textit{falls } c_n \to c, \\ 1 & \textit{falls } c_n \to -\infty. \end{cases}$$

Erdős und Sachs [ES63] zeigten, daß es stets für zwei ganze Zahlen $r \ge 2$ und $g \ge 3$ einen r-regulären Graphen mit Taillenweite g gibt. Bezeichne $f(r,g)$ die minimale Ordnung eines solchen Graphen. Ein r-regulärer Graph mit Taillenweite g und nur $f(r,g)$ vielen Knoten heißt dann (r,g)-*Käfig* (engl. cage).

Übung 3.2.28 a) *Der* Petersen-*Graph ist der eindeutige $(3,5)$-Käfig.*
b) *Für $r \ge 2$ ist $f(r,4) = 2r$ und der Graph $K_{r,r}$ der eindeutige $(r,4)$-Käfig.*

Kapitel 4

Matching

4.1 Die Sätze von BERGE und GALLAI

Wir erinnern daran, daß ein Knoten v eine Kante e überdeckt und umgekehrt, falls $v \in e$. Zwei Kanten e und f in einem Graphen $G = (V, E)$ heißen *unabhängig*, falls sie nicht inzidieren: $e \cap f = \emptyset$. Eine Kantenmenge $M \subseteq E$ in G heißt unabhängig oder ein *Matching*[1], falls die Kanten in M paarweise unabhängig sind. Eine Kantenmenge M ist also genau dann ein Matching, wenn im Subgraphen (V, M) von G gilt: $d_{(V,M)}(v) \leq 1$ für alle $v \in V$. Beachte: eine Kantenmenge $M \subseteq E$ in einem Graphen G ist also unabhängig genau dann, wenn sie im Linegraphen $L(G)$ als Knotenmenge unabhängig ist. Ein Knoten heißt *frei* (bezüglich eines bestimmten Matchings), wenn er von keiner Matchingkante überdeckt wird, sonst *gematcht*. Die sogenannte *Matchingzahl* $\nu(G)$ eines Graphen G bezeichnet die Größe eines maximum (d.h. kardinalitätsmaximalen) Matchings in G. Ein Matching M heißt *perfekt* oder ein *1-Faktor* des Graphen, falls es alle Knoten des Graphen überdeckt, d.h. falls jeder Knoten $v \in V$ gematcht ist. Nur Graphen gerader Ordnung können mithin ein perfektes Matching besitzen, und ein Graph besitzt ein perfektes Matching genau dann, wenn $\nu(G) = n/2$. Ein Matching M mit $2|M| = n - 1$ heißt *fast-perfekt*.

Beispiele. Ein gerader Kreis hat genau zwei perfekte Matchings. Ein maximum Matching in C_{2k+1} hat k Kanten. Der Graph $K_{r,r}$ hat gerade $r!$ viele perfekte Matchings. Insbesondere bildet für jedes $j \in \{0, \ldots, r-1\}$ die Menge $M_j := \left\{\{u_i, v_{i+j(\mathrm{mod}p)}\} : i = 0, \ldots, r-1\right\}$ ein perfektes Matching. Wegen $\dot{\bigcup}_{j=0}^{r-1} M_j = E$ zerfällt die Kantenmenge des $K_{r,r}$ also disjunkt in 1-Faktoren, man sagt, der $K_{r,r}$ ist *1-faktorisierbar*. □

Übung 4.1.1 *Wieviele verschiedene perfekte Matchings hat* **a)** *der* K_{2p}, **b)** *ein Baum?* **c)** *Der* K_{2p} *ist 1-faktorisierbar.*[2] □

Greedy-Matching. Betrachten wir die naheliegende Greedy-Heuristik, um ein (inklusions-) maximales Matching M in einem Graphen G zu konstruieren. Hierbei gibt das Feld $Match : V \to V + 0$ den Matchingpartner eines Knotens $v \in V = \{1, \ldots, n\}$ an mit $Match[v] = 0$ genau dann, wenn v ungematcht ist.

[1] im Deutschen auch Zuordnung oder Korrespondenz

[2] Erheblich schwieriger zu zeigen ist, daß auch der vollständige uniforme Hypergraph $K_n^{(k)}$ (mit Kantenmenge $\binom{V}{k}$) 1-faktorisierbar ist, falls nur $k|n$ (BARANYAI 1975) (siehe z.B. [LW92a, Jun94]).

```
PROCEDURE MAXIMAL_MATCHING (G, M);

BEGIN
  H := G;    M := ∅;
  FOR v ∈ V DO  Match[v] := 0;
  WHILE E(H) ≠ ∅ DO BEGIN
    wähle eine Kante e = {u, v} ∈ E(H);
    M := M + e;
    Match[u] := v;  Match[v] := u;
    H := H − u − v;
  END;
END; {Maximal_Matching}
```

Da jedesmal, wenn eine Kante in M aufgenommen wird, ihre Endknoten sofort aus H entfernt werden, ist M ein Matching. Da durch die Anweisung $H := H - u - v$ nur Kanten aus H entfernt werden, die mit einer Matchingkante inzidieren, ist das konstruierte Matching zudem maximal. Offenbar kann MAXIMAL_MATCHING mit linearer Laufzeit implementiert werden. Wie gut ist ein von MAXIMAL_MATCHING konstruiertes Matching?[3]

Proposition 4.1.2 *Sei G ein Graph und M ein maximales Matching in G. Dann gilt $|M| \geq \nu(G)/2$.*

Beweis. Da M maximal ist, bildet die Menge C der von M überdeckten Knoten eine Knotenüberdeckung von G, d.h. C enthält von jeder Kante von G mindestens einen Endknoten. Sei M^* ein maximum Matching in G. Dann gilt folglich $2|M| = |C| \geq |M^*| = \nu(G)$. □

Wie der Pfad P_4 zeigt, ist diese Schranke i.a. schon bestmöglich. Bei hohem Minimalgrad sollte es leicht sein, ein „großes“ Matching zu konstruieren.

Proposition 4.1.3 *In einem Graphen G mit $\delta(G) \geq 2k$, $k \in \mathbb{N}$, findet der Algorithmus* MAXIMAL_MATCHING *ein Matching M der Kardinalität mindestens k.*

Beweis. Wegen $\delta(G) \geq 2k$ hat G mindestens $2k+1$ viele Knoten. Falls das produzierte Matching M nicht perfekt ist und beispielsweise einen Knoten v nicht überdeckt, so müssen alle $d(v) \geq 2k$ Nachbarn von v von M überdeckt sein, da MAXIMAL_MATCHING sonst die entsprechende Kante zu M hinzugefügt hätte. □

Durch einfaches Nachbessern der Lösung, die MAXIMAL_MATCHING liefert, erhält man eine erheblich schärfere Schranke für $\nu(G)$ in Abhängigkeit vom Minimalgrad.

Proposition 4.1.4 *Sei $G = (V, E)$ ein Graph. Es gelte $d(x) + d(y) \geq 2k$ für alle nicht benachbarten Knoten x und y in G. Dann gilt $\nu(G) \geq k$. Insbesondere gilt für jeden Graphen $\nu(G) \geq \delta(G)$.*

Beweis. Sei M ein maximales Matching in G, wie es beispielsweise der Algorithmus MAXIMAL_MATCHING konstruiert, und bezeichne $U \subseteq V$ die Menge der von M überdeckten Knoten, $|U| = 2|M|$. Seien weiter x und y zwei bezüglich M freie Knoten in G. Da M

[3] In [ADFS95] wird gezeigt, daß der Erwartungswert für $|M|$ größer ist, wenn man, statt zufällig gleichverteilt eine Kante $e \in E(H)$ auszuwählen und dem Matching hinzuzufügen, zunächst zufällig einen Knoten v in H auswählt und dann zufällig eine mit v inzidierende Kante (bzw., falls v isolierter Knoten in H ist, v lediglich aus H entfernt). Für dünne Graphen scheint es zudem günstiger zu sein, v unter den Knoten minimalen Grades in H auszuwählen [Tin84, FRS95].

maximal ist, sind x und y nicht benachbart, und es gilt $\Gamma(x) \cup \Gamma(y) \subseteq U$. Falls nun aber $|M| < k$, so folgt

$$d(x) + d(y) \;\geq\; 2k > 2|M|,$$

so daß es eine Matchingkante $\{u, v\} \in M$ gibt, die durch mindestens drei Kanten mit x und y verbunden ist – o.B.d.A. seien darunter die Kanten $\{x, u\}$ und $\{v, y\}$. Dann ist aber

$$M' := \; M - \{u, v\} + \{x, u\} + \{v, y\}$$

ein größeres maximales Matching in G. Durch Iteration findet man schließlich ein Matching der Kardinalität $|M| \geq k$. □

Beachte, daß diese Schranke beispielsweise für Kreise beliebig schlecht wird. Der Algorithmus, wie er im letzten Beweis angedeutet wurde, vergrößert in jedem Schritt das vorliegende maximale Matching, indem er auf einem P_4, dessen Endknoten frei sind und dessen mittlere Kante eine Matchingkante ist, Matching- und Nichtmatchingkanten vertauscht. Auf einem P_6 findet er also nicht notwendig ein maximum Matching.

Übung 4.1.5 a) *Man konstruiere ein Beispiel, in dem das gefundene Matching nur Kardinalität $|M| = \delta(G) \leq 2/3\nu(G)$ hat.*
b) *Der beschriebene Algorithmus konstruiert in allen Graphen mit $m \geq kn$ (d.h. Durchschnittsgrad $\frac{1}{n}\sum_{v \in V} d(v) \geq 2k$) ein Matching der Kardinalität mindestens k.*

Augmentierende Pfade. Sei M ein Matching im Graphen G. Ein (bezüglich M) alternierender Pfad (d.h. mit Kanten abwechselnd aus M und $E \setminus M$) in G heißt *M-augmentierend*, falls sein Anfangs- und Endknoten ungematcht sind (d.h. nicht von M überdeckt werden). Offenbar besitzt ein solcher M-augmentierender Pfad gerade viele Knoten, hat also ungerade Länge und enthält genau eine Nicht-Matchingkante mehr als Matchingkanten. Die folgende Charaktersierung von maximum Matchings ist Grundlage aller Matching-Algorithmen.

Satz 4.1.6 (Berge 1957) *Ein Matching M in einem Graphen G ist maximum genau dann, wenn G keinen M-augmentierenden Pfad enthält.*

Beweis. Wir beweisen diesen Satz in der Form: Ein Matching M in einem Graphen G ist nicht maximum genau dann, wenn G einen M-augmentierenden Pfad enthält.
„⇐“: Wenn G einen M-augmentierenden Pfad p enthält, so könnten wir das maximum Matching M offenbar durch Vertauschen der gematchten mit den ungematchten Kanten entlang p um 1 vergrößern: die inneren Knoten von p sind (nun anders, aber) wiederum gematcht und zusätzlich nun auch die beiden Endknoten, Widerspruch.
„⇒“: Sei M^* ein maximum Matching in G. Betrachte den Subgraphen von G mit der Kantenmenge $M^* \Delta M = (M^* \setminus M) \cup (M \setminus M^*)$. Nach Definition von Matchings ist der Grad eines jeden Knotens in $(V, M^* \Delta M)$ höchstens zwei, d.h. $M^* \Delta M$ ist eine disjunkte Vereinigung von alternierenden Pfaden und alternierenden geraden Kreisen (alternierend heißt hierbei, daß die Kanten der Pfade bzw. Kreise jeweils abwechselnd aus M bzw. M^* sind). Die Kreise enthalten, weil sie gerade sind, genauso viele M- wie M^*-Kanten. Wegen $|M| < |M^*|$ gibt es also einen alternierenden Pfad, der mehr M^*- als M-Kanten enthält. □

Die Bedingung aus Satz 4.1.6 heißt auch Berge-*Bedingung*. Ein maximum Matching M^* in einem Graphen läßt sich also aus z.B. dem leeren Matching konstruieren, indem man

das Startmatching sukzessive entlang augmentierender Pfade vergrößert. Offensichtlich muß das Startmatching dabei höchstens $n/2$ mal augmentiert werden. Im nächsten Abschnitt werden wir sehen, wie man in bipartiten Graphen augmentierende Pfade effizient konstruiert.

Überdeckung und Unabhängigkeit. Die Matchingzahl $\nu(G)$ und die Größe $\tau(G)$ einer minimum Knotenüberdeckung eines Graphen G stehen in enger Beziehung zueinander.

Proposition 4.1.7 $$\nu(G) \;\leq\; \tau(G) \;\leq\; 2\nu(G).$$

Beweis. Jede Knotenüberdeckung C in G muß von jeder Kante eines maximum Matchings M^* in G mindestens einen Knoten enthalten, so daß $|C| \geq |M^*| = \nu(G)$.
Die Menge $2|M|$ von einem maximalen Matching M überdeckten Knoten ist andererseits schon eine Knotenüberdeckung, denn gäbe es eine Kante $\{u,v\} \in E$, so daß weder u noch v von M überdeckt würden, so wäre $M+\{u,v\}$ ein größeres Matching und M nicht maximal gewesen. Also gilt $\tau(G) \leq 2|M| \leq 2\nu(G)$. □

Wie Matchings und die vollständigen Graphen zeigen, lassen sich die beiden Abschätzungen i.a. nicht verbessern. Wir wollen nun Beziehungen zwischen $\nu(G)$, $\tau(G)$, $\alpha(G)$ und den folgenden beiden Graphenparametern herleiten.

Definition 4.1.8 *Für einen Graphen $G=(V,E)$ sind die* Kantenüberdeckungszahl $\rho(G)$ *und die* Cliquenpartitionszahl $\theta(G)$ *wie folgt definiert:*

$$\begin{aligned} \rho(G) &:= \min\{|N| : N \subseteq E \text{ überdeckt alle Knoten}\};\\ \theta(G) &:= \min\{k : V = \dot{\bigcup}_{i=1}^{k} V_i \text{ mit } G[V_i] \text{ ist vollständig für alle } i\}. \end{aligned}$$

Der nächstfolgende Satz beschreibt das Verhältnis zwischen den Begriffen der Unabhängigkeit und der Überdeckung – das eine Mal auf Knoten, das andere Mal auf Kanten bezogen.

Satz 4.1.9 (Gallai 1959) *Sei $G=(V,E)$ ein Graph und $n=|V|$. Dann gilt:*

$$(i) \qquad \tau(G)+\alpha(G) \;=\; n;$$

Hat G zudem keine isolierten Knoten, so gilt:

$$(ii) \qquad \rho(G)+\nu(G) \;=\; n.$$

Beweis. Ad (i): Eine Knotenmenge $C \subseteq V$ ist eine Knotenüberdeckung genau dann, wenn $S := V \setminus C$ unabhängig ist.
Ad (ii). „$\leq$“: Sei M ein maximum Matching in G. Wähle für jeden nicht von M überdeckten Knoten eine beliebige mit ihm inzidierende Kante und bilde daraus die Kantenmenge N. Dann ist $M \cup N$ eine Kantenüberdeckung von G, und es gilt:

$$\rho(G) \;\leq\; |M|+|N| \;=\; \nu(G)+(n-2\nu(G)) \;=\; n-\nu(G).$$

„$\geq$“: Sei N eine minimum Kantenüberdeckung von G. Dann ist (V,N) offenbar kreisfrei. Seien (V,N_i) die Komponenten von (V,N). Bilde ein Matching M in G durch Auswahl je einer Kante aus jeder Komponente (V,N_i). Die Beziehung $|N| = n - |M|$ zwischen der Anzahl Kanten und der Anzahl der Komponenten in einem Wald liefert dann $\rho(G)+\nu(G) \geq |N|+|M| = n$. □

Wir sehen insbesondere, daß eine Kantenüberdeckung, die wie in Teil (ii) „$\leq$“ aus einem maximum Matching gebildet wurde, schon optimal ist.

Ferner liegen in einer Cliquenüberdeckung der Knoten eines Graphen natürlich alle Knoten einer maximum stabilen Menge in verschiedenen Cliquen. Wir finden also die folgende Beziehung für Graphen ohne isolierte Knoten:

$$\alpha(G) \;\leq\; \theta(G) \;\leq\; \rho(G). \tag{4.1}$$

Übungen

Die Güte des Greedy-Algorithmus für gewichtsmaximales Matching wurde in [RT81] bestimmt.

Übung 4.1.10 *Auch die Variante des Algorithmus* MAXIMAL_MATCHING, *bei der stets eine Kante* $\{u,v\}$ *mit minimalem Wert* $d(u)+d(v)$ *dem Matching hinzufügt wird, während die Knoten* u *und* v *aus dem Graphen entfernt werden, hat asymptotisch keine bessere Güteratio, d.h. es existiert eine Graphenfamilie* $\{\mathcal{G}_k\}_{k\in\mathbb{N}}$, *so daß diese Variante jeweils angewandt auf* $\mathcal{G}_k$ *ein Matching* M_k *konstruiert mit* $|M_k| \searrow \nu(\mathcal{G}_k)/2$ *für* $k \to \infty$.

Übung 4.1.11 *Eine Kantenüberdeckung* N *ist minimal genau dann, wenn* N *keine Pfade oder Kreise der Länge* ≥ 3 *enthält.*

Der Satz 4.1.9 von GALLAI verallgemeinert auf monotone Eigenschaften in Mengensystemen. Sei S eine n-elementige Menge. Eine Eigenschaft $P : 2^S \to \{0,1\}$ heißt *monoton*, falls für alle $X \subseteq S$ gilt

$$P(X) = 1 \;\Rightarrow\; P(Y) = 1 \quad \forall Y \subseteq X.$$

Eine Menge $X \subseteq S$ heißt *P-Menge*, falls $P(X) = 1$, sonst *$\bar{P}$-Menge*. Eine *Transversale* einer Familie $\mathcal{F} \subseteq 2^S$ ist eine Menge $T \subseteq S$, so daß $|T \cap X| \geq 1$ für alle $X \in \mathcal{F}$.

Übung 4.1.12 [CHL88] **a)** *Sei* P *eine monotone Eigenschaft auf* 2^S. *Eine Menge* $X \subseteq S$ *ist eine maximale* P*-Menge genau dann, wenn* $S \setminus X$ *eine minimale Transversale aller* $\bar{P}$*-Mengen ist.*
b) *Eine Menge ist eine minimale* $\bar{P}$*-Transversale genau dann, wenn sie eine minimale Transversale aller minimalen* $\bar{P}$*-Mengen ist.* **c)** *Folgere den Satz 4.1.9 von* GALLAI.

Übung 4.1.13 [HK73] *Sei* M *ein Matching im Graphen* G. *Dann gibt es genau* $k := \nu(G) - |M|$ *viele knotendisjunkte* M*-augmentierende Pfade in* G, *von denen mindestens einer höchstens Länge* $\lfloor n/k \rfloor - 1$ *hat.*

Für dichte Graphen sowie Bäume gibt es einfache lineare Algorithmen für maximum Matching:

Übung 4.1.14 *Sei* G *ein Graph mit* $d(x)+d(y) \geq n-1$ *für alle nicht benachbarten Knoten* x *und* y. *Dann kann für* n *(un-) gerade ein (fast-) perfektes Matching in* G *in Zeit* $\mathcal{O}(n+m)$ *konstruiert werden.*

Übung 4.1.15 [Sav80] *Sei* $T = (V,E)$ *ein Baum und* $r \in V$ *(die Wurzel von* T*). Für alle* $v \in V - r$ *heiße der Nachbar von* v *auf dem eindeutigen* $v-r$*-Pfad in* T *der* Vater $f(v)$ *von* v. *Ein Matching* M *in* T *heiße* proper *genau dann, wenn es für alle bezüglich* M *freien Knoten* $v \in V - r$ *einen Knoten* $w \in V - r - v$ *gibt mit* $f(w) = f(v)$ *und* $\{w, f(v)\} \in M$.
a) *Jedes proper Matching in* T *ist ein maximum Matching in* T.
b) *Sei* $r = v_1, \ldots, v_n$ *die Reihenfolge, in der eine Breitensuche mit Startknoten* r *die Knoten von* G *besucht. Dann konstruiert der folgende Algorithmus ein maximum Matching in* T *in Zeit* $\mathcal{O}(n)$:

```
M := ∅;
FOR i := n DOWNTO 1 DO BEGIN
  IF ∃ ungematchter Knoten w ∈ Γ(v_i) \ {v_1, ..., v_i} THEN
    M := M + {v_i, w};
END;
```

4.2 Matching in bipartiten Graphen

Viele Matching-Probleme in den Anwendungen beziehen sich auf (kantengewichtete) bipartite Graphen $G = (U, V, E)$. Man betrachte beispielsweise das folgende Zuordnungsproblem: aus einer Gruppe von Bewerbern U sollen möglichst viele auf offene Stellen V vermittelt werden.

Wir betrachten zunächst zwei Algorithmen, um ein kardinalitätsmaximales Matching in einem bipartiten Graphen zu konstruieren. Beide Algorithmen liefern Strukturinformation, mit deren Hilfe wir dann leicht die Formel von ORE über die Größe eines kardinalitätsmaximalen Matchings in bipartiten Graphen ableiten können. Als Korollare erhalten wir die Sätze von HALL und KÖNIG. Ein weiterer Satz von KÖNIG ist sodann der Schlüssel, das Problem INDEPENDENT SET auf bipartiten Graphen in polynomieller Zeit zu lösen. Schließlich untersuchen wir, wie sich das gewichtete bipartite Matching-Problem auf das ungewichtete reduzieren läßt (die sogenannte Ungarische Methode).

4.2.1 Zwei Matching-Algorithmen

Nach Satz 4.1.6 von BERGE läßt sich ein maximum Matching in einem Graphen dadurch bestimmen, daß man – ausgehend vom leeren Matching – das bestehende Matching sukzessive entlang eines augmentierenden Pfades vergrößert, bis es keinen augmentierenden Pfad mehr gibt.

Ein solcher augmentierender Pfad läßt sich beispielsweise mit Hilfe eines Flußalgorithmus finden: betrachte das (ungewichtete) Netzwerk N, das aus einem bipartiten Graphen $G = (U, V, E)$ durch Hinzufügen zweier Knoten s und t entsteht, wobei s vollständig zu U und t vollständig zu V verbunden wird. Alle Kanten seien von s nach U, von U nach V bzw. von V nach t gerichtet. Dann entsprechen sich maximum $s-t$–Flüsse in N und kardinalitätsmaximale Matchings in G bijektiv.

Effizientere Algorithmen ergeben sich, wenn man die spezielle Struktur des Problems ausnutzt. Der Standardalgorithmus für bipartites Matching findet einen augmentierenden Pfad durch eine modifizierte Breitensuche. Da höchstens $n/2$ Augmentierungen nötig sind, die jeweils in Zeit $\mathcal{O}(n)$ bewerkstelligt werden können, hat dieser Algorithmus Laufzeit $\mathcal{O}(nm)$. Der Algorithmus von HOPCROFT und KARP [HK73] dagegen arbeitet in Phasen und erreicht durch quasi simultanes Augementieren mehrerer Pfade pro Phase eine Laufzeit von $\mathcal{O}(\sqrt{n}m)$.

Der Standardalgorithmus. Für den Standardalgorithmus würde es genügen, gegeben ein bipartiter Graph G und ein Matching M in G, *irgendeinen* M-augmentierenden Pfad in Zeit $\mathcal{O}(n+m)$ zu finden. Im Hinblick auf den Algorithmus von HOPCROFT und KARP zeigen wir etwas stärker:

Proposition 4.2.1 *Sei M ein Matching in einem bipartiten Graphen $G = (U, V, E)$. Dann läßt sich in Zeit $\mathcal{O}(n+m)$ die Länge $k \in \mathbb{N}$ eines kürzesten M-augmentierenden Pfades sowie die Menge aller augmentierender Pfade der Länge k bestimmen (oder entscheiden, daß es keinen M-augmentierenden Pfad mehr gibt).*

Beweis. Die Prozedur UNGARISCHER_DAG auf der übernächsten Seite bestimmt k und konstruiert einen sogenannten Ungarischen dag (für directed acyclic graph)[4], in Zeichen

[4] Die Bezeichnung „Ungarisch“ bezieht sich auf die beiden Ungaren KÖNIG und EGERVÁRY und wurde wohl von KUHN (1955, 1956) eingeführt.

$dag = (S \cup T, A)$, der alle kürzesten M-augmentierenden Pfade in G codiert. Dieser Graph dag ist also ein Graph, dessen Kanten so orientiert sind, daß es keine gerichteten Kreise gibt. Er ist wie folgt definiert. (Beachte: ein augmentierender Pfad in G hat, da er ungerade Länge hat und G bipartit ist, je einen Endknoten in U und einen in V; beide Endknoten sind frei.) Die Menge der freien Knoten in U werde mit $U\_Terminale$ bezeichnet; dies sind die im Teil U möglichen Anfangsknoten von augmentierenden Pfaden in G. Die Knoten $u \in U\_Terminale$ stellen die Quellen von dag dar, d.h. die Knoten, in die keine gerichtete Kante hineinführt, sondern nur Kanten herausführen. Alle mit Knoten aus $U\_Terminale$ inzidierenden Kanten in G gehören auch zu dag und werden so gerichtet, daß sie aus den Knoten $u \in U\_Terminale$ herausführen. Eine Kante $\{x, y\} \in E$ wird nun in dag aufgenommen und wie (x, y) gerichtet, wenn es von einem $u \in U\_Terminale$ einen gerichteten, bezüglich M alternierenden Pfad in dag nach x gibt und die Kante $\{x, y\}$ den Pfad alternierend bis nach y fortsetzt. S ist dann also die Menge der Knoten von G im Teil U (inklusive $U\_Terminale$), die auf bezüglich M alternierenden Pfaden von Knoten aus $U\_Terminale$ aus erreichbar sind. Entsprechendes gilt für die Knoten T im Teil V.

Jedem Knoten x in dag ist eine Zahl $level[x]$ zugeordnet, die der Länge eines kürzesten gerichteten (bezüglich M alternierenden) Pfades von einem Knoten $u \in U\_Terminale$ nach x entspricht. Die Knoten $u \in U\_Terminale$ befinden sich also auf Level 0. Alle Knoten mit geradem Level liegen im Teil $S \subseteq U$, alle Knoten mit ungeradem im Teil $T \subseteq V$ und alle (gerichteten) Kanten verlaufen zwischen Knoten auf einem Level ℓ zu Knoten auf dem nächsthöheren Level $\ell + 1$.

Interessant sind nun die Senken des dag im Teil T, d.h. die Knoten $t \in T$, in die zwar eine Kante hineinführt, aber keine mehr hinaus: da G bipartit ist, ist die hineinführende Kante offenbar eine ungematchte Kante, und wenn keine Kante aus t hinausführt, so heißt das, daß t ungematcht ist: also endet in t ein augmentierender Pfad. Offensichtlich ist k das Minimum über alle Werte $level[t]$ mit $t \in T$ freier Endpunkt eines alternierenden, gerichteten Pfades in dag. Die Menge der freien Knoten $t \in T$ mit $level[t] = k$ sei mit $V\_Terminale$ bezeichnet – sie stellen die in V möglichen Endpunkte kürzester M-augmentierender Pfade in G dar. Um später über solche Pfade augmentieren zu können, ist es nötig, den Digraphen dag so zu codieren, daß man die zu einem Knoten $t \in V\_Terminale$ gehörigen augmentierenden Pfade zurückverfolgen kann. Dazu speichern wir zu jedem Knoten $t \in T$ die Menge der Knoten $s \in S$ auf Level $level[t] - 1$, über die man t auf einem kürzesten augmentierenden Pfad erreichen kann, in einer Liste $Vor[t]$ der „Vorgänger“ von t in dag. Der Vorgänger eines Knotens $s \in S$ mit $level[s] > 0$ ist hingegen durch das Matching eindeutig festgelegt: es ist der Matchingpartner $Match[s]$ von s.

Die Knotenmengen S und T, die Liste Vor, der Wert k sowie die Menge $V\_Terminale$ werden nun durch eine Breitensuche, die mit der Menge $U\_Terminale$ auf Level 0 beginnt, bestimmt. Da die Nachfolger von Knoten im Teil V durch die Matchingkanten eindeutig bestimmt sind, genügt es, während der Breitensuche nur Knoten des dag's aus dem Teil U zu queuen. Darüberhinaus genügt es, da wir nur an kürzesten augmentierenden Pfaden interessiert sind, Knoten $s \in S$ mit $level[s] < k$ zu queuen, denn sonst hätte jeder augmentierende Pfad, der s benutzt, Länge größer als k.

Offensichtlich hat die Prozedur UNGARISCHER_DAG lineare Laufzeit. □

```
PROCEDURE Ungarischer_dag;

BEGIN
  {alle Knoten v ∈ V als unbesucht markieren:}
  FOR v ∈ V DO level[v] := ∞;
  k := ∞;   S := ∅;   T := ∅;
  InitQueue (Q);
  FOR u ∈ U DO IF Match[u] = 0 THEN BEGIN
    S := S + u;
    level[u] := 0;
    Insert (Q, u);
  END;
  {nun gilt S = U_Terminale}
  WHILE Q ≠ ∅ DO BEGIN
    {alle Knoten aus Q haben level < k}
    s := ExtractHead (Q);
    FOR t ∈ Γ(s) DO IF level[t] ≥ level[s] + 1 THEN BEGIN
      {es gilt nun insbesondere {s,t} ∉ M; die Kante {s,t}}
      {verlängert den alternierenden Pfad nach s bis nach t}
      IF level[t] = ∞ THEN BEGIN
        {wir besuchen t erstmalig}
        T := T + t;
        level[t] := level[s] + 1; {   ≤ k}
        InitQueue (Vor[t]);
        Insert (Vor[t], s);
        IF Match[t] = 0 THEN BEGIN
          {in t endet ein kürzester M-augmentierender Pfad}
          V_Terminale := V_Terminale + t;
          IF k = ∞ THEN k := level[t];
        END
        ELSE IF level[t] < k THEN BEGIN
          {die t überdeckende Matchingkante verlängert den}
          {bez. M alternierenden Pfad nach t bis nach Match[t]}
          S := S + Match[t];
          level[Match[t]] := level[t] + 1;
          Insert (Q, Match[t]);
        END; {if Match[t]}
      END
      ELSE BEGIN
        {es ist bereits t ∈ T und damit level[t] = level[s] + 1}
        Insert (Vor[t], s);
      END; {if level[t]}
    END; {for}
  END; {while}
END; {Ungarischer_dag}
```

Der Algorithmus von HOPCROFT und KARP. Mit nur wenig mehr Aufwand gewinnt man einen $\mathcal{O}(\sqrt{n}m)$-Algorithmus. Der Algorithmus von HOPCROFT und KARP arbeitet in Phasen, während derer das bestehende Matching jeweils entlang mehrerer disjunkter augmentierender Pfade „gleichzeitig“ vergrößert wird. Es bezeichne M das bereits konstruierte Matching (zu Beginn also $M = \emptyset$). Genauer wird in jeder Phase eine maximale Menge $\mathcal{M}$ knotendisjunkter, kürzester M-augmentierender Pfade bestimmt und das bestehende Matching M entlang der Pfade aus $\mathcal{M}$ augmentiert. Die erste Phase bestimmt also schon ein maximales Matching in dem Graphen G. Es zeigt sich, daß einerseits jede Phase nur lineare Zeit erfordert und daß es andererseits nach maximal $\frac{3}{2}\sqrt{n}$ vielen Phasen keine M-augmentierende Pfade mehr gibt, d.h. daß M maximum ist.

Proposition 4.2.2 *Sei M ein Matching in einem bipartiten Graphen $G = (U, V, E)$. Dann läßt sich, gegeben ein Ungarischer dag, die Länge $k \in \mathbb{N}$ eines kürzesten M-augmentierenden Pfades und die Menge der $V\_Terminale$, in Zeit $\mathcal{O}(n + m)$ über eine maximale Menge $\mathcal{M}$ knotendisjunkter M-augmentierender Pfade in G der Länge k augmentieren.*

Beweis. Betrachte die Prozedur AUGMENT_SIMULTAN. Die Idee ist, für jeden Knoten $v \in V\_Terminale$ Tiefensuchen-artig nach einem (durch die Vor- und $Match$-Felder codierten) Pfad zu einem Knoten $u \in U\_Terminale$ zu suchen. Alle Knoten aus U, denen man dabei begegnet, werden als „tot“ markiert: sie gehören entweder zum augmentierenden Pfad (so man einen findet) und dürfen somit von keinem anderen augmentierenden Pfad mehr benutzt werden, oder sie führen in eine Sackgasse, d.h. es gibt von ihnen aus keinen zu den bereits konstruierten augmentierenden Pfaden disjunkten, alternierenden Pfad nach $U\_Terminale$ mehr. Alle Knoten $u \in U$ seien zu Anfang mit $tot[u] = false$ markiert.

Da die augmentierenden Pfade disjunkt sind, sind die mit den Augmentationen verbundenen Operationen sicherlich linear in der Anzahl der Kanten des Ungarischen *dag* und damit in $n + m$. Durch die Zeile, in der der jeweils nächstmögliche Vorgänger von $Pfad[\ell]$ vermerkt wird, wird sichergestellt, daß jede Kante des dag nur einmal abgelaufen wird (exklusive der Augmentierungen). Die Anzahl der Operationen ist daher insgesamt $\mathcal{O}(|U| + |E(dag)|) = \mathcal{O}(n + m)$. □

```
PROCEDURE Augment_Simultan;

BEGIN
  FOR v ∈ V_Terminale DO BEGIN
    {suche in v endenden augmentierenden Pfad (Pfad[k],...,Pfad[0])}
    Pfad[k] := v;
    ℓ := k;
    v_fertig := false;
    REPEAT
      {Pfad[ℓ] ∈ V}
      {suche einen noch unbenutzten Vorgänger von Pfad[ℓ] im dag}
      VorPtr := Vor[Pfad[ℓ]];
      WHILE VorPtr ≠ Nil AND tot[VorPtr ↑ .Knoten] DO
        VorPtr := VorPtr ↑ .Next;
      IF VorPtr = Nil THEN BEGIN
        IF ℓ = k THEN BEGIN
          {es gibt keinen augmentierenden Pfad mehr nach v}
          v_fertig := true;
        END
        ELSE BEGIN
          {kein augmentierender Pfad durch Pfad[ℓ] und damit
            durch Match[Pfad[ℓ]] = Pfad[ℓ + 1] mehr möglich}
          tot[Pfad[ℓ + 1]] := true;
          ℓ := ℓ + 2;
        END; {if ℓ}
      END
      ELSE BEGIN
        {nächstmöglichen Vorgänger von Pfad[ℓ] vermerken}
        Vor[Pfad[ℓ]] := VorPtr ↑ .Next;
        {Pfad über VorPtr ↑ .Knoten verlängern}
        Pfad[ℓ − 1] := VorPtr ↑ .Knoten;
        tot[Pfad[ℓ − 1]] := true;
        ℓ := ℓ − 1;
        {Pfad[ℓ] ∈ U}
        IF ℓ = 0 THEN BEGIN
          augmentiere entlang (Pfad[k],...,Pfad[0]);
          v_fertig := true;
        END
        ELSE BEGIN
          {nächste Pfadkante ist eindeutig bestimmt}
          Pfad[ℓ − 1] := Match[Pfad[ℓ]];
          ℓ := ℓ − 1;
        END; {if ℓ}
      END; {if VorPtr}
    UNTIL v_fertig;
  END; {for}
END; {Augment_Simultan}
```

Der Beweis, daß höchstens $\frac{3}{2}\sqrt{n}$ Phasen notwendig sind, benutzt die folgenden zwei Hilfsaussagen.

Lemma 4.2.3 *Sei P ein M-augmentierender Pfad minimaler Länge, $M' := M \Delta P$ das Matching, das man aus M durch Augmentieren entlang P erhält, und P' ein M'-augmentierender Pfad. Dann gilt*

$$\ell(P') \;\geq\; \ell(P) + 2|E(P) \cap E(P')|, \tag{4.2}$$

wobei $\ell(P)$ bzw. $\ell(P')$ jeweils die Länge des Pfades P bzw. P' bezeichnet.

Beweis. Zunächst beobachtet man:

(i) Die Endpunkte von P' sind nicht in P enthalten, da nach Augmentierung von P alle Knoten in P von M' überdeckt werden.

(ii) Alle Kanten in $E(P') \setminus E(P)$ sind in M enthalten genau dann, wenn sie in M' enthalten sind, da die Augmentierung entlang P nur die Kanten von P betrifft.

(iii) P und P' sind knotendisjunkt genau dann, wenn sie kantendisjunkt sind.

Falls die beiden Pfade P und P' nun kantendisjunkt sind, so ist P' auch ein augmentierender Pfad für M und (4.2) gilt, da P ein M-augmentierender Pfad minimaler Länge ist. Andernfalls betrachte man einen Knoten, in dem der Pfad P' auf den Pfad P trifft oder ihn verläßt. Nach der Augmentierung entlang P wird jeder Knoten in P von einer Matchingkante aus $M' \cap E(P)$ überdeckt. Da mit jedem Knoten in G nur höchstens eine Kante aus M' inzidiert, ist die erste (letzte) gemeinsame Kante von P und P' stets eine M'-Kante. Mithin ist das Anfangsstück von P' bis zum ersten gemeinsamen Knoten mit P (der möglicherweise ein Endknoten von P ist) gegebenenfalls verlängert bis zum entsprechenden Endknoten von P ein M-augmentierender Pfad in G. Entsprechendes gilt am anderen Ende von P'. Da $E(P)\Delta E(P')$ genau die vier Endknoten von P und P' als bezüglich M freie Knoten enthält, besteht $E(P)\Delta E(P')$ also aus diesen beiden M-augmentierenden Pfaden und darüberhinaus möglicherweise aus einigen disjunkten Kreisen. Nach Wahl von P sind diese beiden M-augmentierenden Pfade mindestens so lang wie P, so daß gilt: $|E(P)\Delta E(P')| \geq 2|E(P)| = 2\ell(P)$. Mithin gilt

$$\ell(P) + \ell(P') \;=\; |E(P)\Delta E(P')| + 2|E(P) \cap E(P')| \;\geq\; 2\ell(P) + 2|E(P) \cap E(P')|,$$

woraus (4.2) auch in diesem Fall folgt. □

Lemma 4.2.4 *Nach jeder Phase vergrößert sich die Länge eines kürzesten augmentierenden Pfades um mindestens 2.*

Beweis. In einer Phase wird das aktuelle Matching M entlang einer maximalen Menge $\mathcal{M}$ von knotendisjunkten, kürzesten M-augmentierenden Pfaden augmentiert, und man erhält ein Matching M'. Sei P' ein M'-augmentierender Pfad. Falls P' zu allen Pfaden aus $\mathcal{M}$ disjunkt ist, so muß P' aufgrund der Maximalität von $\mathcal{M}$ länger sein als die M-augmentierenden Pfade in $\mathcal{M}$, und da augmentierende Pfade stets ungerade Länge haben, überschreitet die Länge von P' die der Pfade in $\mathcal{M}$ sogar um mindestens 2. Falls P' andernfalls einen Pfad $P \in \mathcal{M}$ schneidet, so hat P' auch mindestens eine Kante mit P gemeinsam, denn jeder Knoten in P wird von (nur) einer Matchingkante aus $M' \cap E(P)$ überdeckt. Nach Gleichung (4.2) überschreitet auch in diesem Fall die Länge von P' die von P um mindestens 2. □

Proposition 4.2.5 *Wenn ein Matching M in einem bipartiten Graphen G $\frac{3}{2}\sqrt{n}$-mal sukzessive entlang einer maximalen Menge knotendisjunkter, kürzester M-augmentierender Pfade augmentiert wird, dann ist es maximum.*

Beweis. Sei M^* ein maximum Matching in G und M ein Matching, daß man nach $\frac{1}{2}\sqrt{n}$ vielen Phase erhalten hat. Nach Lemma 4.2.4 hat nun jeder M-augmentierende Pfad Länge mindestens $\sqrt{n}+1$. Nach Übung 4.1.13 enthält $M^*\Delta M$ genau $|M^*|-|M| = \nu(G)-|M|$ viele knotendisjunkte M-augmentierende Pfade. Da jeder dieser Pfade aber mindestens $\sqrt{n}+2 \geq \sqrt{n}$ viele Knoten enthält, folgt

$$n \;\geq\; (\text{Anzahl der Pfade}) \cdot (\text{min. Anzahl der Knoten eines Pfades}) \geq\; (\nu(G)-|M|)\cdot\sqrt{n}$$

und daraus $\nu(G)-|M| \leq \sqrt{n}$. Da jedes Augmentieren von M entlang eines augmentierenden Pfades $|M|$ um mindestens eins erhöht, kann M also nur noch höchstens $\sqrt{n}$-mal augmentiert werden. Also gibt es auch nur noch höchstens $\sqrt{n}$ viele weitere Phasen. □

Wir fassen zusammen:

Satz 4.2.6 (HOPCROFT, KARP 1973) *Der Algorithmus von* HOPCROFT *und* KARP *konstruiert ein maximum Matching in einem bipartiten Graphen $G = (U,V,E)$ in Zeit $\mathcal{O}(\sqrt{n}m)$.*

Übung und Anmerkungen

Übung 4.2.7 *Man konstruiere einen Fall, wo der Algorithmus von* HOPCROFT *und* KARP *tatsächlich $\Omega(\sqrt{n})$ viele Phasen benötigt.*

Bemerkenswerterweise hat der Algorithmus von HOPCROFT und KARP *erwartete* Laufzeit $\mathcal{O}(m)$, da die augmentierenden Pfade „in der Regel“ sehr kurz sind [Mot94]. Eine Verbesserung des HOPCROFT-KARP-Algorithmus hat auf eine (worst-case-) Laufzeit von $\mathcal{O}(\sqrt{n}m/\log n)$ [FM95].
Der HOPCROFT-KARP-Algorithmus läßt sich interpretieren als der Flußalgorithmus von DINIC [Din70] angewandt auf das Netzwerk, das man aus dem bipartiten Graphen $G = (U \dot\cup V, E)$ erhält, wenn man zwei Knoten s und t hinzufügt und s vollständig mit U sowie t vollständig mit V verbindet [ET75]. Einen weiteren $\mathcal{O}(\sqrt{n}m)$-Algorithmus für das bipartite Matchingproblem findet man bei [BG91, GL95b].

4.2.2 Die Sätze von ORE, HALL und KÖNIG

Aus dem obigen bipartiten Matching-Algorithmus ergibt sich ein einfacher Beweis für die folgende Charakterisierung der Matchingzahl eines bipartiten Graphen. Für $S \subseteq U$ sei $\Gamma(S) := \bigcup_{s\in S}\Gamma(s) \setminus S$.

Satz 4.2.8 (ORE 1955) *Für einen bipartiten Graphen $G = (U,V,E)$ gilt:*

$$\nu(G) \;=\; |U| - d,$$

wobei $d := \max_{S\subseteq U}\{|S| - |\Gamma(S)|\}$.

Beweis. „$\leq$“ ist trivial, da schon in einer Knotenmenge $S \subseteq U$ mit $|S|-|\Gamma(S)| = d$, also $|\Gamma(S)| = |S| - d$, mindestens d Knoten nicht gematcht werden können.
Für „$\geq$“ beweisen wir, daß der oben vorgestellte Algorithmus zu einem bipartiten Graphen $G = (U,V,E)$ ein Matching M der Größe $|M| \geq |U| - d$ konstruiert. Der Algorithmus

terminiert, wenn in der Prozedur UNGARISCHER_DAG kein augmentierender Pfad mehr gefunden wird. Betrachten wir die Mengen S und T am Ende dieses Durchlaufs der Prozedur UNGARISCHER_DAG. Einerseits, da kein augmentierender Pfad gefunden wurde, hat k noch immer den Wert ∞ und die `WHILE`-Schleife hat die Warteschlange Q vollständig abgearbeitet. Also gilt $\Gamma(S) \subseteq T$. Folglich sind alle gematchten Knoten aus S nach T gematcht. In T sind jedoch alle Knoten gematcht (sonst gäbe es einen augmentierende Pfad), und nach Definition von S sind sie alle nach S gematcht. Somit gilt:

M matcht die Knoten aus T genau mit den gematchten Knoten aus S.

Da S alle freien Knoten aus U enthält, gilt also

$$|\{u \in U : \ u \text{ frei}\}| = |S| - |\{s \in S : \ s \text{ gematcht}\}| = |S| - |T|.$$

Andererseits gilt, da nach Konstruktion von T trivialerweise stets $T \subseteq \Gamma(S)$ ist, sogar die Gleichheit $\Gamma(S) = T$. Insgesamt erhalten wir

$$\begin{array}{rcccccl} |U| & = & |\{u \in U : \ u \text{ gematcht}\}| & + & |\{u \in U : \ u \text{ frei}\}| & & \\ & = & |M| & + & |S| - |T| & = & |M| + |S| - |\Gamma(S)| \end{array}$$

und mithin

$$|M| \ = \ |U| - (|S| - |\Gamma(S)|) \ \geq \ |U| - d. \qquad \square$$

Als Spezialfall des Satzes von ORE erhalten wir als nächstes den sogenannte „Heiratssatz". Dieser Name rührt von folgender Veranschaulichung her: ein Heiratsinstitut kann eine Menge U heiratswilliger Damen, die aus einer Menge V von Herren mögliche Partner ausgewählt haben, stets mit Herren ihrer Wahl verheiraten, wenn sich nur keine Gruppe von $k \leq |U|$ Damen auf zusammengenommen weniger als k Herren festgelegt hat.

Satz 4.2.9 (HALL 1935) *Sei $G = (U, V, E)$ ein bipartiter Graph und o.B.d.A. $|U| \leq |V|$. Dann existiert in G ein Matching, welches ganz U überdeckt genau dann, wenn G die sogenannte* HALL-Bedingung *erfüllt:*

$$|\Gamma(X)| \ \geq \ |X| \qquad \forall X \subseteq U \tag{4.3}$$

Aus ersichtlichen Gründen trägt der Satz 4.2.8 von ORE auch den Namen „Defektform des Satzes von HALL". Als einfache Folgerung finden wir:

Korollar 4.2.10 (FROBENIUS 1917) *Ein bipartiter Graph besitzt ein perfektes Matching genau dann, wenn seine Teile gleich groß sind und einer der Teile die* HALL*-Bedingung erfüllt.* $\square$

Als Anwendungen des Satzes von HALL wollen wir die eingangs gemachte Beobachtung, daß der $K_{n,n}$ 1-faktorisierbar ist, zweimal verallgemeinern.

Korollar 4.2.11 (KÖNIG 1916) *Ein r-regulärer bipartiter Graph G ist 1-faktorisierbar.*

Beweis. Wir induzieren nach r. Ein 1-regulärer Graph ist ein perfektes Matching. Sei also $r > 1$. Aus $r|U| = m = r|V|$ folgt zunächst $|U| = |V|$. Genauso, da die Nachbarschaft $\Gamma(X)$ einer Menge $X \subseteq U$ alle Kanten aus X aufnehmen muß: $r|\Gamma(X)| \geq r|X|$. Nach Satz 4.2.9 besitzt G also ein perfektes Matching. Entfernt man dieses, verbleibt ein $(r-1)-$regulärer Graph, der nach Induktionsannahme in 1-Faktoren zerfällt. $\square$

Übung 4.2.12 **a)** *Man gebe für jedes $r \geq 2$ einen r-regulären Graphen ohne perfektes Matching an.* **b)** *Man gebe einen bipartiten Graphen $G = (V, E)$ an, der nicht Subgraph eines $\Delta(G)$-regulären, bipartiten Graphen H auf derselben Knotenmenge V ist.* □

Betrachten wir nun nicht notwendig reguläre Graphen.

Korollar 4.2.13 (MENDELSOHN, DULMAGE 1958) *Jeder bipartite Graph enthält ein Matching, das alle Knoten maximalen Grades überdeckt.*

Beweis. Sei $G = (V_1, V_2, E)$ ein bipartiter Graph mit Teilen V_1 und V_2. Nach Satz 4.2.9 gibt es in G ein Matching M_1 (das u.U. leer ist), das alle Knoten vom Grad $\Delta(G)$ in V_1 überdeckt (betrachte den Subgraphen von G, der von den Kanten gebildet wird, die mit Knoten $u \in U$ vom Grad $d(u) = \Delta(G)$ inzidieren). Analog gibt es ein Matching M_2 in G, das alle Knoten vom Grad $\Delta(G)$ in V_2 überdeckt.

Setze $M := M_1 \cap M_2$. Der Subgraph $M_1 \Delta M_2$ von G besteht aus einer Menge disjunkter alternierender Pfade und (gerader) Kreise. Für jeden geraden Kreis $C \subseteq M_1 \Delta M_2$ füge eines der beiden Matchings $M_i \cap C$, $i \in \{1, 2\}$ beliebig, zu M hinzu. Wie man sich leicht überzeugt, hat jeder Pfad in $M_1 \Delta M_2$ genau einen Endknoten vom Grad $\Delta(G)$ in G. Man füge für jeden solchen Pfad P (aufgefaßt als Kantenmenge) dasjenige Matching $M_i \cap P$, $i \in \{1, 2\}$, zu M hinzu, das den Endknoten von P vom Grad $\Delta(G)$ überdeckt. Dann überdeckt M schließlich alle Knoten vom Grad $\Delta(G)$ in G. □

Jede Zerlegung der Kanten eines Graphen in Matchings besteht aus mindestens $\Delta(G)$ vielen Matchings, da die mit einem Knoten maximalen Grades inzidierenden Kanten offenbar in paarweise verschiedenen Matchings liegen müssen. Während es i.a. NP-schwer ist, die Kanten eines Graphen in minimal viele Matchings zu partitionieren (siehe Satz 5.6.12), folgt für bipartite Graphen aus Korollar 4.2.13 die folgende Max-Min-Aussage.

Satz 4.2.14 (KÖNIG 1916)
Die Kanten eines bipartiten Graphen G sind die Vereinigung von $\Delta(G)$ Matchings. □

Satz 5.6.4 gibt einen $\mathcal{O}(nm)$-Algorithmus für dieses Problem. Für allgemeine Graphen existiert ein $\mathcal{O}(nm)$-Algorithmus, der die Kanten eines jeden Graphen in immerhin höchstens $\Delta(G) + 1$ viele Matchings partitioniert, siehe Korollar 5.6.6.

4.2.3 INDEPENDENT SET in bipartiten Graphen

Ein weiteres klassisches Resultat von KÖNIG besagt, daß für bipartite Graphen Gleichheit in der linken Ungleichung von Proposition 4.1.7 gilt.

Satz 4.2.15 (KÖNIG 1931) *Für einen bipartiten Graphen G gilt:*

$$\tau(G) \;=\; \nu(G).$$

Beweis [FF56].
„$\geq$“: Sei $G = (U, V, E)$ ein bipartiter Graph und M ein Matching sowie C eine Knotenüberdeckung in G. Da C insbesondere die Kanten von M überdecken muß, gilt $|C| \geq |M|$. Folglich ist

$$\tau(G) \;=\; \min_{C} \{|C|\} \;\geq\; \max_{M} \{|M|\} \;=\; \nu(G).$$

„$\leq$“: Wir konstruieren zu einem maximum Matching M^* in G eine Knotenüberdeckung C^* mit $|C^*| \leq |M^*|$. Dann folgt wie gewünscht $\tau(G) \leq |C^*| \leq |M^*| = \nu(G)$. Insbesondere ist C^* eine minimum Knotenüberdeckung.
Sei $U\_Terminale \subseteq U$ die Menge der freien, d.h. von M^* nicht überdeckten Knoten im Teil U, und S bzw. T die Menge der von $U\_Terminale$ aus auf bez. M^* alternierenden Pfaden erreichbaren Knoten in U bzw. V, wie sie die Prozedur UNGARISCHER_DAG konstruiert. Beachte: da es keine M^*-augmentierenden Pfade gibt, wird die Warteschlange Q in der `WHILE`-Schleife vollständig abgearbeitet, und es gilt $\Gamma(S) \subseteq T$ (vergleiche den Beweis zu Satz 4.2.8).
Behauptung 1: $C^* := U \setminus S \cup T$ *ist eine Knotenüberdeckung von* G.
Sei $\{u, v\} \in E$. Falls nicht schon u in der Überdeckung C^* liegt, d.h. falls $u \in S$, so liegt wegen $\Gamma(S) \subseteq T$ der Knoten v in T und damit in der Überdeckung C^*.
Behauptung 2: Alle Knoten aus C^* *sind gematcht.*
Wegen $U\_Terminale \subseteq S$ sind alle Knoten aus $U \setminus S$ gematcht. Da es keine M^*-augmentierenden Pfade gibt, sind auch alle Knoten in T gematcht.
Behauptung 3: $|C^*| \leq |M^*|$.
Der Matchingpartner eines Knotens $t \in T$ liegt nach Konstruktion des Ungarischen dag in S. Also gibt es keine Matchingkante zwischen $U \setminus S$ und T, d.h. jeder Knoten der Überdeckung C^* wird von einer anderen Matchingkante überdeckt. □

Für einen bipartiten Graphen $G = (U, V, E)$ gilt offensichtlich $\alpha(G) \geq \max\{|U|, |V|\}$.

Übung 4.2.16 *Man finde den kleinsten zusammenhängenden bipartiten Graphen* $G = (U, V, E)$ *mit* $\alpha(G) > \max\{|U|, |V|\}$.

Aus dem Beweis von Satz 4.2.15 leitet sich unmittelbar ein linearer Algorithmus ab, der aus einem maximum Matching in einem bipartiten Graphen eine minimum Knotenüberdeckung und eine maximum stabile Menge konstruiert. Während diese Optimierungsprobleme also für allgemeine Graphen NP-schwer sind (vgl. Satz 1.4.10), folgt mit Satz 4.2.6:

Korollar 4.2.17 *Für bipartite Graphen besitzen die Optimierungsprobleme* NODE COVER *und* INDEPENDENT SET $\mathcal{O}(\sqrt{n}m)$*-Algorithmen.* □

Dies kontrastiert mit dem folgenden NP-Vollständigkeitsresultat.

Proposition 4.2.18 (POLJAK 1974) *Sei* $p \geq 3$ *fest. Dann ist das Problem* INDEPENDENT SET, *eingeschränkt auf die Menge aller* p*-partiten Graphen,* NP*-vollständig.*

Beweis. Da ein p-partiter Graph auch $(p+1)-$partit ist, genügt es, den Fall $p = 3$ zu betrachten. Wir reduzieren von INDEPENDENT SET. Sei G ein Graph mit n Knoten und m Kanten. Wir konstruieren aus G einen 3-partiten Graphen H, indem wir auf jeder Kante zwei Knoten einfügen oder, äquivalent, indem wir jede Kante in G durch einen Pfad der Länge drei ersetzen. H hat also $n+2m$ Knoten und $3m$ Kanten und kann in polynomieller Zeit konstruiert werden. Es gilt $\alpha(H) = \alpha(G) + m$ (Übung). □

Desweiteren ergibt sich aus dem Satz 4.2.15 mit Hilfe des Satzes 4.1.9 von GALLAI:

Korollar 4.2.19 (KÖNIG 1932) *Für einen bipartiten Graphen* G *ohne isolierte Knoten gilt Gleichheit in (4.1):* $\rho(G) = \alpha(G)$. □

4.2.4 Gewichtetes bipartites Matching

Das Matching-Problem wird erheblich schwerer, wenn man Gewichte $w : E \to \mathbb{Q}^+$ auf den Kanten des bipartiten Graphen $G = (U \dot\cup V, E)$ einführt und nun ein Matching M maximalen Gewichts $w(M) = \sum_{e \in M} w(e)$ in G sucht. Das gewichtete (bipartite) Matching-Problem läßt sich jedoch mit Hilfe der primal-dual-Methode aus der Linearen Optimierung auf das einfachere Kardinalitäts-Matching-Problem zurückführen. Auf diesen Hintergrund können wir hier allerdings nicht eingehen (siehe z.B. [Jun94]); wir nutzen lediglich die Technik, gleichzeitig zu einer primalen Lösung auch eine duale Lösung zu konstruieren, die eine obere Schranke für das Gewicht eines maximum Matchings liefert. Unsere Darstellung orientiert sich an [Law76a, Gal86].

Eine *(zulässige) Knotengewichtung* für den bipartiten Graphen $G = (U \dot\cup V, E)$ ist eine Funktion $c : U \dot\cup V \to \mathbb{Q}_0^+$ mit der Eigenschaft

$$w(e) \;\le\; c(u) + c(v) \qquad \forall e = \{u, v\} \in E. \tag{4.4}$$

Für jede Kantengewichtsfunktion $w : E \to \mathbb{Q}^+$ existiert stets eine solche Knotengewichtung, man betrachte z.B. zu $w_{\max} := \max_{e \in E} w(e)$ die Funktion

$$c(x) \quad := \quad \begin{cases} w_{\max} & \text{falls } x \in U, \\ 0 & \text{falls } x \in V. \end{cases} \tag{4.5}$$

Die folgende einfache Beobachtung wird der Schlüssel zur Reduktion des gewichteten Matchingproblems auf das ungewichtete (oder Kardinalitäts-) Problem sein:

Proposition 4.2.20 *Sei G ein bipartiter Graph, M^* ein Matching in G und c eine zulässige Knotengewichtung von G. Falls die beiden Bedingungen*

(i) für alle $e = \{u, v\} \in M^*$ gilt: $w(e) = c(u) + c(v)$;

(ii) für alle freien Knoten $x \in U \dot\cup V$ gilt: $c(x) = 0$,

erfüllt sind, so ist M^* ein maximum Matching in G.

Beweis. Für jedes beliebige Matching M in G und jede zulässige Knotengewichtung c gilt

$$w(M) \;=\; \sum_{e \in M} w(e) \;\le\; \sum_{e = \{u,v\} \in M} c(u) + c(v) \;\le\; \sum_{x \in U \dot\cup V} c(x).$$

Aufgrund von (i) und (ii) gilt für M^* in beiden Ungleichungen sogar Gleichheit. Es folgt $w(M) \le w(M^*)$ für alle Matchings M in G. □

Der Algorithmus für gewichtetes bipartites Matching startet nun mit dem leeren Matching und der Knotengewichtung c aus (4.5). Um Bedingung (i) zu garantieren, betrachtet man das Matching M selbst, einen Ungarischen dag und M-augmentierende Pfade nur im sogenannten *Gleichheitsgraphen* $G_c^= = (U \dot\cup V, E^=)$ von G bezüglich c, der durch $E^= := \{e = \{u, v\} \in E |\; w(e) = c(u) + c(v)\}$ definiert ist. Der Algorithmus gliedert sich in Phasen; in jeder Phase wird das Matching entlang eines augmentierenden Pfades vergrößert und, wenn nötig, c so abgeändert, daß sich $\sum_{x \in U \dot\cup V} c(x)$ verkleinert. Wenn dies schließlich nicht mehr möglich ist, werden wir eine Knotengewichtung c angeben, so daß die Bedingungen

(i) und (ii) erfüllt sind; das konstruierte Matching ist dann also schon von maximalem Gewicht.

Wie sieht eine einzelne Phase aus? Zunächst wird M – wie in der `WHILE`-Schleife der Prozedur UNGARISCHER_DAG – in $G_c^=$ augmentiert, bis man keinen M-augmentierenden Pfad (in $G_c^=$) mehr findet (zu Beginn wird also ein maximum Matching unter den Kanten $e \in E$ mit $w(e) = w_{\max}$ gebildet). Seien S und T die durch den Ungarischen dag in $G_c^= \subseteq G$ definierten Knotenmengen (beachte: $\Gamma_{G_c^=}(S) = T$). Definiere $\delta := \min\{c(u) : u \in S\}$ und

$$\epsilon := \begin{cases} \min\{c(u) + c(v) - w(\{u,v\}) : u \in S \wedge v \in \Gamma_G(u) \setminus T\} & \text{falls } \langle S, \Gamma_G(S) \setminus T\rangle \neq \emptyset, \\ \delta & \text{sonst.} \end{cases}$$

Für jede zulässige Knotengewichtung c gilt offenbar $\epsilon \geq 0$. Wir nehmen zunächst $\epsilon < \delta$ (d.h. insbesondere $\langle S, \Gamma_G(S) \setminus T\rangle \neq \emptyset$) an. Dann können wir die Knotengewichtung c (die duale Lösung) wie folgt abändern:

$$c'(x) \quad := \quad \begin{cases} c(x) - \epsilon & \text{falls } x \in S, \\ c(x) + \epsilon & \text{falls } x \in T, \\ c(x) & \text{sonst.} \end{cases} \tag{4.6}$$

Wegen $\epsilon < \delta$ gilt $c'(x) \geq 0$ für alle $x \in U \dot\cup V$, und auch (4.4) wird von c' erfüllt: offenbar könnte die Bedingung (4.4) höchstens für eine Kante $\{u,v\} \in E$ mit $u \in S$ und $v \in \Gamma_G(S) \setminus T$ verletzt sein – ϵ wurde jedoch minimal gewählt unter allen solchen Kanten; also ist c' wiederum eine zulässige Knotengewichtung von G. Der Gleichheitsgraph $G_{c'}^=$ enthält zudem alle Matchingkanten aus $G_c^=$, alle Kanten den Ungarischen dag's sowie mindestens eine Kante von S nach $\Gamma_G(S) \setminus T$, die in $G_c^=$ noch nicht enthalten war und um die der dag nun erweitert werden kann. Die Konstruktion des dag's kann nun (mit neuer Gewichtsfunktion $c := c'$) solange fortgesetzt werden, bis entweder ein M-augmentierender Pfad in $G_c^=$ gefunden und eine neue Phase eingeleitet wird, oder aber schließlich $\epsilon \geq \delta$ gilt. In diesem Fall ist M jedoch schon ein Matching maximalen Gewichts in G: um dies einzusehen, ändere man c gemäß (4.6) um δ statt ϵ ab. Die erhaltene Knotengewichtung c' ist wiederum zulässig, doch nun gilt $c(x) = 0$ für alle freien Knoten $x \in U \dot\cup V$, denn:

- zu Beginn hatten alle Knoten $u \in U$ dieselbe Knotengewichtung $c(u) = w_{\max}$;
- in jeder Phase enthielt S stets alle zu diesem Zeitpunkt noch freien Knoten in U, und die Knotengewichtung wurde für alle diese Knoten, wenn sie geändert wurde, um denselben Betrag ϵ geändert;
- da die noch freien Knoten $u \in U$ alle Abänderungen von c „mitgemacht“ haben, haben sie am Ende alle minimalen c-Wert: $c(u) = \delta$;
- zu Beginn hatten alle Knoten $v \in V$ dieselbe Knotengewichtung $c(v) = 0$;
- im Teil V wird jedoch nur bei gematchten Knoten der c-Wert erhöht, und ein Knoten $v \in G$, der einmal von einem Matching überdeckt wird, wird auch von allen im Weiteren konstruierten Matchings überdeckt.

Da darüberhinaus stets $M \subseteq E^=$ gilt (beachte: nur Kanten der Form $e \in \langle U \setminus S, T\rangle$, also Nicht-Matchingkanten, können aus $G_c^=$ wieder herausfallen), erfüllt M alle Voraussetzungen von Proposition 4.2.20 und ist daher in der Tat von maximalem Gewicht.

Überlegen wir uns nun, wie dieser Algorithmus mit Laufzeit $\mathcal{O}(n^3)$ implementiert werden kann. Da jede Phase $|M|$ vergrößert, gibt es offenbar nur höchstens $n/2+1$ viele Phasen. Eine einzelne Phase kann jedoch in Zeit $\mathcal{O}(n^2)$ implementiert werden; das Vorgehen ist hierbei ähnlich wie in der Prozedur UNGARISCHER_DAG eine Breitensuche von der Menge *U_Terminale* der freien Knoten im Teil U aus, außer daß hier alle augmentierenden Pfade recht sind (d.h. die Abfrage „`IF` $level[t] < k$“ entfällt) und daß hier Kanten $e = \{u, v\} \in E \setminus E^=$ eine Sonderbehandlung erfahren: über sie wird schon während der Breitensuche ein Feld $eps[v]$, $v \in \Gamma_G(S) \setminus T$, berechnet, so daß nach Ablauf der `WHILE`-Schleife, wenn nicht augmentiert werden konnte, für alle $v \in \Gamma_G(S) \setminus T$ gilt: $eps[v] = \min\{w(\{s, v\}) : \{s, v\} \in E \wedge s \in S\}$. Dazu wird zu Beginn $eps[v] = \infty$ für alle $v \in V$ initialisiert. Im Verlauf der Breitensuche wird dann für jeden Knoten $u \in S$, der einen Nachbarn $v \in V$ in G, aber nicht in $G^=$ hat, $eps[v]$ auf $c(u) + c(v) - w(\{u, v\})$ gesetzt, falls dies $eps[v]$ verkleinert (analog zum Algorithmus von DIJKSTRA: hier bildet sozusagen $\Gamma = \Gamma_G(S) \setminus T$ die Grenzschicht). Falls nun M in der `WHILE`-Schleife der Prozedur UNGARISCHER_DAG nicht augmentiert werden konnte, so läßt sich offenbar ϵ in Zeit $\mathcal{O}(n)$ bestimmen; und falls $\epsilon < \delta$, so kann die Knotengewichtung c (und das Feld $eps[\cdot]$) in Zeit $\mathcal{O}(n)$ abgeändert werden. Falls es dann einen freien Knoten $v \in \Gamma_G(S) \setminus T$ mit $eps[v] = \epsilon$ gibt, so wird das Matching entlang des augmentierenden *U_Terminale*$-v-$Pfades augmentiert, andernfalls setzt man die `WHILE`-Schleife der Prozedur UNGARISCHER_DAG fort – mit Q als der Menge der Matchingpartner der Knoten $\{v \in \Gamma_G(S) \setminus T : eps[v] = \epsilon\}$.

Nach jeder Minimumbildung für ϵ gefolgt von einer Abänderung der Knotengewichtung c wird mindestens ein Knoten $v \in V$ im Graphen *dag* zu T hinzugefügt. Also werden diese Schritte nur $\mathcal{O}(n)$-mal ausgeführt und kosten pro Phase $\mathcal{O}(n^2)$ Zeit. Alle anderen Operationen in einer Phase sind wie bei einer Breitensuche $\mathcal{O}(n+m)$.

Wir fassen zusammen:

Proposition 4.2.21 (KUHN 1955; MUNKRES 1957) *In einem bipartiten Graphen kann ein Matching maximalen Gewichts in Zeit $\mathcal{O}(n^3)$ konstruiert werden.*

Der schnellste bekannte Algorithmus für dieses Problem erreicht durch Verwendung von FIBONACCI-Heaps Laufzeit $\mathcal{O}(n(m + n\log n))$, siehe [FT87].

Übungen

Übung 4.2.22 *Ist die Bipartitionsmatrix eines bipartiten Graphen nicht-singulär, so besitzt er ein perfektes Matching.*

Übung 4.2.23 *Ein bipartiter Graph $G = (U \dot\cup V, E)$ mit $|U| = |V| = n$ und $m > (k-1)n$ Kanten besitzt ein Matching mit mindestens k Kanten.*

Übung 4.2.24 *Man leite aus dem Satz 4.2.9 von* HALL *seine Defektform (d.h. Satz 4.2.8) ab.* [*Hinweis:* füge G so Knoten hinzu, daß der Satz von HALL anwendbar wird.]

Sei $S_1, \ldots, S_m$ eine Familie von (nicht notwendig paarweise verschiedenen) Teilmengen einer endlichen Menge $\mathcal{S}$. Ein *Repräsentantensystem* von $S_1, \ldots, S_m$ ist eine injektive Abbildung $f : S_1, \ldots, S_m \to \mathcal{S}$, so daß $f(S_i) \in S_i$, $i = 1, \ldots, m$.

Übung 4.2.25 [Hal35] *Eine Familie $S_1, \ldots, S_m$ von endlichen Mengen besitzt ein Repräsentantensytem genau dann, wenn für alle Indexmengen $I \subseteq \{1, \ldots, m\}$ gilt*
$|\bigcup_{i \in I} S_i| \geq |I|$.

Übung 4.2.26 *Eine Menge $S \subseteq V$ ist eine maximum stabile Knotenmenge genau dann, wenn jede von S disjunkte stabile Menge $U \subseteq V \setminus S$ nach S gematcht werden kann.*

Übung 4.2.27 [Tov84] *Die Einschränkung von 3-*SAT *(genau drei Literale pro Klausel, keine Wiederholungen) auf Boolesche Formeln, in denen jede Variable höchstens ($s = 3$)-mal (positiv oder negiert) auftritt, liegt in* P. [*Bemerkung:* NP-vollständig für $s = 4$. Vergleiche auch Übung 1.4.17]

Die folgende Übung verallgemeinert sowohl Proposition 4.1.4 wie auch Übung 1.1.38a.

Übung 4.2.28 [Bra94b] *Ein Graph $G = (V, E)$ enthält jeden Wald W mit $k \leq \delta(G)$ Kanten als Subgraphen, falls nur $n(W) \leq n(G)$.* [*Hinweis:* O.B.d.A. enthalte W keine isolierten Knoten. Schneide bei jeder Komponente von W ein Blatt ab. Identifiziere die Menge X der Nachbarn der abgeschnittenen Blätter B mit einer Knotenmenge $X' = argmin\ \{m(G[V']) : V' \in \binom{V}{|X|}\}$ in G. Bette sodann $W - X - B$ und schließlich B (Satz von HALL) in G ein.] *Man gebe einen $\mathcal{O}(k^2n^2)$-Einbettungsalgorithmus an.*

Es gibt eine Funktion $f(k, n) \leq \max\{2k^2, kn\}$, so daß auch jeder Graph mit $m \geq f(k, n)$ Kanten jeden Wald mit k Kanten und ohne isolierte Knoten als Subgraph enthält [Bra94b].

Übung 4.2.29 a) *Man folgere den Satz 4.2.15 von* KÖNIG *auch aus dem Satz 2.2.1 von* MENGER, *indem man dem Graphen zwei Knoten s und t geeignet hinzufügt.*
b) *Man leite die Sätze 4.2.15 und 4.2.9 von* KÖNIG *bzw.* HALL *wechselseitig auseinander ab.*

Übung 4.2.30 a) *Ein Graph G ist bipartit genau dann, wenn $\rho(H) = \alpha(H)$ für alle Subgraphen H von G.* **b)** *Für einen bipartiten Graphen G gilt $\theta(G) = \alpha(G)$.* **c)** *Es gibt einen polynomiellen Algorithmus für minimum Cliquenüberdeckung in bipartiten Graphen.*

Übung 4.2.31 [Het64] *Für einen bipartiten Graphen $G = (U \dot\cup V, E)$, $n \geq 4$, sind äquivalent:*

(i) $|U| = |V|$ *und* $|\Gamma(X)| > |X|$ *für alle* $X \subset U$ *mit* $\emptyset \neq X \neq U$;

(ii) $\forall u \in U\ \forall v \in V$: *($G - u - v$ besitzt ein perfektes Matching);*

(iii) G ist zusammenhängend und jede Kante ist in einem perfekten Matching enthalten.

(iv) G besitzt genau die minimum Knotenüberdeckungen U und V.

Die Verallgemeinerung von 1-Faktoren sind *r-Faktoren*. Ein r-Faktor ist ein spannender r-regulärer Subgraph. Ein 2-Faktor ist also beispielsweise eine Vereinigung disjunkter Kreise, die alle Knoten überdeckt, und ein zusammenhängender 2-Faktor ein Hamiltonkreis. Ein Graph heißt *r-faktorisierbar*, falls sich seine Kantenmenge disjunkt in r-Faktoren zerlegen läßt.

Übung 4.2.32 [Pet91] *Ein Graph ist* 2-faktorisierbar *genau dann, wenn er $2r$-regulär ist für ein $r \in \mathbb{N}$.* [*Hinweis* zu „$\Leftarrow$“: orientiere E gemäß einer Eulertour und benutze Korollar 4.2.11.]

Man beachte, daß in Übung 3.2.22 2-zusammenhängende, 4-reguläre Graphen ohne Hamiltonkreis konstruiert wurden.

Übung 4.2.33 [Kön31, Ege31] *Sei A eine (0,1)-Matrix. Dann ist die minimale Anzahl von Zeilen und Spalten, die alle Einsen von A enthalten, gleich der maximalen Anzahl von Einsen, von denen keine zwei in derselben Zeile oder Spalte stehen.*

Einen äußerst eleganten, aber nicht konstruktiven Beweis des Satzes 4.2.15 von KÖNIG hat DE CAEN gegeben:

Übung 4.2.34 [Cae88] *Sei $G = (V, E)$ ein bipartiter Graph. Ein Knoten $v \in V$ heißt* universal, *falls er von jedem maximum Matching M in G (d.h. $|M| = \nu(G)$) überdeckt wird.*
a) *Sei $\{x, y\}$ eine Kante in G. Dann ist mindestens einer der Knoten x und y ein universaler Knoten; Insbesondere besitzt also jeder bipartite Graph einen universalen Knoten.*
b) *Beweise hiermit Satz 4.2.15 [Hinweis: Induktion nach $\nu(G)$].*

Übung 4.2.35 *Für den Graphen Q_d des d-dimensionalen Hyperwürfels gilt:*
a) $\alpha(Q_d) = \tau(Q_d) = 2^{d-1} = \rho(Q_d) = \nu(Q_d)$;
b) *Q_d ist r-faktorisierbar genau dann, wenn $r|d$.*

Übung 4.2.36 *Die Umkehrung von Satz 4.2.15 ist falsch: man finde den kleinsten Graphen mit $\tau(G) = \nu(G)$, der nicht bipartit ist. Es gilt jedoch: Ein Graph G ist bipartit genau dann, wenn $\tau(H) = \nu(H)$ für alle Subgraphen $H \subseteq G$.*

Übung 4.2.37 a) *Die Knotenmenge eines Graphen G kann durch $\alpha(G)$ knotendisjunkte Pfade überdeckt werden* [GM60]. [*Hinweis:* betrachte eine Folge $S_1, S_2, \ldots, S_k$, wobei S_1 eine maximum stabile Menge in G ist und S_{i+1} eine maximum stabile Menge in $G - S_1 - \ldots - S_i$]
b) Ein Subgraph S eines Graphen G heiße *transversal*, wenn er alle maximum stabilen Mengen in G schneidet, d.h. wenn $\alpha(G - V(S)) < \alpha(G)$. *Jeder maximale Pfad in einem Graphen ist transversal. Jeder Graph besitzt einen transversalen Kreis, K_2 oder K_1.* [*Hinweis:* betrachte einen längsten transversalen Pfad in G] *Die Knotenmenge eines Graphen G kann durch höchstens $\alpha(G)$ knotendisjunkte Kreise, K_2's und K_1's überdeckt werden* [CJ96].

4.3 Matching in allgemeinen Graphen

Wir werden in diesem Abschnitt eine dem Satz 4.2.8 ähnelnde Charakterisierung der Matchingzahl $\nu(G)$ in allgemeinen Graphen herleiten, die Einblick in die Struktur von maximum Matchings gewährt. Als Korollar erhalten wir den Satz von PETERSEN. Anschließend behandeln wir einen $\mathcal{O}(n^3)$-Algorithmus, um ein maximum Matching in einem Graphen zu bestimmen. Während also das Problem INDEPENDENT SET NP-schwer ist, ist das analoge Problem für Kanten (das dem INDEPENDENT SET Problem auf Linegraphen entspricht) polynomiell lösbar.

4.3.1 Die Sätze von BERGE, TUTTE und PETERSEN

Überlegen wir uns aber folgendes zur Struktur von maximum Matchings. Sei die Anzahl ungerader Komponenten eines Graphen H mit $q(H)$ bezeichnet. Für jede Knotenmenge $S \subseteq V$ muß ein maximum Matching in jeder ungeraden Komponente von $G - S$ (mindestens) einen Knoten entweder ungematcht lassen oder aber zu einem Knoten aus S matchen. Ist $q(G - S) > |S|$, so bleiben mindestens $q(G - S) - |S|$ Knoten ungematcht - es folgt

$$2\nu(G) \;\leq\; n - \max_{S \subset V}\{q(G - S) - |S|\}. \tag{4.7}$$

Wegen $S = \emptyset$ folgt $\max\{q(G - S) - |S| : \; S \subset V\} \geq 0$. Auch in Ungleichung (4.7) gilt tatsächlich wieder Gleichheit, was uns die folgende Charakterisierung der Matching-Zahl $\nu(G)$ verschafft.

Satz 4.3.1 (BERGE 1958) $\qquad 2\nu(G) \;=\; n - \max_{S \subset V}\{q(G - S) - |S|\}.$

Der Wert $d(G) := \max_{S \subset V}\{q(G - S) - |S|\}$ heißt auch der „Defekt“ von G. Bevor wir diesen Satz beweisen, definieren wir noch (vgl. Übung 4.3.16):

Definition 4.3.2 *Ein Graph G heißt* faktorkritisch, *falls G selbst kein perfektes Matching besitzt, aber $G - v$ für alle $v \in V$.*

Beweis von Satz 4.3.1. Nach obigem bleibt lediglich $n - 2\nu(G) \leq d(G)$ zu zeigen. Wir induzieren über n. Der Fall $n = 1$ ist trivial.
Sei also G ein Graph der Ordnung $n \geq 2$ und $S \subseteq V$ eine kardinalitätsmaximale Knotenmenge mit der Eigenschaft $q(G - S) - |S| \;=\; d(G)$. Wir werden nun die Existenz eines Matchings M^* in G nachweisen, das nur d Knoten von G nicht überdeckt. Damit ist dann $2\nu(G) \geq 2|M^*| \geq n - d(G)$ und der Satz bewiesen. Insbesondere ist M^* wegen (4.7) ein maximum Matching in G.

Behauptung 1: Jede Komponente von $G - S$ ist ungerade.
Angenommen, D wäre eine gerade Komponente von $G - S$. Dann wäre $|D| \geq 2$, und für jeden Knoten $v \in D$ gälte $q(D - v) \geq 1$, woraus

$$q(G - (S + v)) - |S + v| \;\geq\; q(G - S) + 1 - (|S| + 1) \;=\; q(G - S) - |S|$$

im Widerspruch zur kardinalitätsmaximalen Wahl von S.

Behauptung 2: Jede Komponente C von $G - S$ ist faktorkritisch.
Sei C ein Komponente von $G - S$ und $v \in C$ beliebig. Nach Induktionsvoraussetzung genügt es, $d(C - v) = 0$ zu zeigen. Da $C - v$ gerade ist, gilt $d(C - v) \geq 0$, und es bleibt,

$q(C-v-R)-|R| \leq 0$ zu zeigen für alle $R \subseteq V(C-v)$. Aus der kardinalitätsmaximalen Wahl von S folgt

$$q(G-S)-|S| \;>\; q(G-(S\cup\{v\}\cup R))-|S\cup\{v\}\cup R| \;=\; q(G-S)-1+q(C-v-R)-(|S|+1+|R|),$$

woraus

$$q(C-v-R)-(|R|+1) \;<\; 1. \tag{4.8}$$

Da C ungerade Ordnung hat, gilt

$$q(C-S')-|S'| \;\neq\; 0 \qquad \forall S' \subseteq V(C).$$

Mit $S' := v+R$ folgt hieraus, daß die linke Seite von (4.8) ungleich 0 ist. Mithin ist die linke Seite von (4.8) sogar ≤ -1, und es ergibt sich wie gewünscht $q(C-v-R)-|R| \leq 0$. Wir werden nun noch jeden Knoten $s \in S$ zu einem Knoten c_s in einer Komponente C_s von $G-S$ matchen, so daß die C_s, $s \in S$, alle verschieden sind. Da die $C_s - c_s$, $s \in S$, ein perfektes Matching besitzen und alle übrigen Komponenten C von $G-S$ ein fast-perfektes, verbleiben dann lediglich $q(G-S)-|S| = d$ nicht gematchte Knoten.
Seien also $C_1,\ldots,C_q$, $q := q(G-S)$, die Komponenten von $G-S$. Wir bilden den bipartiten Graphen $B = (S, \{C_1,\ldots,C_q\}, E_B)$ und verbinden einen Knoten $s \in S$ mit einer Komponente C_i genau dann, wenn s Nachbarn in C_i hat.

Behauptung 3: B besitzt ein Matching, das alle Knoten aus S überdeckt.
Wir weisen die HALL-Bedingung (4.3) nach. Sei $X \subseteq S$. Offenbar gilt

$$q(G-S) \;\leq\; q(G-(S\setminus X))+|\Gamma_B(X)|.$$

Andererseits ist nach Wahl von S

$$q(G-S)-|S| \;\geq\; q(G-(S\setminus X))-|S\setminus X|.$$

Subtrahiert man die zweite von der ersten Ungleichung, findet man wie gewünscht

$$\begin{array}{rrcl} & |S| & \leq & |S\setminus X|+|\Gamma_B(X)| \\ \Leftrightarrow & |X| & \leq & |\Gamma_B(X)|. \end{array}$$

□

Der obige Beweis geht auf Ideen von GALLAI, ANDERSON und MADER zurück (siehe [Hal89]) und gibt Einblick in die Struktur von maximum Matchings, da diese alle so gebildet sein müssen, wie das im Beweis konstruierte M^*. Dies führt z.B. zu einem einfachen Beweis der Aussage aus Übung 4.3.13a. Als Spezialfall des Satzes 4.3.1 finden wir den historisch früheren Satz von TUTTE:

Korollar 4.3.3 (TUTTE 1947) *Ein Graph G besitzt ein perfektes Matching genau dann, wenn gilt*

$$q(G-S) \;\leq\; |S| \qquad \forall S \subset V. \tag{4.9}$$

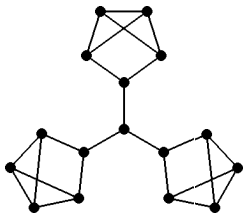
Bedingung (4.9) heißt auch TUTTE-Bedingung. TUTTE bewies auch ein Analogon hierzu über die Existenz von r-Faktoren in Graphen, siehe [Bol78a].

Als Anwendung dieses Ergebnisses betrachten wir reguläre Graphen. Ein 1-regulärer Graph ist offenbar ein perfektes Matching. Ein 2-regulärer Graph ist die disjunkte Vereinigung von Kreisen. Also besitzt ein 2-regulärer Graph ein perfektes Matching genau dann, wenn seine Kreise gerade sind.

Lemma 4.3.4 *Sei G ein kubischer Graph, der ein perfektes Matching M besitzt. Dann ist jede Brücke von G in M enthalten.*

Beweis. Sei G o.B.d.A. zusammenhängend und $e \in E$ eine Brücke in G. Seien G_i, $i = 1, 2$, die Komponenten von $G - e$ mit n_i Knoten und m_i Kanten. Aus der Gleichung für die Gradsumme $3n_i = 2m_i + 1$ folgt, daß n_i ungerade ist. Die Graphen G_i besitzen also kein perfektes Matching und mithin auch nicht $G - e$. □

Wir folgern aus diesem Lemma, daß beispielsweise der folgende kubische Graph kein perfektes Matching besitzt.



Mit Hilfe des Satzes von TUTTE ist es aber ein leichtes, die folgende hinreichende Bedingung herzuleiten.

Korollar 4.3.5 (PETERSEN 1891) *Ein zusammenhängender kubischer Graph G mit höchstens zwei Brücken besitzt ein perfektes Matching.*

Beweis. Wir weisen die TUTTE-Bedingung (4.9) nach. Wegen $3n = 2m$ ist n gerade und (4.9) für $S = \emptyset$ erfüllt. Sei also $|S| \geq 1$, und seien $C_1, \ldots, C_q$ die ungeraden Zusammenhangskomponenten von $G - S$. Wenn e_i die Anzahl der Kanten zwischen S und C_i bezeichnet, so ergibt ein Zählen der Kanten $3|V(C_i)| = 2|E(C_i)| + e_i$. Mit $|V(C_i)|$ sind mithin auch die e_i, $i = 1, \ldots, q$, ungerade. Da G zudem nur höchstens zwei Brücken besitzt, ist e_i also mindestens $q - 2$ mal mindestens 3, und es folgt

$$\begin{array}{rccc} & 3|S| & \geq & \sum_{i=1}^{q} e_i \quad \geq \; 2 + (q-2) \cdot 3 \\ \Rightarrow & q & \leq & |S| + 4/3. \end{array}$$

Da nun aus Paritätsgründen stets $q \equiv |S| \pmod 2$ gilt, wird schließlich $q \leq |S|$. □

Der obige Graph zeigt, daß dieser Satz bezüglich der Anzahl der Brücken bestmöglich ist.

Bemerkung. Entfernt man ein perfektes Matching aus einem kubischen, brückenlosen Graphen, verbleibt ein 2-Faktor. Dessen Kreise brauchen nicht notwendig gerade zu sein, wie der PETERSEN-Graph (siehe Seite 88) zeigt. Mithin ist ein kubischer, brückenloser Graph i.a. nicht 1-faktorisierbar. Planare, kubische, brückenlose Graphen sind jedoch stets 1-faktorisierbar – dies ist gerade äquivalent zum 4-Farben-Satz, vgl. Korollar 6.3.8!

4.3.2 Ein Algorithmus für das allgemeine Matchingproblem

Wir werden in diesem Abschnitt einen polynomiellen Algorithmus für das allgemeine Matching-Problem vorstellen. Dieser Algorithmus geht auf EDMONDS [Edm65] zurück und beruht, wie auch schon der Algorithmus für den bipartiten Fall, auf dem Auffinden augmentierender Pfade.

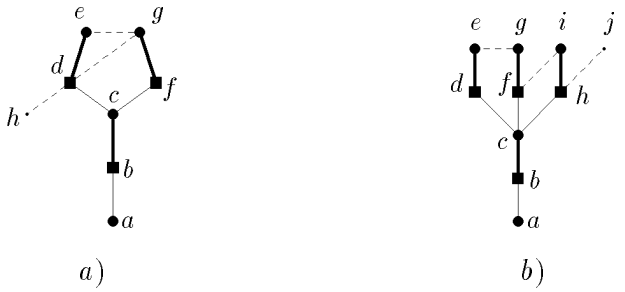
Ausgehend vom leeren Matching wird in jeder Phase des Algorithmus ein augmentierender Pfad gefunden, oder aber festgestellt, daß kein augmentierender Pfad existiert, womit nach Satz 4.1.6 nachgewiesen ist, daß das gefundene Matching ein maximum Matching ist. Offensichtlich terminiert dieser Algorithmus nach $\mathcal{O}(n)$ Phasen.

Für die Suche nach einem augmentierenden Pfad innerhalb einer Phase startet man von einem ungematchten Knoten v (falls kein solcher existiert, ist das Matching offensichtlich bereits ein maximum Matching) und führt von diesem ausgehend eine spezielle Variante des BFS durch, die die Knoten des Graphen mit *even* oder *odd* kennzeichnet, entsprechend der Existenz eines geraden oder ungeraden alternierenden von v ausgehenden Pfades zu diesem Knoten. Der Graph enthält sodann genau dann einen von v ausgehenden augmentierenden Pfad, falls es einen mit *odd* gelabelten Knoten gibt, der ungematcht ist. Indem man den BFS von allen ungematchten Knoten des Graphen durchführt, wird man also einen augmentierenden Pfad in dem Graphen finden, oder nachgewiesen haben, daß kein solcher existiert.

Wir beschreiben nun genauer, wie man mithilfe einer speziellen Variante des BFS die gerade beschriebene Knotenlabellung bestimmen kann. Wir gehen von einem beliebigen ungematchten Knoten v aus und kennzeichnen diesen mit *even*. Sodann fügen wir für alle Nachbarn u von v die *gerichtete* Kante uv in eine Queue Q ein. Eine Eigenschaft von Q wird sein, daß der Anfangspunkt einer jeden in ihr enthaltenen Kante stets das Label *even* trägt. Wir entfernen nun die erste Kante xy aus Q und gehen wie folgt vor:

1. Falls y das Label *odd* trägt, so ignorieren wir die Kante xy.
2. Falls y ein noch ungelabellter Knoten ist geben wir y das Label *odd* und unterscheiden zwei Fälle:
   - 2.1 Falls y ein ungematchter Knoten ist, so haben wir einen augmentierenden Pfad gefunden und können die aktuelle Phase beenden;
   - 2.2 ansonsten sei yz die mit y inzidente Matchingkante. Wir labeln z mit *even* und fügen alle mit z inzidenten Kanten mit Ausnahme der Kante zy zu Q hinzu.
3. Falls y das Label *even* trägt, so bedeutet dies, daß wir einen ungeraden Kreis gefunden haben. In diesem Fall fügen wir alle von mit *odd* gelabelten Knoten des Kreises ausgehenden Kanten in Q ein und labeln diese Knoten *even*. Alle Kanten aus Q, die zwischen zwei Knoten des ungeraden Kreises verlaufen, werden aus Q entfernt.

Zur Verdeutlichung des letzten Falles siehe die Abbildung $x.a)$.



$a)$ $b)$

Abbildung 4.1: Das Auftreten ungerader Kreise im Matchingalgorithmus

In diesem Beispiel seien alle nicht gestrichelten Kanten bereits abgearbeitet worden, wobei die fett gezeichneten Kanten Matching-Kanten sind. Runde Knoten sind mit *even*, eckige Knoten mit *odd* gelabelt,während der Knoten h noch ungelabelt ist. In der Queue Q befinden sich zum aktuellen Zeitpunkt die Kanten ge, gd und eg. Wird die Kante ge

abgearbeitet, so sind deren beide Endknoten *even* gelabelt, womit wir uns im obigen Fall 3. befinden. Wir haben den ungeraden Kreis $cdegf$ gefunden und fügen alle von d und f ausgehenden nicht-Matchingkanten in die Queue Q ein und labeln d und f mit *even*. Alle Kanten, die beide Endknoten im ungeraden Kreis $cdegf$ haben, entfernen wir aus Q. Als einzige Kante verbleibt somit nach diesem Schritt die Kante dh in Q und im nächsten Schritt finden wir den augmentierenden Pfad $abcfgedh$.

Die obige Vorgehensweise wirft jedoch einige Probleme auf: Sobald man auf ineinandergeschachtelte ungerade Kreise stößt, ist es nicht sofort klar, in welcher Form man sich die ungeraden Kreise merken muß, um feststellen zu können, welchen neuen ungeraden Kreis man mit einer neu eingefügten Kante, deren beide Endknoten *even* gelabelt sind, schließt.Außerdem ist noch nicht klar, wie man nicht nur feststellt, daß ein augmentierender Pfad vorhanden ist, sondern diesen auch tatsächlich angeben kann. In Abbildung x.b) ist ein Beispiel für das Auftreten ineinandergeschachtelter ungerader Kreise gezeigt. Es seien zum aktuellen Zeitpunkt die Kanten if, eg, ge in der Queue. Die Kante if wird ignoriert, da f *odd* gelabelt ist. Die Kante eg schließt hingegen den ungeraden Kreis $egfcd$. Daher wird der Knoten f als *even* gelabelt und die Kante fi in die Queue aufgenommen. Sie verbindet nun wiederum zwei *even* gelabelte Knoten und der mit dieser Kante geschlossene ungerade Kreis ist $fihcdeg$. Damit wird der Knoten h als *even* gelabelt und die Kante hj der Queue hinzugefügt und somit schließlich der augmentierende Pfad $abcdegfihj$ gefunden. Dieses Beispiel zeigt zudem, daß es durchaus ein Unterschied ist, ob man die Kante if oder fi betrachtet. Während erstere ignoriert wurde, schloß letztere einen ungeraden Kreis.

Zur Umgehung der gerade angedeuteten Schwierigkeiten und zur Erzielung einer besseren Laufzeit, werden wir daher im folgenden die von EDMONDS eingeführte Idee des Kontrahierens ungerader Kreise benutzen.

Ist $G = (V, E)$ ein Graph und S eine beliebige Teilmenge von V, so versteht man unter der Kontraktion von S denjenigen Graphen, den man erhält, indem man in den Graphen $G - S$ einen Knoten s einfügt, der mit allen Knoten adjazent ist, die zu einem Knoten aus S in G adjazent sind. Den so erhaltenen Graphen bezeichnen wir mit $G \times S$. Enthält G ein Matching M so induziert dieses ein Matching in $G \times S$, falls höchstens eine Matching-Kante zwischen S und $G - S$ verläuft. Dieses Matching bezeichnen wir mit $M \times S$.

Schritt 3. des oben angegebenen Verfahrens ersetzen wir nun durch die folgende Variant 3'., in der der gefundene ungerade Kreis kontrahiert wird:

3'. Falls y das Label *even* trägt, so bedeutet dies, daß wir einen ungeraden Kreis gefunden haben. In diesem Fall kontrahieren wir die Knotenmenge des Kreises zu einem Knoten s und fügen alle von s ausgehenden und noch nicht in Q enthaltenen nicht-Matching-Kanten in Q ein. Der Knoten s wird mit *even* gelabelt.

Damit liefert uns dieses Verfahren ausgehend von einem beliebigen ungematchten Knoten v in jedem Schritt einen sogenannten *alternierenden Baum*: die Knoten des Baumes sind alle mit *even* oder *odd* gelabelten Knoten, die Kanten des Baumes sind alle diejenigen Kanten, über die erstmals ein Knoten des Baumes erreicht wird. Nach Definition des oben angegebenen Verfahrens sind alle Blätter des Baumes stets mit *even* gelabelt, während alle mit *odd* gelabelten Knoten stets den Grad 2 im Baum haben. Falls das Verfahren ausgehend von dem ungematchten Knoten v abbricht, ohne daß ein augmentierender Pfad gefunden wurde, so nennt man den zu diesem Zeitpunkt vorhandenen in v gewurzelten

alternierenden Baum einen *ungarischen Baum*. Ein solcher stellt ein Zertifikat dafür dar, daß es keinen von v ausgehenden augmentierenden Pfad in dem Graphen gibt.

Der Terminologie von EDMONDS folgend bezeichnen wir die in Schritt 3'. auftretenden ungeraden Kreise als *Blossoms* (engl.: "Blüte"). Zu jedem Blossom führt von dem ungematchten Knoten v, in dem wir die Suche nach augmentierenden Pfaden gestartet haben, ein eindeutig bestimmter alternierender Pfad zu dem sogenannten *Fußpunkt* des Blossoms. Den von v aus zu diesem Punkt führenden alternierenden Pfad bezeichnen wir als *Stamm* des Blossoms (dieser kann leer sein). Das folgende Lemma ist die Grundlage des Korrektheitsnachweises des oben angegebenen Algorithmus. Es stellt sicher, daß der Algorithmus stets einen augmentierenden Pfad findet, sofern ein solcher existiert.

Lemma 4.3.6 *Ist S ein Blossom in G bezüglich eines Matchings M, so ist M genau dann ein maximum Matching in G, wenn $M \times S$ ein maximum Matching in $G \times S$ ist.*

Beweis. Angenommen $M \times S$ ist kein maximum Matching in $G \times S$. Dann gibt es einen $M \times S$-augmentierenden Pfad P in $G \times S$. Falls $s \notin P$ so ist P auch M-augmentierender Pfad in G. Falls $s \in P$, so gibt es einen (eindeutigen) Pfad Q gerader Länge in S, der den Fußpunkt von S als einen Endpunkt hat, so daß der Pfad P', den man erhält, indem man das Pfadstück Q in P einfügt, ein M-augmentierender Pfad in G ist. Also war auch M kein maximum Matching.

Sei nun umgekehrt M kein maximum Matching in G. Ohne Einschränkung dürfen wir annehmen, daß der Fußpunkt von S ein ungematchter Knoten ist. Dies kann man z.B. durch Alternieren der Kanten des Stammes von S erreichen. Entsprechend ist dann auch der Knoten s in $G \times S$ ungematched. Es sei P ein augmentierender Pfad in G. Falls $P \cap S = \emptyset$, so ist P auch augmentierender Pfad in $G \times S$. Ansonsten seien a und b die Endpunkte von P und f der Fußpunkt von S, wobei $a \neq f$ sei. Dann entspricht dem Teilpfad von P, der von a ausgehend erstmals einen Knoten aus S enthält ein augmentierender Pfad in $G \times S$. Somit war auch $M \times S$ kein maximum Matching. □

Datenstrukturen. Wir werden nun Datenstrukturen angeben, mit Hilfe derer man obigen Algorithmus effizient implementieren kann. Es zeigt sich, daß der größte Teil des Algorithmus und der Datenstrukturen lediglich für die explizite Augmentierung des gefundenen augmentierenden Pfades dienen. D.h., benötigt man lediglich einen Algorithmus, der für ein gegebenes Matching testet, ob dieses maximum ist, so ließe sich das mit wesentlich weniger Aufwand erreichen.

Für alle Knoten des Graphen und die Knoten, die durch das Kontrahieren von Blossoms entstanden sind (letztere werden wir im folgenden *Pseudoknoten* nennen), werden eine Reihe von Werten gespeichert, die im folgenden genauer erläutert werden: *matched[v]* gibt an, ob Knoten v gematcht ist, *parity[v]* enthält die Parität von v (even, odd oder none). Das Feld *matching_edge[v]* enthält die mit v inzidente Matchingkante, falls v gematcht ist. Die Felder *pred_node[v]* und *pred_edge[v]* enthalten den Knoten bzw. die Kante, über die v zum ersten Mal bei der Konstruktion des ungarischen Baumes erreicht wurde. Das Feld *deep[v]* enthält alle diejenigen Knoten aus G, die sich hinter dem (Pseudo-)Knoten v verbergen, d.h. *deep[v]* ist rekursiv definiert als:

$$deep[v] = \begin{cases} v, & \text{fallsv} \in \text{G} \\ \cup_x deep[x] & \text{sonst} \end{cases}$$

Umgekehrt benutzen wir ein Feld *outer[v]*, das für einen Knoten aus G den Pseudo-Knoten enthält, in dem dieser Knoten enthalten ist.

Mit Hilfe dieser Datenstrukturen sieht der Matching-Algorithmus wie folgt aus:

```
PROCEDURE Maximum Matching
  BEGIN
   { Initialisierungen }
   FOR ALL e ∈ E  DO matched[e] := false;
   FOR ALL v ∈ V  DO  BEGIN
     matched[v]          :=  false;
     parity [v]          :=  none;
     pred_node[v]        :=  nil;
     pred_edge[v]        :=  nil;
     matching_edge[v]    :=  nil;
     deep[v]             :=  {v};
     outer[v]            :=  v;
   END ;

   { lasse ungarischen Baum aus jedem ungematchten Knoten wachsen }
   FOR ALL v ∈ V  DO
     IF not matched[v]
       THEN  BEGIN
         initialisiere Q;
         markiere v als besucht;
         parity[v]                :=  even;
         augmentation_found       :=  false;
         füge alle mit v inzidenten nicht-Matchingkanten zu Q hinzu;

         WHILE (Q.size > 0) and not augmentation_found  DO  BEGIN
           sei e = (u, v) nächste Kante aus Q mit outer[u] ≠ outer[v];
           CASE parity[outer[v]]  OF
             none:  IF matched[outer[v]]
                       THEN extend_tree
                       ELSE  augment_path;
             odd : ;
             even: shrink_blossom;
           END ;
         END ;

         expandiere alle noch nicht expandierten Blossoms;

         IF augmentation_found
           THEN setze parity von allen besuchten Knoten auf none
           ELSE  setze parity von allen besuchten Knoten auf odd
         markiere alle Knoten als unbesucht;
       END ;
```

END ;

Die Routinen "extend_tree", "augment_path" und "shrink_blossom" werden im folgenden angegeben. Besonders einfach ist die Routine "extend_tree". Hier müssen lediglich die Werte *pred_node*, *pred_edge* und *parity* für de beiden neu hinzukommenden Knoten aktualisiert werden, sowie die mit dem zweiten Knoten inzidenten nicht-Matchingkanten in die Queue Q eingefügt werden.

PROCEDURE EXTEND_TREE;
BEGIN
parity $[v]$:= odd;
pred_node $[v]$:= u;
pred_edge $[v]$:= e;
markiere v als besucht;

w := der von v verschiedene Endknoten von $matching\_edge[v]$;
parity $[w]$:= even;
pred_node $[w]$:= v;
pred_edge $[w]$:= $matching\_edge[v]$;
markiere w als besucht;

füge alle mit w inzidenten nicht-Matchingkanten zu Q hinzu
END ;

Ähnlich einfach ist die Routine "shrink_blossom". Die Bestimmung der zu einem Blossom gehörenden Knoten und Kanten erfolgt mit Hilfe der Felder *pred_node* und *pred_edge*. Nach obiger Rekursionsvorschrift kann *deep* für den neuen Pseudoknoten leicht berechnet werden und die *outer*-Werte aller zum Blossom gehörenden Knoten aktualisiert werden.

PROCEDURE SHRINK_BLOSSOM;

BEGIN
initialisiere neues Blossom b;
bestimme mit Hilfe der Felder *pred_node* und *pred_edge* die zu b gehörenden
Knoten und Kanten und sortiere sie in zyklischer Reihenfolge in $shallow[b]$ ein;

$deep[b] := \emptyset$;
FOR ALL $x \in shallow[b]$ DO $deep[b] := deep[b] \cup deep[x]$;

füge alle nicht-Matchingkanten, die von odd-Knoten des Blossoms ausgehen, zu Q hinzu;

```
    FOR ALL x ∈ deep[b] DO outer[x] := b;
    update pred_node[b], pred_edge[b], matching_edge[b], outer[b] und parity[b];
  END ;
```

Schließlich folg noch die Routine "augment_path", die zunächst den augmentierenden Pfad im aktuellen Graphen durch Zurücklaufen über die Felder *pred_node* und *pred_edge* bestimmt. In einem zweiten Schritt werden sodann die in diesem Pfad enthaltenen Pseudoknoten sukzessive expandiert. DAzu muß jeweils das geegnete Pfadstück gerader Länge des Blossoms gewählt werden, dessen einer Endpunkt der Fußpunkt des Blossoms ist.

```
PROCEDURE AUGMENT_PATH;
  BEGIN
    augmentation_found := true;
    initialisiere Blossom-Queue B;
    initialisiere Pfad P;
    füge v und e zu P hinzu;

    w := v;

    WHILE w ≠ nil DO BEGIN
      e := pred_edge[outer[w]];
      füge outer[w] und e zu P hinzu;
      falls outer[w] ein Blossom-Knoten ist, so füge w zu B hinzu;
      w := pred_node[outer[w]];
    END ;

    WHILE B.size > 0 DO BEGIN
      sei b erster Knoten aus B;

      FOR ALL x ∈ shallow[outer[b]] DO
        FOR ALL y ∈ deep[x] DO
          outer[y] := x;

      bestimme die Knoten p und q über die das Blossom outer[b] betreten und verlassen
wird;

      füge dasjenige Pfadstück des Blossoms outer[b] in P anstelle des Knotens b ein,
das gerade Länge hat;

    END ;

    FOR ALL f ∈ P DO matched[f] := notmatched[f];
    bestimme matching_edge für die Knoten des Pfades P;
    setze matched auf true für den Anfangs- und Endknoten von P;
```

END ;

Laufzeitanalyse. Wir werden zeigen, daß der oben angegebene Algorithmus eine Laufzeit von $\mathcal{O}(n^3)$ hat. Da wir ausgehend vom leeren Matching höchstens $n/2$ Augmentierungen durchführen, reicht es also zu zeigen, daß jede Phase $\mathcal{O}(n^2)$ Laufzeit benötigt. Jede Kante wird im Laufe einer Phase höchstens einmal in Q eingefügt. Die Routine "extend_tree" wird abgesehen vom Einfügen der Kanten in Q in konstanter Zeit ausgeführt. Die Routine "augment_path", die nur einmal am Ende einer Phase aufgerufen wird, hat $\mathcal{O}(m)$ Laufzeit. Es bleibt also die Laufzeit der Routine "shrink_blossom" zu untersuchen. Das Bestimmen der zu einem Blossom S gehörenden Knoten und Kanten benötigt lediglich $\mathcal{O}(|S|)$ Laufzeit. Das Berechnen von *deep* braucht höchstens $\mathcal{O}(n)$ Laufzeit. Da im Laufe einer Phase höchstens $\mathcal{O}(n)$ Blossoms auftreten können, ist damit die Gesamtlaufzeit durch $\mathcal{O}(n^2)$ beschränkt.

Der laufzeitkritische Teil in obigem Algorithmus ist die Aktualisierung des Feldes *deep*. Gabow und Tarjan [GT83] haben gezeigt, daß eine spezielle Variante des set-union Algorithmus hier anwendbar ist, die ein implizites updating des Feldes *deep* in $\mathcal{O}(m)$ ermöglicht. Damit ergibt sich eine Gesamtlaufzeit von $\mathcal{O}(nm)$ für den Matchingalgorithmus.

Der effizienteste Algorithmus zur Lösung des Matching-Problems stammt von Micali und Vazirani [MV80] und hat eine Laufzeit $\mathcal{O}(\sqrt{n}m)$, also die gleiche Laufzeit, die auch der Algorithmus von Hopcroft und Karp für bipartite Graphen erreicht. Eine vollständige Analyse dieses Algorithmus wurde erst kürzlich von Vazirani [Vaz94] publiziert. Ein einfacherer Algorithmus, der ebenfalls diese Laufzeit erreicht, stammt von Blum [Blu90].

4.3.3 Anwendungen

Das Briefträgerproblem. Der Chinese Mei-Ko [Mei62] stellte sich das Problem, den Weg eines Postboten zu minimieren, der alle Straßen einer Stadt ablaufen muß. Wenn man das Straßennetz der Stadt als Graph modelliert, bei dem die Knoten den Kreuzungen und die Kanten den Straßen entsprechen, so führt das zu dem Problem, in einem Graphen einen geschlossenen Kantenzug zu finden, der jede Kante des Graphen mindestens einmal abläuft (auch Chinese-Postman-Problem). In dem Fall, daß der zugrundeliegende Graph G Eulersch ist, haben wir dieses Problem bereits gelöst - ein Eulerscher Kantenzug durchläuft jede Kante genau einmal, was sicher bestmöglich ist. Ist G nicht Eulersch, so muß der Postbote offenbar mindestens eine Kante in G mehrfach ablaufen. Interpretiert man vom Postboten mehrfach abgelaufene Kanten in G als parallele Kanten oder Mehrfachkanten, so stellt jede Lösung des Postbotenproblems eine Eulertour in diesem Supergraphen $\tilde{G}$ von G dar. Offensichtlich wird keine Kante mehr als zweimal abgelaufen, denn sonst könnte man zwei dieser parallelen Kanten aus $\tilde{G}$ entfernen und es verbliebe noch immer ein Eulerscher Supergraph von G. Daher müssen in diesem Fall die Kanten von G zunächst so verdoppelt werden, daß der entstehende Supergraph $\tilde{G}$ Eulersch wird, oder äquivalent, daß in $\tilde{G}$ jeder Knoten (Kanten entsprechend ihrer Multiplizität gezählt) geraden Grad hat. Sei H der Subgraph von G gebildet aus den in G verdoppelten Kanten und $T \subseteq V$ die Menge der Knoten ungeraden Grades in G (beachte: $|T| \equiv 0 \pmod 2$). Dann gilt für die Grade in

H:

$$d_H(v) = \begin{cases} 1 & \text{falls } v \in T, \\ 0 \pmod 2 & \text{sonst.} \end{cases}$$

Ein solcher Subgraph H von G bzw. seine Kantenmenge heißt *T-Join*. Ein einfacher Induktionsschluß zeigt:

Lemma 4.3.7 *Jeder T-Join in einem Graphen G ist die kantendisjunkte Vereinigung von $s_i - t_i$−Pfaden in G, $1 \le i \le |T|/2$, so daß $\{s_1, \ldots, s_{|T|/2}, t_1, \ldots, t_{|T|/2}\} = T$.* □

Wenn durch $c : E \to \mathbb{Q}^+$ eine Längenfunktion auf den Kanten von G gegeben ist, dann suchen wir also nach einem T-Join minimalen Gewichts $\sum_{e \in H} c(e)$ in G.

Satz 4.3.8 [EJ73] *Das Problem* Minimum Weight T-Join *läßt sich in polynomieller Zeit auf das Problem* Minimum Weight Perfect Matching *in allgemeinen Graphen reduzieren.*

Beweis. Sei also $G = (V, E, c)$ ein kantengewichteter Graph und T eine gerade Teilmenge seiner Knotenmenge. Betrachte den Distanzgraphen $G_{dist}[T]$ zu G und T, der Knotenmenge T und Kantenmenge $\binom{T}{2}$ hat und in dem eine Kante $\{u, v\}$, $u, v \in T$, gewichtet ist durch das Gewicht $d_G(u, v)$ eines bezüglich c kürzesten $u - v$−Weges in G. Die Gewichte im Graphen $G_{dist}[T]$ lassen sich beispielsweise durch $|T|$-malige Anwendung des Algorithmus von Dijkstra in polynomieller Zeit berechnen. Ein perfektes Matching M^* minimalen Gewichts in $G_{dist}[T]$ definiert nun in naheliegender Weise ein $s_i - t_i$−Pfadsystem wie in obigem Lemma (die Pfade sind kantendisjunkt, weil es sonst offenbar ein perfektes Matching in $G_{dist}[T]$ mit noch kleinerem Gewicht gäbe) und damit einen T-Join J^* in G mit $c(J^*) = c(M^*)$. Da umgekehrt jeder T-Join ein perfektes Matching in $G_{dist}[T]$ definiert, dessen Gewicht höchstens so groß ist wie das des T-Joins, gibt es in G nach Wahl von M^* auch keinen T-Join kleineren Gewichts als J^*. □

Das Problem Minimum Weight Perfect Matching wiederum ist, wie gesehen, in polynomieller Zeit berechenbar.

Mehr zum Chinese Postman Problem in [Bar90a, EGL95]; Anwendungen finden sich bei [Bar90b].

Maximum gradbeschränkte Subgraphen. Das folgende Problem verallgemeinert das maximum Matching Problem.

Maximum Degree-Constrained Subgraph:

INSTANZ:	ein ungerichteter Graph $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$, und ein Vektor $b = (b_1, \ldots, b_n) \in \mathbb{N}^n$,
LÖSUNGEN:	b-beschränkte Subgraphen (oder b-Faktoren) von G, d.h. Subgraphen $S = (V, E_S)$ von G mit $d_S(v_i) \le b_i$, $i = 1, \ldots, n$,
ZIELFUNKTION:	$\lvert E_S \rvert \stackrel{!}{=} \max$.

Auch dieses Problem reduziert sich auf das (allgemeine) maximum Matching Problem. Zu $G = (V, E)$ konstruiere wie folgt einen Hilfsgraphen $H = (V_H, E_H)$. Ersetze zunächst jede Kante $\{v_i, v_j\} \in E$ durch einen Pfad (v_i, u_i, u_j, v_j). Sodann ersetze jeden Knoten $v_i \in V$ durch eine stabile Knotenmenge $V_i := \{v_i^{(1)}, \ldots, v_i^{(b_i)}\}$ der Kardinalität b_i, wobei

für jede Kante $\{v_i, v_j\} \in E$ der Knoten u_i (bzw. u_j) nun zu allen Knoten $v \in V_i$ (bzw. V_j) benachbart ist. Formal:

$$\begin{aligned} V_H &:= \bigcup_{i=1}^{n} V_i \ \cup \bigcup_{\{v_i,v_j\}\in E} \{u_i, u_j\} \\ E_H &:= \bigcup_{\{v_i,v_j\}\in E} [\{\{v, u_i\} : v \in V_i\} \cup \{\{u_i, u_j\}\} \cup \{\{u_j, v\} : v \in V_j\}]. \end{aligned}$$

Für eine Kante $e = \{v_i, v_j\} \in E$ bezeichne $E_H(e)$ die Kantenmenge in eckigen Klammern. $E_H(e)$ in H entspricht sozusagen der Kante e in G, nachdem die Knoten von V jeweils ver-(b_i)-facht wurden.

Satz 4.3.9 (TUTTE 1954b; SHILOACH 1981) *Sei $G = (V, E)$ ein Graph, $b = (b_1, \ldots, b_n) \in \mathbb{N}^n$, H der wie oben konstruierte Hilfsgraph und M^* ein maximum Matching in H. Dann ist der Subgraph $S = (V, E_S)$ von G definiert durch*

$$\forall e \in E: \qquad e \in E_S \ :\Leftrightarrow \ |E_H(e) \cap M^*| = 2.$$

ein maximum b-beschränkter Subgraph von G.

Beweis. Nach Konstruktion von H ist S b-beschränkt. Da das Matching M^* insbesondere maximal ist, gilt

$$\begin{array}{rrcll} & 1 & \leq & E_H(e) \cap M^* \ \leq \ 2 & \forall e \in E \\ \Rightarrow & |M^*| & \leq & |E| + |E_S| & \\ \Leftrightarrow & |E_S| & \geq & |M^*| - |E| & \end{array}$$

Dies ist aber auch schon bestmöglich: Sei $S_{OPT} = (V, E_{OPT})$ ein maximum b-beschränkter Subgraph von G. Dann induziet S_{OPT} ein Matching M in H der Größe

$$\begin{array}{rrcl} & |M| & \geq & 2|E_{OPT}| + |E \setminus E_{OPT}| \\ & & = & |E_{OPT}| + |E| \\ \Rightarrow & |M| - |E| & \geq & |E_{OPT}| \end{array}$$

Wegen $|M^*| \geq |M|$ ergeben die beiden jeweils letzten Ungleichungen hintereinander geschaltet $|E_S| \geq |E_{OPT}|$, d.h. E_S ist maximum. □

Übung 4.3.10 *Im kantengewichteten Fall ($w : E \to \mathbb{Q}^+$) ist die Maximierung von $w(E_S)$ äquivalent ist zur Maximierung von $w(M)$, M ein maximales Matching in H, wenn die Kanten von H durch $w(e_H) := w(e) \Leftrightarrow e_H \in E_H(e)$ gewichtet werden.*

Eine weitere wichtige Anwendung des allgemeinen Matchings ist die Berechnung kürzester Wege in kantengewichteten, ungerichteten Graphen mit nicht notwendig positiver Gewichtsfunktion $c : E \to \mathbb{Q}$, siehe [Jun94, Abschnitt 13.6].

Übungen

Übung 4.3.11 (CHUNGPAISAN) *Ein Baum T besitzt ein perfektes Matching genau dann, wenn $q(T - v) = 1$ für alle $v \in V$.*

Übung 4.3.12 a) *Ein Graph G habe ein perfektes Matching M. Eine Kante $e \in E \setminus M$ gehört zu einem (anderen) perfekten Matching von G genau dann, wenn es einen alternierenden Kreis in G gibt, der e enthält.*
b) *Sei $G = (U, V, E)$ ein bipartiter Graph. Falls es in G ein Matching gibt, das U überdeckt, und falls $d(u) \geq 2$ für alle $u \in U$ gilt, dann gibt es sogar zwei verschiedene solche Matchings.*

Übung 4.3.13 a) (MADER 1973) *Sei G ein zusammenhängender Graph mit genau einem perfekten Matching M. Dann enthält G eine Brücke e, die zwei ungerade Komponenten von $G-e$ verbindet (und also $e \in M$).* **b)** *Konstruiere für jedes $k \in \mathbb{N}$ einen Graphen G_k mit $\delta(G_k) \geq k$ und einem eindeutigen perfekten Matching.*

Übung 4.3.14 *Das Entscheidungsproblem* MATCHING*:*

INSTANZ: ein Graph G und ein $k \in \mathbb{N}$,
FRAGE: $\nu(G) \geq k$?

liegt in NP $\cap$ co-NP*.*

Übung 4.3.15 a) *Man folgere die* TUTTE-BERGE*-Formel (Satz 4.3.1) aus dem Satz von* TUTTE *(Korollar 4.3.3)* [*Hinweis:* Man wende den Satz von TUTTE auf einen entsprechend modifizierten Graphen an]. **b)** *Man folgere den Satz von* HALL *aus dem von* TUTTE*.*

Übung 4.3.16 a) *Ein faktorkritischer Graph enthält einen ungeraden Kreis.*
b) *Ein zusammenhängender Graph G ist faktorkritisch genau dann, wenn*

$$\nu(G-v) \;=\; \nu(G) \qquad \forall v \in V.$$

[*Hinweis* zur Rückrichtung: Um zu zeigen, daß ein maximum Matching in G nur höchstens einen Knoten nicht überdeckt, betrachte man nach Widerspruchsannahme ein maximum Matching M^* in G, das die folgende Größe minimiert: $d_M := \min\{dist(x,y) : x \text{ und } y \text{ sind von } M \text{ nicht überdeckte Knoten}\}$.]
c) (LOVÁSZ 1972c) *Ein Graph ist faktorkritisch genau dann, wenn er eine schwache Ohrenzerlegung* [vgl. Übung 2.3.11] *besitzt, in der der Kreis C und die Wege P_i sämtlich ungerade sind.*

Die folgenden zwei Übungen verallgemeinern den Satz 4.3.5 von PETERSEN.

Übung 4.3.17 *In einem kubischen, brückenlosen Graphen ist jede Kante in einem perfekten Matching enthalten.*

Übung 4.3.18 (BÄBLER 1938) *Sei $r \geq 3$ ungerade. Ein $(r-1)$-kantenzusammenhängender, r-regulärer Graph besitzt ein perfektes Matching.*

Übung 4.3.19 (CHARTRAND, POLIMENI, STEWART 1973) *Der Linegraph eines zusammenhängenden Graphen G hat genau dann einen 1-Faktor, wenn $|E(G)|$ gerade ist.*

Übung 4.3.20 *Zwei Spieler A und B spielen folgendes Spiel auf einem Graphen G. A besetzt zunächst einen Knoten, und dann besetzen beide abwechselnd jeweils einen noch unbesetzten Knoten, der dem vom Gegner im vorhergehenden Zug gewählten Knoten benachbart sein muß. Wer nicht mehr setzen kann, verliert. Wann hat Spieler A eine Gewinnstrategie?*

Sei $G = (\{1,\ldots,n\}, E)$ ein Graph. Für jede Kante $\{i,j\} \in E$, $i<j$, führe eine Variable x_{ij} ein. Dann heißt die schiefsymmetrische $n \times n$ Matrix $T(G) = (t_{ij})_{ij}$, definiert durch

$$t_{ij} := \begin{cases} x_{ij} & \text{falls } \{i,j\} \in E \text{ und } i<j, \\ -x_{ji} & \text{falls } \{i,j\} \in E \text{ und } i>j, \\ 0 & \text{sonst} \end{cases}$$

für alle $1 \leq i,j \leq n$, die TUTTE-Matrix von G. Aus der linearen Algebra ist bekannt, daß die Determinante einer Matrix ein Polynom in den Matrixeinträgen ist.

Übung 4.3.21 (TUTTE 1952) *Ein Graph G besitzt ein perfektes Matching genau dann, wenn $det(T(G)) \not\equiv 0$.*

Da ein Polynom, das nicht das Nullpolynom ist, nur an „wenigen“ Stellen verschwindet, führt dieses Ergebnis zu einem einfachen randomisierten Algorithmus, der mit großer Wahrscheinlichkeit das Entscheidungsproblem „besitzt G ein perfektes Matching?“ richtig beantwortet. Mit Hilfe dieses Ansatzes lassen sich randomisierte parallele Algorithmen entwickeln, um perfekte Matchings in Graphen zu konstruieren, siehe [MVV87] und [God93, Kapitel 7].

Übung 4.3.22 *Das maximum-Matching-Problem läßt sich in polynomieller Zeit auf das perfekte-Matching-Problem reduzieren.*

Kapitel 5

Färbung

5.1 Die chromatische Zahl

Unter einer *zulässigen k-Färbung* der Knoten eines Graphen $G = (V, E)$ versteht man eine Abbildung $c : V \to \{1, \ldots, k\}$, so daß gilt:

$$\{u, v\} \in E \;\Rightarrow\; c(u) \neq c(v),$$

d.h. benachbarte Knoten sind verschieden gefärbt. Wir meinen im folgenden immer zulässige (Knoten-)Färbungen, wenn wir von Färbungen eines Graphen sprechen. Ein Graph heißt *k-färbbar*, wenn er eine k-Färbung besitzt. Jeder Graph ist also n-färbbar, und ein Graph ist 1-färbbar genau dann, wenn er leer ist. Die *chromatische Zahl* $\chi(G)$ eines Graphen G ist definiert als das minimale $k \in \mathbb{N}$, so daß G k-färbbar ist. Ein Graph G heißt *k-chromatisch*, falls $\chi(G) = k$.

Beispiele und untere Schranken. Offenbar müssen die Knoten einer Clique in jeder zulässigen Färbung paarweise verschiedene Farben tragen. Es gilt also $\chi(K_n) = n$ und

$$\chi(G) \;\geq\; \omega(G). \tag{5.1}$$

I.A. gilt hier keine Gleichheit; so haben die ungeraden Kreise beispielsweise chromatische Zahl 3. In Abschnitt 5.3 werden wir sogar sehen, daß es zu jedem $k \in \mathbb{N}$ Graphen mit Cliquenzahl 2 und chromatischer Zahl k gibt. Ein Graph ist zweifärbbar genau dann, wenn er bipartit ist; Nach dem Satz 1.1.18 von KÖNIG gilt somit, daß $\chi(G) \geq 3$ dann und nur dann, wenn G einen ungeraden Kreis enthält, und daß 2-chromatische Graphen in linearer Zeit erkennbar und (optimal) färbbar sind.
Eine *Farbklasse* in einer Färbung von G ist eine Menge von Knoten, die dieselbe Farben tragen. Offenbar induziert jede Farbklasse einer Färbung von G eine stabile Menge in G. Die chromatische Zahl $\chi(G)$ eines Graphen G läßt sich also auch äquivalent definieren als die minimale Anzahl von stabilen Mengen, in die sich seine Knotenmenge partitionieren läßt, oder das minimale k, so daß G k-partit ist. Damit ist $\chi(G) = \theta(\overline{G})$. Da jede Farbklasse S_i in einer $\chi(G)$-Färbung eines Graphen G höchstens $\alpha(G)$ viele Knoten enthalten kann, gilt $n = \sum_{i=1}^{\chi(G)} |S_i| \leq \chi(G)\alpha(G)$, und wir erhalten die untere Schranke

$$\chi(G) \;\geq\; \frac{n}{\alpha(G)}, \tag{5.2}$$

die für den K_n oder den $K_{n,n}$ beispielsweise scharf ist. Die Graphenfamilie $\{G_k\}_{k\in N}$ von Seite 55 zeigt jedoch, daß die Schranke (5.2) i.a. beliebig schlecht werden kann: Es gilt: $\chi(G_k) = k+1$, während $\frac{n}{\alpha(G_k)} = 2$. In Kapitel 13 werden wir jedoch sehen, daß die Abschätzung (5.2) „fast immer gut“ ist.

Reduktionen. Bei der Untersuchung der chromatischen Zahl eines Graphen können wir uns auf zweifach zusammenhängende Graphen beschränken:

Proposition 5.1.1 $\chi(G) = \max\{\chi(B) : B \text{ ist Block von } G\}$.

Beweis. Betrachte den Block-Artikulationsgraphen $bc(G)$ (vgl. Satz 2.3.8). Färbe zunächst einen beliebigen Block B von G. Durchlaufe nun $bc(G)$ gemäß einer Breiten- oder Tiefensuche mit Startknoten B. In jedem neu entdeckten Block ist genau ein (Artikulations-) Knoten bereits gefärbt. Erweitere diese Färbung jeweils zu einer des ganzen Blockes. □

Anwendungen. Färbungsprobleme treten in vielen Situationen auf, z.B. bei der Terminplanung von Examensprüfungen, die man unter der Bedingung, daß jeder Student an den Prüfungen zu allen Fächern teilnehmen kann, die er belegt hat, auf möglichst wenige Zeitintervalle verteilen möchte. Oder bei der Terminplanung von Ausschußsitzungen des Bundestages, die man auf möglichst wenige Zeitintervalle so verteilen möchte, daß jeder Abgeordnete die Sitzungen aller Ausschüsse besuchen kann, denen er angehört. Im graphentheoretischen Modell G entspricht die Knotenmenge den Fächern (rsp. Ausschüssen), und zwei Knoten sind benachbart genau dann, wenn es Studenten gibt, die beide Fächer gleichzeitig belegt haben (bzw. Abgeordnete, die beiden Ausschüssen angehören). Eine stabile Menge von Fächern (rsp. Ausschüssen) kann dann ohne Konflikte auf denselben Termin gelegt werden, und das Problem, einen Terminplan mit minimal vielen Zeitintervallen zu finden, ist äquivalent dazu, eine $\chi(G)$-Färbung in G zu konstruieren.

In der Praxis sind jedoch die benötigten Ressourcen, wie Beaufsichtigungspersonal oder Raumkapazitäten in obigem Beispiel, beschränkt, und man möchte sie möglichst gleichmäßig auslasten. Das führt auf das Problem, Färbungen von Graphen zu finden, in denen die Farbklassen möglichst gleich groß sind (s.u.).

Graphfärbungsalgorithmen finden auch Anwendung beim Registerbelegungsproblem im Compilerbau, wo möglichst viele Variablen gleichzeitig in Registern gehalten werden sollen, siehe z.B. CHAITIN [Cha82].

Übungen und Anmerkungen

Ein Graph $G = (V, E)$ heißt kritisch k-färbbar genau dann, wenn gilt: $\chi(G) = k$ und $\chi(G - x) < k$ für alle $x \in (V \cup E)$.

Übung 5.1.2 **a)** *Jeder Graph G enthält einen kritisch $\chi(G)$-färbbaren Subgraphen.* **b)** *Für einen kritisch $\chi(G)$-färbbaren Graphen G gilt $\delta(G) \geq \chi(G) - 1$.* **c)** $m(G) \geq \binom{\chi(G)}{2}$. **d)** *Ein kritisch k-färbbarer Graph G ist $(k-1)$-fach kantenzusammenhängend* [Dir53]. [*Hinweis:* man nehme an, G ließe sich durch $k-2$ Kanten in zwei Teile $G[A]$ und $G[B]$ trennen, und setze eine $(k-1)$-Färbung von $G[A]$ auf B fort.]

Übung 5.1.3 [NG56] **a)** $\chi(G) \cdot \chi(\overline{G}) \geq n$; **b)** $\chi(G) + \chi(\overline{G}) \leq n + 1$. [*Hinweis* für b): Induktion über n]

Übung 5.1.4 **a)** *Die maximale Anzahl Knoten in einem Graphen G, die mit k Farben zulässig gefärbt werden können, ist $\alpha(G \times K_k)$* [Ber85].

b) *Das Entscheidungsproblem „gegeben ein Graph G und eine natürliche Zahl k – ist G k-färbbar?“ ist* NP*-vollständig* [Kar72].
c) $\chi(G \times H) = \max\{\chi(G), \chi(H)\}$ [Viz63, Abe64].

Übung 5.1.5 (Dirac 1952b) *Jede $\chi(G)$-Färbung eines Graphen $G = (V, E)$ liefert eine Partition von V in stabile Mengen (die Farbklassen). Ein Graph G heißt* eindeutig färbbar*, falls jede $\chi(G)$-Färbung von G dieselbe Partition von V induziert. Für einen eindeutig färbbaren Graphen G bzw. für seine $\chi(G)$-Färbung gilt:*
a) $\delta(G) \geq \chi(G) - 1$;
b) [CH68] *Die Vereinigung je zweier Farbklassen bildet einen zusammenhängenden Subgraphen;*
c) [CG69] *G ist $(\chi(G) - 1)$-fach zusammenhängend.*
d) [Jen96] *Es gibt keine eindeutig 3-färbbaren, kubischen Graphen.*

Eine Schnitt-Überdeckung eines Graphen $G = (V, E)$ ist eine Menge $\{S_1, \ldots, S_k\}$, $\emptyset \neq S_i \subset V$, $k \in \mathbb{N}$, so daß für alle $e \in E$ ein $i \in \{1, \ldots, k\}$ existiert mit $e \in \langle S_i, V \setminus S_i \rangle$. Offenbar bilden die Farbklassen einer optimalen Knotenfärbung von G eine Schnittüberdeckung von G der Größe $\chi(G)$.

Übung 5.1.6 *Jeder Graph besitzt eine Schnitt-Überdeckung der Größe $\lceil \log n \rceil$.*

5.2 Greedy-Färbung und der Satz von Brooks

Obwohl eins der meist-studierten Probleme der Graphentheorie, gibt es keine Formel oder polynomiell überprüfbare Charakterisierung für die chromatische Zahl (im Gegensatz zur Matching-Zahl beispielweise). In Abschnitt 5.4 werden wir sehen, daß das Problem, die chromatische Zahl eines Graphen zu berechnen, NP-schwer ist, so daß es vermutlich (d.h. falls P $\neq$ NP) keinen Algorithmus gibt, der für jeden Graphen die chromatische Zahl in polynomieller Zeit berechnet. Deshalb gewinnen Schranken für $\chi(G)$ an Bedeutung. Doch auch an Schranken für die chromatische Zahl heranzukommen ist schwierig. Beachte, daß die beiden unteren Schranken für $\chi(G)$ aus (5.1) und (5.2) vom algorithmischen Standpunkt aus nicht sehr hilfreich sind: die Berechnung der Parameter $\omega(G)$ oder $\alpha(G)$ ist selbst NP-schwer.

Angenommen, ein Algorithmus $\mathcal{A}$ konstruiert auf jedem Eingabegraphen eine zulässige Färbung. Dann ist die Anzahl der verwendeten Farben offenbar eine obere Schranke für $\chi(G)$. Wir untersuchen in diesem Abschnitt einige einfache Färbungsheuristiken und leiten daraus obere Schranken für die chromatische Zahl ab.

Zunächst untersuchen wir eine Greedy-Heuristik für das Graphen-Färbungsproblem. Sei $G = (V, E)$ ein Graph und eine beliebige Anordnung $\sigma : \{1, \ldots, n\} \to V$ seiner Knoten gegeben. Der Kürze halber setzen wir im folgenden $v_i := \sigma(i)$, $i = 1, \ldots, n$, und schreiben dann auch einfach $\sigma = (v_1, \ldots, v_n)$. Der Greedy-Färbungsalgorithmus geht die Knoten von G in der durch σ vorgegebenen Reihenfolge $v_1, \ldots, v_n$ durch und färbt jeden Knoten mit der niedrigsten zulässigen Zahl. Für $i = 1, \ldots, n$ setze $V_i := \{v_1, \ldots, v_i\}$ und $G_i := G[V_i]$.

```
FUNCTION GREEDY_FÄRBUNG (G,σ)    : INTEGER;

BEGIN
  c(v_1) := 1;
  FOR i := 2 TO n DO
    c(v_i) := min {k ∈ ℕ | k ≠ c(u) für alle u ∈ Γ(v_i) ∩ V_{i-1}};
  GREEDY_FÄRBUNG:= |{c(v): v ∈ V}|;
```

```
END;
```

Per definitionem liefert uns dieser einfache Algorithmus eine zulässige Knotenfärbung c.

Proposition 5.2.1 *Für jede Anordnung σ der Knoten eines Graphen G gilt*

$$\chi(G) \overset{(i)}{\leq} \text{GREEDY\_FÄRBUNG } (G,\sigma) \overset{(ii)}{\leq} \Delta(G)+1. \tag{5.3}$$

Der Algorithmus GREEDY_FÄRBUNG *kann mit Laufzeit $\mathcal{O}(n+m)$ implementiert werden.*

Beweis. In jedem Färbungsschritt können höchstens $|\Gamma(v_i) \cap V_{i-1}| \leq d(v_i) \leq \Delta(G)$ viele Farben für den Knoten v_i verboten sein. Also kommt GREEDY_FÄRBUNG unabhängig von σ stets mit den Farben $\{1,\ldots,\Delta(G)+1\}$ aus.

Was die Implementierung angeht, so ist die einzige Hürde wohl die Bestimmung von $c(v_i)$. Hierzu benutzt man ein zu Beginn auf 1 initialisiertes Feld $Zulässig : \{1,...,\Delta(G)+1\} \to \{0,1\}$, wobei $Zulässig[f]=1$ bedeuten soll, daß v_i mit der Farbe f zulässig gefärbt werden darf. Die Bestimmung von $c(v_i)$ besteht dann aus drei Phasen:

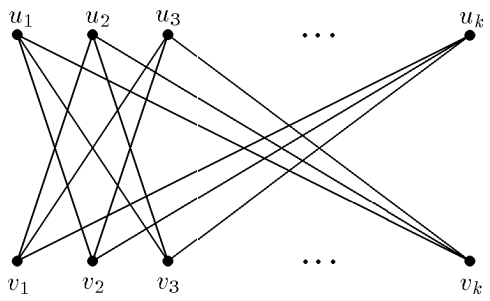
Phase I: die für den Knoten v_i verbotenen Farben werden im Feld $Zulässig$ mit 0 markiert (genau ein Durchgang durch die Adjazenzliste von v_i mit $\mathcal{O}(d(v_i))$ vielen Operationen);

Phase II: Suche der ersten 1 (wegen $\Delta(G)+1 > d(v_i)$ gibt mindestens eine) im Feld $Zulässig$ ($\mathcal{O}(d(v_i)+1)$ Operationen);

Phase III: Zurücksetzen des Feldes $Zulässig$ auf 1 (wieder ein Durchgang durch die Adjazenzliste von v_i mit $\mathcal{O}(d(v_i))$ vielen Operationen).

Summiert über alle Knoten ist der Aufwand für GREEDY_FÄRBUNG mithin $\mathcal{O}(n+m)$. □

Für ungerade Kreise und vollständige Graphen gilt Gleichheit in der Ungleichungskette (5.3). Wie der $K_{1,\Delta}$ zeigt, kann (unabhängig von σ) die Abschätzung (ii) beliebig schlecht werden, während Gleichheit in (i) gilt. Für die folgende Familie $\{B_n\}$ bipartiter Graphen gibt es andererseits eine Anordnung σ der Knoten, bei der (ii) eine Gleichheit und die Schranke (i) beliebig schlecht wird [Joh74b]. B_n hat $n=2k$, $k \geq 2$, viele Knoten und entsteht aus dem vollständig bipartiten Graphen $K_{k,k} = (U,V,E)$ durch Herausnahme des perfekten Matchings $M_k := \{\{u_i,v_i\} : i=1,\ldots,k\}$:



Obwohl B_n zweifärbbar ist, benötigt der Greedy-Algorithmus bei jeder Anordnung der Knoten, bei der auf u_i stets v_i folgt für $i=1,\ldots,k$,

$$k \;=\; \Delta(B_n)+1 \;=\; \Omega(n)$$

viele Farben. Der Greedy-Algorithmus liefert also i.a. beliebig schlechte Färbungen.[1]
Der Greedy-Algorithmus färbt u.U. sogar Bäume schlecht:

Übung 5.2.2 *Man gebe eine Familie $\{T_n\}_{n\in\mathbb{N}}$ von Bäumen an, so daß T_n n Knoten hat und der Greedy-Algorithmus auf T_n bei entsprechender Anordnung der Knoten $\Omega(\log n)$ viele Farben benötigt.*

Tatsächlich gibt es sogar Graphen, für die der Greedy-Algorithmus bei fast allen Anordnungen der Knotenmenge schlecht färbt:

Satz 5.2.3 (KUČERA 1991) *Seien $\epsilon, \delta > 0$ und $c < 1$ Konstanten. Dann gibt es für n groß genug Graphen G_n mit chromatischer Zahl $\chi(G_n) \leq n^\epsilon$, für die der Anteil aller Anordnungen der Knotenmenge, auf denen der Greedy-Algorithmus weniger als $c\frac{n}{\log n}$ Farben verwendet, $o(n^{-\delta})$ ist.* □

Es gilt aber:

Lemma 5.2.4 *Für jeden Graphen G gibt es eine Anordnung σ^* seiner Knoten, auf der der Greedy-Algorithmus eine optimale Färbung konstruiert, d.h. es gilt*

$$\chi(G) \;=\; \min_{\sigma\in\mathcal{S}_n} \textsc{Greedy\_Färbung}\ (G,\sigma).$$

Beweis. Seien $S_1,\ldots,S_{\chi(G)}$ die Farbklassen von G in einer $\chi(G)$-Färbung. Enthalte $\sigma^* = (v_1,\ldots,v_n)$ zunächst die Knoten von S_1 in beliebiger Reihenfolge, dann die von S_2 u.s.w. Es folgt $\textsc{Greedy\_Färbung}\ (G,\sigma^*) = \chi(G)$. □

Leider kennen wir jedoch die Anordnung σ^* im Voraus nicht. In Kapitel 13 werden wir sehen, daß GREEDY_FÄRBUNG „in aller Regel recht gute" Resultate liefert.

Eine naheliegende Idee, um an bessere Färbungen mit weniger Farben zu kommen, ist, zunächst eine „günstige" Anordnung $\sigma = (v_1,\ldots,v_n)$ der Knoten zu bestimmen und GREEDY_FÄRBUNG dann auf diese Anordnung anzuwenden. Für den i-ten Knoten v_i sind im Algorithmus GREEDY_FÄRBUNG höchstens $d_{G_i}(v_i)$ (der sogenannte Rückwärtsgrad von v_i) viele Farben verboten. Also kommt GREEDY_FÄRBUNG stets mit $1+\max_{1\leq i\leq n} d_{G_i}(v_i)$ vielen Farben aus. Wir schreiben im folgenden abkürzend

$$b(\sigma) := \max_{1\leq i\leq n} d_{G_i}(v_i), \qquad \text{falls } \sigma = (v_1,\ldots,v_n).$$

Durch Minimierung über alle $n!$ Anordnungen $\sigma\in\mathcal{S}_n$ der Knotenmenge erhält man damit folgende obere Schranke an $\chi(G)$.

$$\chi(G) \;\leq\; 1 + \min_{\sigma\in\mathcal{S}_n} b(\sigma).$$

Überraschenderweise läßt sich eine Anordnung der Knoten, bei der dieses Minimum angenommen wird, leicht konstruieren; betrachte dazu die folgende Prozedur.

```
FOR i := n DOWNTO 1 DO BEGIN
  wähle einen Knoten minimalen Grades in G als v_i;
```

[1] Der Greedy-Algorithmus verwendet für die Graphenfamilie B_n allerdings im Erwartungswert (d.h. gemittelt über alle Anordnungen der Knotenmenge) und asymptotisch (d.h. für $n\to\infty$) nur zwei Farben [Big90].

$G := G - v_i$;
END;

Jede Permutation der Knotenmenge eines Graphen, die man so erhalten kann, heißt smallest-last Anordnung. Es gilt nun die folgende interessante Maximum-Minimum-Beziehung.

Proposition 5.2.5 (Halin 1967b; Matula 1968; Finck, Sachs 1969) *Für jede smallest-last Anordnung σ_{SL} der Knotenmenge eines Graphen gilt:*

$$b(\sigma_{SL}) \;=\; \max_{H} \delta(H) \;=\; \min_{\sigma \in \mathcal{S}_n} b(\sigma),$$

wobei das Maximum über alle Subgraphen H von G genommen wird.

Beweis. Für jede smallest-last Anordnung σ_{SL} der Knotenmenge eines Graphen gilt

$$b(\sigma_{SL}) \;\leq\; \max_{i} \delta(G_i) \;\leq\; \max_{H} \delta(H).$$

Sei andererseits H^* ein Subgraph von G mit $\delta(H^*) = \max_H \delta(H)$. Dann gilt für jede Permutation σ der Knotenmenge von G, wenn $j = j(\sigma)$ den kleinsten Index bezeichnet, so daß H^* Subgraph von G_j ist:

$$\max_{H} \delta(H) \;=\; \delta(H^*) \;\leq\; d_{H^*}(v_j) \;\leq\; d_{G_j}(v_j) \;\leq\; b(\sigma).$$

Also gilt $\max_H \delta(H) \leq \min_{\sigma \in \mathcal{S}_n} b(\sigma)$. Die Ungleichung $\min_{\sigma \in \mathcal{S}_n} b(\sigma) \leq b(\sigma_{SL})$ schließlich ist trivial. □

Insbesondere läßt sich also in einem Graphen G ein Subgraph H von maximalem Minimalgrad in polynomieller Zeit berechnen. Die Färbungsheuristik Smallest-Last ordnet nun die Knoten eines Graphen zunächst gemäß einer smallest-last-Anordnung σ_{SL} an und wendet sodann Greedy_Färbung (G, σ_{SL}) an. Aus der letzten Proposition erhalten wir die folgende Schranke an die Anzahl der verwendeten Farben.

Korollar 5.2.6 (Szekeres, Wilf 1968) *Sei G ein Graph und σ_{SL} eine smallest-last Anordnung seiner Knotenmenge. Dann gilt*

$$\chi(G) \;\leq\; \text{Greedy\_Färbung } (G, \sigma_{SL}) \;\leq\; 1 + \max_{H} \delta(H), \tag{5.4}$$

wobei das Maximum über alle Subgraphen H von G genommen wird. □

Übung 5.2.7 a) (Matula, Beck 1983) *Die* Smallest-Last*-Heuristik kann mit Laufzeit $\mathcal{O}(n+m)$ implementiert werden.*
b) *Die* Smallest-Last*-Heuristik färbt alle Wälder optimal.*
c) *Für jede Anordnung $v_1, \ldots, v_n$ der Knoten eines Graphen G gilt $\max_i d_{G_i}(v_i) \geq m/n$.*
d) *Man gebe eine Folge bipartiter Graphen an, auf denen die Smallest-last-Heuristik $\Omega(n)$ viele Farben vergibt.*

Wie bereits der $K_{1,\Delta}$ (und die Familie B_k von oben) zeigte, kann die Schranke $\chi(G) \leq \Delta(G) + 1$ beliebig schlecht werden. Während sie für ungerade Kreise und vollständige Graphen scharf ist, kann sie für alle anderen Graphen immerhin um eins verbessert werden.

Korollar 5.2.8
Falls ein Graph G zusammenhängend, aber nicht $\Delta(G)$-regulär ist, so gilt $\chi(G) \leq \Delta(G)$.

Beweis. Die Aussage folgt einerseits unmittelbar aus dem Beweis von Gleichung (5.4) (Übung). Wir wollen sie jedoch im Hinblick auf den folgenden Satz noch auf eine andere Art beweisen. Sei v_1 ein Knoten in G mit $d(v_1) < \Delta(G)$. Numeriere die Knoten von G in einer Reihenfolge $v_1, \ldots, v_n$, in der eine Breitensuche auf G mit Startknoten v_1 die Knoten besucht hat. Greedy-färbe nun G in der Reihenfolge $\sigma = (v_n, \ldots, v_1)$. Da jeder Knoten $v \neq v_1$ einen Nachbarn mit kleinerer Nummer hat, hat v, wenn er an der Reihe ist, gefärbt zu werden, höchstens $\Delta(G) - 1$ bereits gefärbte Nachbarn. Also gilt GREEDY_FÄRBUNG $(G, \sigma) \leq \Delta(G)$. □

Den regulären Fall behandelt der folgende Satz.

Satz 5.2.9 (BROOKS 1941) *Sei G ein zusammenhängender Graph auf mindestens drei Knoten. Wenn G nicht vollständig und kein ungerader Kreis ist, dann gilt:*

$$\chi(G) \leq \Delta(G).$$

Beweis [Lov75a]. Angenommen, G ist nicht zweifach zusammenhängend. Falls ein Block von G regulär ist, so ist er nicht $\Delta(G)$-regulär (betachte einen Artikulationsknoten an der Nahtstelle zu einem anderen Block von G), und man kann den Greedy-Algorithmus zum Färben dieses Blockes benutzen; andernfalls verwendet man den Algorithmus aus Korollar 5.2.8. Man durchlaufe nun den Block-Artikulationsgraphen $bc(G)$ von G gemäß einer Breiten- oder Tiefensuche und passe die Färbungen auf den Blöcken einander an (vergleiche Proposition 5.1.1).

Sei G nun also o.B.d.A. zweifach zusammenhängend, regulär und weder ein Kreis noch vollständig. Wir verwenden ähnlich wie im Beweis von Korollar 5.2.8 den Greedy-Algorithmus zum Färben. Angenommen, G besitzt einen Knoten v_1, der zwei nicht-adjazente Nachbarn v_{n-1} und v_n hat, so daß $G - v_{n-1} - v_n$ noch zusammenhängend ist. Definiere $v_2, \ldots, v_{n-2}$ wie oben durch Breitensuche auf $G - v_{n-1} - v_n$ mit Startknoten v_1. Dann färbt der Greedy-Algorithmus auf der Ordnung $v_n, v_{n-1}, \ldots, v_1$ den Graphen G mit höchstens $\Delta(G)$ Farben, denn v_{n-1} und v_n erhalten dieselbe Farbe, so daß am Schluß mindestens eine Farbe für v_1 verbleibt. Das folgende Lemma schließt somit den Beweis. □

Lemma 5.2.10 *Sei ein Graph G 2-zusammenhängend und weder ein Kreis noch vollständig. Dann existieren Knoten x und y in V mit $dist(x, y) = 2$, so daß $G - x - y$ noch zusammenhängend ist.*

Beweis. Sei $v \in V$ ein Knoten mit $d(v) = \Delta(G)$. Dann ist $H := G[\{v\} \cup \Gamma(v)]$ nicht vollständig, denn sonst wäre entweder $G = H$ vollständig oder der Schnitt $\langle V(H), V(G) \setminus V(H) \rangle$ in G leer, im Widerspruch zur Voraussetzung. Also gibt es $\hat{x}$ und $\hat{y}$ in $\Gamma(v)$ mit $dist(\hat{x}, \hat{y}) = 2$. Ist $G - \hat{x} - \hat{y}$ noch zusammenhängend, so sind wir fertig. Andernfalls ist die Knotenmenge $\{\hat{x}, \hat{y}\}$ minimal trennend. Da G 2-zusammenhängend, aber kein Kreis ist, folgt $\Delta(G) \geq 3$ und damit $d(v) \geq 3$. Also enthält die Komponente C von $G - \hat{x} - \hat{y}$, die v enthält, noch weitere Knoten. Da v keine Artikulation in G ist, ist neben v noch ein weiterer Knoten $x \in C - v$ zu $\hat{x}$ oder zu $\hat{y}$ benachbart. Da die Menge $\{\hat{x}, \hat{y}\}$ minimal trennend ist, enthält jede Komponente von $G - \hat{x} - \hat{y}$ Nachbarn von $\hat{x}$ und $\hat{y}$; insbesondere gibt es also einen Knoten y mit $dist(x, y) = 2$ aus einer anderen Komponente von $G - \hat{x} - \hat{y}$

als C. Wir behaupten nun, daß $G-x-y$ noch zusammenhängend ist. Da $\hat{x}$ und $\hat{y}$ in $G-x-y$ über v durch einen Pfad verbunden sind, genügt es hierfür, zu zeigen, daß es in $G-x-y$ für alle Knoten einen Pfad nach $\hat{x}$ oder $\hat{y}$ gibt. Da aber x keine Artikulation von G ist, d.h. $G-x$ noch zusammenhängt, ist in $G-x$ jeder Knoten aus $C-x$ durch einen Pfad mit $\hat{x}$ oder $\hat{y}$ verbunden, der y nicht benutzt. Und da auch y keine Artikulation von G ist, d.h. $G-y$ noch zusammenhängt, ist in $G-y$ jeder Knoten aus $(V \setminus C)-y$ durch einen Pfad mit $\hat{x}$ oder $\hat{y}$ verbunden, der x nicht benutzt. □

Aus dem Beweis ergibt sich zudem (Übung):

Korollar 5.2.11 *Eine Färbung gemäß Satz 5.2.9 läßt sich in linearer Zeit konstruieren.*

Übungen und Anmerkungen

Wie bereits eingangs angemerkt, ist man in den Anwendungen daran interessiert, Graphen so zu färben, daß die Farbklassen möglichst gleich groß sind. Ein Graph G heißt *gleichmäßig k-färbbar* (engl. equitably k-colorable), wenn sich seine Knotenmenge $V(G)$ derart in k stabile Mengen $S_1, \ldots, S_k$ partitionieren läßt, daß $|S_i| - |S_j| \leq 1$ für alle $1 \leq i < j \leq k$. Jeder Graph G ist gleichmäßig $(\Delta(G)+1)$-färbbar [HS70] (einen Beweis findet man auch in [Bol78a, Chp. VI.5]), und man vermutet, daß außer den Graphen K_n, C_{2k+1} und $K_{2k+1,2k+1}$ jeder Graph sogar schon gleichmäßig $\Delta(G)$-färbbar ist, siehe [CLW94]. Die kleinste Zahl k, so daß ein Graph G gleichmäßig k-färbbar ist, heißt die *gleichmäßig chromatische Zahl* $\chi_e(G)$.

Übung 5.2.12 *Die gleichmäßig chromatische Zahl $\chi_e(G)$ läßt sich nicht durch eine Funktion in $\chi(G)$ beschränken.*

Vergleiche auch Übung 5.4.12.

Übung 5.2.13 [Wil67] *Der Greedy-Algorithmus verwendet höchstens $\lceil\sqrt{2m}\rceil$ viele Farben.* [*Hinweis:* Induktion nach m, entferne eine Farbklasse]

Für jede Anordnung σ der Knoten V eines Graphen kann GREEDY_FÄRBUNG im i-ten Schritt offenbar stets eine der Farben $1, \ldots, \min\{i, d(\sigma(i))+1\}$ vergeben und kommt daher sicherlich mit

$$\text{GREEDY\_FÄRBUNG}(G,\sigma) \leq \max_{1\leq i\leq n} \min\{i, d(\sigma(i))+1\} \tag{5.5}$$

(und oft mit weniger) vielen Farben aus.

Übung 5.2.14 (Largest-first-Heuristik) (WELSH, POWELL 1967)
Die Zahl $1+\max_{1\leq i\leq n} \min\{i-1, d(\sigma(i))\}$ wird minimal für eine Anordnung $\sigma : \{1,\ldots,n\} \to V$ der Knoten in Reihenfolge nicht zunehmender Knotengrade $d(\sigma(1)) \geq \cdots \geq d(\sigma(n))$. Man vergleiche diese Schranke mit denen aus dem Text.

Diese Aussage ist cum grano salis zu genießen: Eine Anordnung der Knoten in Reihenfolge nicht zunehmender Knotengrade minimiert zwar die obere Schranke für $\chi(G)$ aus (5.5) - doch, wie SYSLO [Sys89] bemerkte, verbessert diese Heuristik nicht notwendig die Ergebnisse der Greedy-Färbung. So gibt es Graphen, die GREEDY_FÄRBUNG auf *keiner* Anordnung σ_{deg} der Knoten in Reihenfolge nicht zunehmender Knotengrade mehr optimal färbt - im Gegensatz zu Lemma 5.2.4. Ein extremes Beispiel erhält man durch leichte Abänderung der Graphen B_k aus dem Text:

Übung 5.2.15 *Es gibt eine Familie $\{\hat{B}_k\}_{k\geq 2}$ bipartiter Graphen, so daß gilt:*

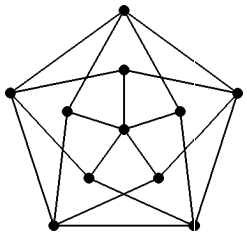
$$\text{GREEDY\_FÄRBUNG}(\hat{B}_k, \sigma_{deg}) = \Omega(|V(\hat{B}_k)|)$$

auf jeder Anordnung σ_{deg} der Knoten von $\hat{B}_k$ in Reihenfolge nicht zunehmender Knotengrade.

5.3 Chromatische Zahl und Taillenweite

Wir haben oben die triviale Beziehung $\chi(G) \geq \omega(G)$ notiert. Bereits der C_5 zeigt, daß hier nicht notwendig Gleichheit zu gelten braucht. Wir werden nun sehen, daß auch diese Schranke beliebig schlecht werden kann und sich die chromatische Zahl i.a. durch keine Funktion der Cliquenzahl nach oben beschränken läßt.

Unter den ersten, die zu jedem k einen Graphen mit chromatischer Zahl k, aber ohne kurze Kreise konstruierten, waren TUTTE [Des47, Des48, Des54], ZYKOV [Zyk49], sowie KELLY und KELLY [KK54]. Die Knotenzahl dieser Graphen wächst jedoch sehr schnell mit k. Ein einfaches, noch relativ „sparsames" Verfahren zur Konstruktion einer Familie $\{M_k : k \geq 3\}$ dreiecksfreier k-chromatischer Graphen hat MYCIELSKI angegeben. Die Graphen M_k sind wie folgt definiert. Der Graph $M_3 := C_5$ ist offenbar dreiecksfrei und 3-chromatisch. Seien v_i, $i = 1, \ldots, n$, die Knoten von M_k. Den Graphen M_{k+1} erhält man aus M_k, indem man neue Knoten $\{u_1, \ldots, u_n, w\}$, hinzufügt und folgendermaßen verbindet: w wird vollständig mit $\{u_1, \ldots, u_n\}$ verbunden sowie u_i jeweils vollständig mit $\Gamma(v_i)$, $i = 1, \ldots, n$. Der Graph M_4:



heißt auch GRÖTZSCH-Graph und ist hinsichtlich der Ordnung der eindeutige, kleinste, 4-chromatische, dreiecksfreie Graph [Chv73].

Proposition 5.3.1 (MYCIELSKI 1955)
Die Graphen M_k sind dreiecksfrei, k-chromatisch und haben $3 \cdot 2^{k-2} - 1$ Knoten.

Beweis. durch Induktion nach k. Für $k = 3$ ist die Aussage offenbar richtig. Der Graph M_{k+1} nun hat die angegebene Knotenzahl und ist wie schon M_k dreiecksfrei: da die Knoten $\{u_1, \ldots, u_n\}$ eine stabile Menge bilden, gehört der Knoten w zu keinem Dreieck; aufgrund der Dreiecksfreiheit von M_k sind die Mengen $\Gamma(v_i)$, $i = 1, \ldots, n$, stabil. Also liegt auch keiner der Knoten u_i in einem Dreieck.
$\chi(M_{k+1}) \leq k + 1$: Sei eine k-Färbung c von M_k gegeben. Erweitere diese auf M_{k+1} durch $c(u_i) := c(v_i)$, $i = 1, \ldots, n$, sowie durch $c(w) := k + 1$.
$\chi(M_{k+1}) \geq k + 1$: Angenommen, es gäbe eine k-Färbung c von M_{k+1}. Sei $M_{k+1}[\{v_1, \ldots, v_n\}]$ die Kopie von M_k in M_{k+1}. O.B.d.A. sei $c(w) = k$. Dann ist also keiner der Knoten $u_1, \ldots, u_n$ mit Farbe k gefärbt. Offensichtlich erhalten wir nun eine neue, zulässige k-Färbung von M_{k+1}, wenn wir jeden Knoten v_i mit $c(v_i) = k$ mit der Farbe $c(u_i)$ färben. Diese induziert aber eine $(k-1)$-Färbung von M_k, im Widerspruch zu $\chi(M_k) = k$. □

Bezeichne $n(k)$ die minimale Anzahl Knoten eines k-chromatischen dreiecksfreien Graphen. ERDŐS [Erd59, Erd61] konnte mit Hilfe probabilistischer Argumente $n(k) = \mathcal{O}\left((k \log k)^2\right)$ zeigen. Eine untere Schranke von $\Omega(k^2 \log k)$ gaben ERDŐS und HAJNAL [EH85].

Man kann sich weiter fragen, wie „dünn“ (gemessen an der Taillenweite) Graphen hoher chromatischer Zahl sein können. 3-chromatische Graphen können, wie schon die ungeraden Kreise zeigen, beliebig dünn werden.

Übung 5.3.2 (DESCARTES 1947, 1948, 1954) *Die folgende Konstruktionsvorschrift liefert Graphen T_k, $k \geq 2$, mit chromatischer Zahl k und Taillenweite $g(T_k) = 6$. Definiere $T_2 := C_6$. T_{k+1} erhält man aus T_k wie folgt. Sei $n_k := |V(T_k)|$, $r := (n_k - 1)k + 1$ und $\mathcal{R}$ eine Menge von r Knoten. Für jede n_k-elementige Teilmenge $U \subset \mathcal{R}$ nehme man nun eine (neue) Kopie von $T_k = (V, E)$ her und füge zwischen U und V derart n_k viele Kanten ein, daß diese Kanten eine Bijektion zwischen U und V bilden (T_{k+1} hat also $r + \binom{r}{n_k} n_k$ viele Knoten).*

Es schien schwierig, dies zu verbessern. Mit der probabilistischen Methode, die wir nun kennenlernen wollen, konnte ERDŐS überraschend einfach die Existenz dünner Graphen beliebig hoher chromatischer Zahl nachweisen. Die chromatische Zahl eines Graphen kann also groß sein unabhängig davon, daß der Graph lokal aussieht wie ein Baum. Dies ist der Grund, warum man die chromatische Zahl eines Graphen auch als einen globalen Graphenparameter bezeichnet (die Cliquenzahl hingegen als lokalen).

Satz 5.3.3 (ERDŐS 1959) *Für je zwei natürlichen Zahlen $k, g \geq 3$ gibt es einen Graphen $G(k, g)$ mit $\chi(G) \geq k$ und $g(G) \geq g$.*

Die probabilistische Methode (ERDŐS 1947). Die Existenz diskreter Strukturen läßt sich oftmals sehr einfach durch elementare wahrscheinlichkeitstheoretische Argumente beweisen. Tatsächlich sind derartige Beweise häufig lediglich Abzählargumente, doch lassen sie sich eleganter in der Sprache der Wahrscheinlichkeitstheorie formulieren. Die Grundidee der probabilistischen Methode ist, eine Wahrscheinlichkeitsverteilung auf den gesuchten Strukturen zu definieren. Wenn man dann nachweist, daß die Wahrscheinlichkeit für das Auftreten einer solchen Struktur > 0 ist, so weiß man also insbesondere, daß es mindestens eine solche Struktur gibt.

Beweis von Satz 5.3.3 [ASE92]:
Sei a eine reelle Zahl mit $0 < a < 1/g$ und $G \in \mathcal{G}_{n,p}$ ein zufälliger Graph auf n Knoten mit Kantenwahrscheinlichkeit $p = n^{a-1}$. Sei X die Zufallsvariable, die die Anzahl Kreise in G der Länge höchstens g zählt. Eine beliebige Folge von i paarweise verschiedenen Knoten von G bildet offenbar mit Wahrscheinlichkeit p^i einen Kreis in G. Wieviele solcher Folgen gibt es? Man hat n Möglichkeiten, den ersten Knoten zu wählen, $n-1$ für den zweiten, usw. Auf diese Weise hat man jeden potentiellen Kreis der Länge i genau $2i$ mal gezählt (von jedem Knoten in jede der zwei Richtungen einmal). Mithin gilt für den Erwartungswert von X:

$$E(X) \;=\; \sum_{i=3}^{g} \frac{n(n-1)\cdots(n-i+1)}{2i}\, p^i \;\leq\; \sum_{i=3}^{g} \frac{n^{ai}}{2i} \;=\; o(n) \qquad (\text{für } n \to \infty),$$

da $ai \leq ag < 1$ und g konstant. Insbesondere gilt also nach der MARKOFFschen Ungleichung A.1.3

$$Prob(X \geq n/2) \;=\; o(1) \qquad (\text{für } n \to \infty),$$

d.h. fast alle Graphen in $\mathcal{G}_{n,p}$ haben weniger als $n/2$ viele Kreise der Länge höchstens g (Eigenschaft 1).

Sei $b = b(n)$ eine natürliche Zahl, $1 \leq b \leq n$. Eine b-elementige Knotenmenge in G induziert höchstens $\binom{b}{2}$ viele Kanten. Da jedes $\{u,v\} \in \binom{V}{2}$ mit Wahrscheinlichkeit $1-p$ keine Kante in G ist, ist jede solche Knotenmenge $S \in \binom{V}{b}$ in G stabil mit Wahrscheinlichkeit $(1-p)^{\binom{b}{2}}$. Also gilt

$$\begin{aligned} Prob\,(\alpha(G) \geq b) &= Prob\,(\exists S \in \binom{V}{b} : G[S] \text{ stabil}) \\ &= Prob\,(\bigvee_{S \in \binom{V}{b}} G[S] \text{ stabil}) \\ &\leq \sum_{S \in \binom{V}{b}} Prob\,(G[S] \text{ stabil}) \\ &= \binom{n}{b} (1-p)^{\binom{b}{2}}. \end{aligned}$$

Wegen $\binom{n}{b} \leq n^b$ und $1 - x < e^{-x}$ für $x > 0$ folgt somit

$$Prob\,(\alpha(G) \geq b) \;<\; \left[n \cdot e^{-p(b-1)/2}\right]^b.$$

Setzt man nun

$$b := \left\lceil \frac{3 \ln n}{p} \right\rceil,$$

so ist der Ausdruck auf der rechten Seite $o(1)$, wie man durch Einsetzen sofort bestätigt. D.h. fast alle Graphen in $\mathcal{G}_{n,p}$ gilt $\alpha(G) < \frac{3 \ln n}{p}$ (Eigenschaft 2).

Mithin gibt es für jedes n groß genug einen Graphen G_n^* auf n Knoten, der die beiden Eigenschaften 1 und 2 vereinigt. Entfernt man bei einem solchen Graphen G_n^* aus jedem der weniger als $n/2$ vielen Kreise der Länge höchstens g einen Knoten, so verbleibt ein Graph G_n' auf mindestens $n/2$ vielen Knoten mit Taillenweite größer als g und Unabhängigkeitszahl $\alpha(G_n') \leq \alpha(G_n^*) < \frac{3 \ln n}{p}$. Seine chromatische Zahl ist folglich

$$\chi(G_n') \;\geq\; \frac{n/2}{\alpha(G_n')} \;>\; \frac{n/2}{(3 \ln n)/p} \;=\; \frac{n^{a-1} n/2}{3 \ln n} \;=\; \frac{n^a}{6 \ln n},$$

was wegen $a > 0$ für n groß genug sicherlich größer als jedes vorgegebene k ist. □

Anmerkungen und Übungen

Es war lange Zeit ein offenes Problem, Graphen, wie sie durch Satz 5.3.3 garantiert werden, auch zu konstruieren. Lovász [Lov68] gab schließlich eine Konstruktion an, die allerdings essentiell Hypergraphen benutzte (siehe auch [NR79]). Erst 1986 gelang Kříž [Kri89] eine Hypergraphen-freie Konstruktion solcher Graphen. Er muß jedoch zugeben, daß auch die Ordnung der von ihm konstruierten Graphen nicht primitiv rekursiv in g ist, während eine sorgfältige Analyse des obigen Beweises zeigt, daß es schon Graphen G der Ordnung höchstens $\lceil (6k \log k)^{g+1} \rceil$ gibt mit chromatischer Zahl größer als k und Taillenweite größer als g (siehe z.B. [Bol91]). Lubotzky, Phillips und Sarnak [LPS88] gelang schließlich in einer bahnbrechenden Arbeit zu sogenannten „Expander-Graphen“ die explizite Konstruktion derartiger Graphen mit Ordnung $n \leq \chi(G)^{3g}$.

Übung 5.3.4 $n \;\geq\; (\chi(G) - 2)^{g(G)/2-1}$. [*Hinweis:* zähle die Knoten im Umkreis eines festen Knotens und benutze Korollar 5.2.6]

Übung 5.3.5 *Der* KNESER-*Graph $K^{(r)}_{2r+k}$ für natürliche Zahlen r und k hat als Knotenmenge die r-elementigen Teilmengen von $\{1,\ldots,2r+k\}$, und zwei Knoten sind in $K^{(r)}_{2r+k}$ genau dann durch eine Kante verbunden, wenn die entsprechenden Teilmengen disjunkt sind.*
a) *Der $K^{(2)}_5$ ist der* PETERSEN-*Graph.* **b)** *Für alle $r,k \in \mathbb{N}$ gilt $\chi(K^{(r)}_{2r+k}) \leq k+2$.*
Für $r=1$ wird offensichtlich $K^{(r)}_{2r+k} = K_{k+2}$, so daß in b) im Fall $r=1$ Gleichheit gilt.
c) *Man zeige, daß auch für $k=1$ und alle $r \in \mathbb{N}$ Gleichheit in b) gilt.*
KNESER *[Kne55] vermutete, daß immer $\chi(K^{(r)}_{2r+k}) = k+2$ gilt – dies konnte jedoch erst über zwanzig Jahre später von* LOVÁSZ *bewiesen werden [Lov78] (siehe auch [Bol78a]). Für $k \in \mathbb{N}$ fest und wachsendes r werden auch die $(k+2)$-chromatischen* KNESER-*Graphen $K^{(r)}_{2r+k}$ „sehr dünn", gemessen (zwar nicht an ihrer Taillenweite, aber) an ihrer „Dichte". Es bezeichne $d(K^{(r)}_{2r+k})$ den Grad eines (jeden) Knotens von $K^{(r)}_{2r+k}$.* **d)** *Es gilt $\lim_{r\to\infty} \frac{d(K^{(r)}_{2r+k})}{n(K^{(r)}_{2r+k})} \leq 2^{-r}$.*

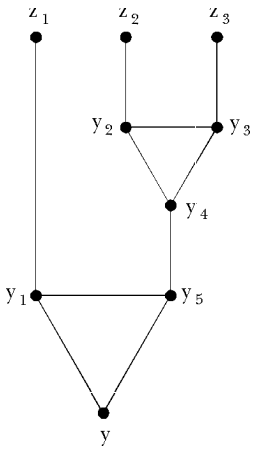
5.4 Die Komplexität des Färbungsproblems

Die Ergebnisses des letzten Abschnittes lassen erahnen, daß das Färbungsproblem „schwierig" ist. So ist denn auch – wie bereits in Übung 5.1.4 gesehen – das Problem, die chromatische Zahl $\chi(G)$ eines Graphen G zu bestimmen, NP-schwer. Dieser Abschnitt wird zeigen (unabhängig von jener Übung), daß das Färbungsproblem selbst in äußerst speziellen Situationen NP-schwer bleibt.

Beim Entscheidungsproblem k-COLORABILITY, $k \in \mathbb{N}$ fest, wird danach gefragt, ob ein als Eingabe gegebener Graph G k-färbbar ist, ob also $\chi(G) \leq k$ gilt. Während bipartite Graphen in linearer Zeit erkannt (und gefärbt) werden können (vergleiche Abschnitt 1.3.1) und somit 2-COLORABILITY $\in$ P ist, gilt schon für $k=3$:

Satz 5.4.1 (STOCKMEYER 1973) *3-*COLORABILITY *ist* NP*-vollständig.*

Beweis [Gib85]. Das naheliegende Zertifikat für 3-Färbbarkeit, eine 3-Färbung eines 3-färbbaren Graphen, kann man offenbar in Zeit $\mathcal{O}(m)$ überprüfen, so daß das Problem 3-COLORABILITY in NP liegt. Um zu zeigen, daß es auch NP-schwer ist, reduzieren wir das NP-vollständige Entscheidungsproblem 3-SAT auf 3-COLORABILITY. Sei $F = \bigwedge_{j=1}^{m} C_j$ eine Instanz von 3-SAT, d.h. eine Konjunktion von m Klauseln C_j über Literalen $x_1,\ldots,x_n,\overline{x}_1,\ldots,\overline{x}_n$, so daß jede Klausel C_j genau drei (o.B.d.A. verschiedene) Literale enthält. Zu jeder solchen Booleschen Formel F wird nun ein Graph $G = G(F)$ konstruiert, der genau dann 3-färbbar ist, wenn F erfüllbar ist. Betrachte den Graphen H:

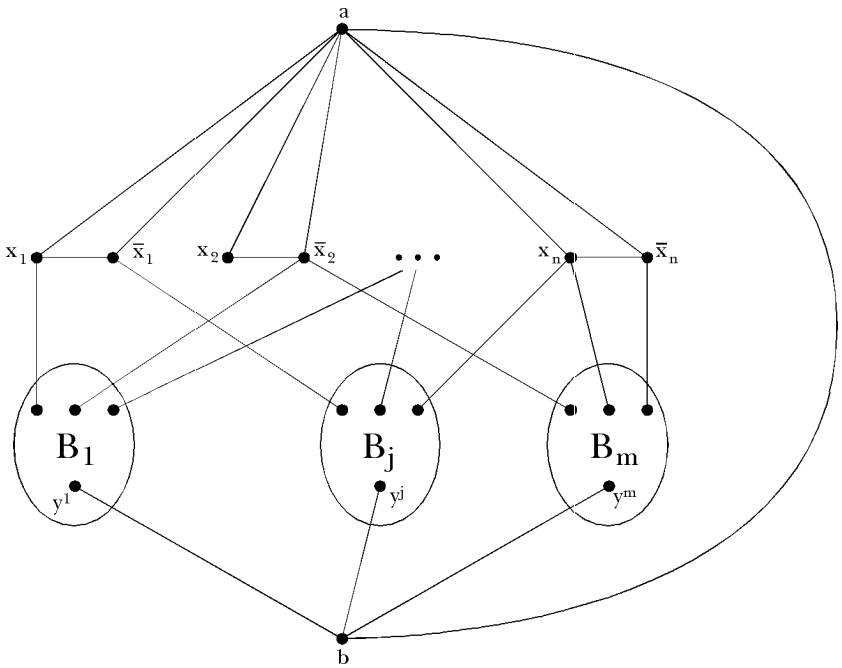


mit den „Eingängen“ z_1, z_2 und z_3 und dem „Ausgang“ y. Wie man sich leicht überzeugt, erfüllt der Graph H die folgenden zwei Eigenschaften.

(i) Sei eine 3-Färbung der Knoten z_1, z_2 und z_3 von H gegeben. Falls dann einer der Knoten z_1, z_2 oder z_3 mit der Farbe 1 gefärbt ist, dann läßt sich diese Färbung so auf ganz H fortsetzen, daß y ebenso mit Farbe 1 gefärbt ist.

(ii) Für jede zulässige 3-Färbung von H gilt: Falls die Knoten z_1, z_2 und z_3 alle mit derselben Farbe gefärbt sind, dann trägt auch y diese Farbe.

Die Reduktion benutzt den Graphen $B := H[\{y, y_1, \ldots, y_5\}]$ als Baustein. Der zur Instanz F konstruierte Graph $G(F)$ enthält Knoten $a, x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n$, wobei für $i \in \{1, \ldots, n\}$ die Knoten $\{x_i, \overline{x}_i, a\}$ ein Dreieck bilden. Sodann enthält $G(F)$ für jede Klausel C_j eine Kopie B_j von B, wobei jeder Knoten y_i^j, $i = 1, 2, 3$, der Kopie B_j statt mit dem Knoten z_i jeweils mit dem Knoten in $G(F)$ verbunden ist, der zum i-ten in der Klausel C_j auftretenden Literal korrespondiert. Der Ausgang y^j der Kopie B_j von B ist mit einem weiteren Knoten b verbunden. Schließlich ist der Knoten b noch mit dem Knoten a verbunden. Die folgende Abbildung zeigt ein Beispiel, in dem $C_m = \overline{x}_2 \vee x_n \vee \overline{x}_n$ ist.



Angenommen nun, F sei erfüllbar. Färbe die Knoten a und b mit den Farben 2 bzw. 0 und die Knoten $x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n$ gemäß einer erfüllenden Belegung von F mit den Farben 0 bzw. 1 (für *falsch* bzw. *wahr*). Da jede Klausel C_j mindestens einen wahren Literal enthält, ist für jeden Baustein B_j mindestens ein Eingang mit Farbe 1 gefärbt, so daß nach (i) auch y^j jeweils mit 1 gefärbt werden kann. Die erhaltene Färbung ist offensichtlich zulässig.
Sei umgekehrt eine (zulässige) 3-Färbung $c : V \to \{0, 1, 2\}$ von G gegeben, wobei o.B.d.A. $c(a) = 2$ und $c(b) = 0$ sei. Dann sind die Knoten $x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n$ in G offenbar mit den Farben 0 und 1 gefärbt, was aufgrund der Kanten $\{x_i, \overline{x}_i\}$, $1 \leq i \leq n$, eine Wertebelegung der Booleschen Variablen von F darstellt. Angenommen, diese Belegung erfüllte F nicht, d.h. es gäbe eine Klausel C_j, in der alle drei Literale den Wert 0 erhalten haben. Dann muß nach (ii) der Knoten y^j mit Farbe 0 gefärbt sein im Widerspruch zur Zulässigkeit der Färbung auf der Kante $\{y^j, b\}$. □

Obiger Satz läßt sich leicht auf jedes beliebige (feste) $k \geq 3$ verallgemeinern:

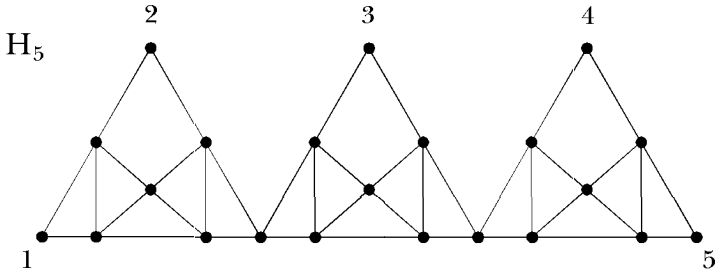
Übung 5.4.2 *Für jedes (feste) $k \geq 3$ ist* k-COLORABILITY NP-*vollständig.*

Unter der Hypothese P $\neq$ NP gibt es also für kein $k \geq 3$ einen effizienten Algorithmus, um k-partite Graphen zu erkennen. Beachte, daß das Problem CLIQUE – im Gegensatz zu k-COLORABILITY – für feste Cliquengröße k durch vollständige Enumeration in polynomieller Zeit berechenbar ist.

Das Problem 3-COLORABILITY bleibt selbst bei Einschränkung auf Graphen vom Maximalgrad 4 NP-vollständig:

Satz 5.4.3 [GJS76] *Das Problem* 3-COLORABILITY ($\Delta \leq 4$) *ist* NP-*vollständig.*

Beweis. Durch Reduktion von 3-COLORABILITY. Sei $G = (V, E)$ ein Graph. Wir ersetzen jeden Knoten $v \in V$ durch einen Baustein H_k mit $k = d(v)$ Anschlüssen. Die folgende Abbildung zeigt beispielsweise den Baustein H_5:



Die in G mit v inzidierenden Kanten werden an die k „Anschlüsse“ (= Knoten vom Grad 2) von H_k „angelegt“, so daß der resultierende Graph G' Maximalgrad 4 hat. Da jeder Baustein H_k 3-chromatisch ist, und in jeder Dreifärbung von H_k wiederum alle Anschlüsse dieselbe Farbe erhalten, ist G 3-färbbar genau dann, wenn G' 3-färbbar ist. □

Die Gradbeschränkung ist bestmöglich:

Übung 5.4.4 *Das Problem* 3-COLORABILITY ($\Delta \leq 3$) *ist polynomiell lösbar.*

Hajós-Konstruktion und Taillenweite. Um zu zeigen, daß k-COLORABILITY auch auf Graphen mit großer Taillenweite NP-vollständig bleibt, benötigen wir die sogenannte HAJÓS-*Konstruktion.*

Definition 5.4.5 (HAJÓS 1961) *Sei $k \in \mathbb{N}$. Ein Graph G heißt* k-konstruierbar, *falls*

- *entweder $G \cong K_k$*
- *oder G geht aus einer der beiden folgenden Operationen hervor:*
  - [HAJÓS Konstruktion] *es gibt zwei disjunkte k-konstruierbare Graphen H_1 und H_2 und Kanten $\{x_1, y_1\} \in E(H_1)$ und $\{x_2, y_2\} \in E(H_2)$, so daß man G aus $H_1 - \{x_1, y_1\}$ und $H_2 - \{x_2, y_2\}$ erhält, indem man die Knoten x_1 und x_2 identifiziert und die Kante $\{y_1, y_2\}$ einfügt;*
  - [HAJÓS Identifikation] *es gibt einen k-konstruierbaren Graphen H und zwei nicht benachbarte Knoten x und y in H, so daß man G aus H erhält, indem man die Knoten x und y identifiziert und etwaige Mehrfachkanten entfernt.*

Satz 5.4.6 (HAJÓS 1961) *Sei G ein Graph. Dann gilt $\chi(G) \geq k$ genau dann, wenn G einen k-konstruierbaren Subgraphen enthält. Insbesondere ist jeder kritisch k-chromatische Graph k-konstruierbar.*

Beweis. „$\Leftarrow$“: Übung. „$\Rightarrow$“: Angenommen, es gäbe einen Graphen G mit $\chi(G) \geq k$, aber ohne einen k-konstruierbaren Subgraphen. Sei $G = (V, E)$ o.B.d.A. kantenmaximal mit dieser Eigenschaft, d.h. wenn immer man G eine Kante e hinzufügt, so enthalte $G + e$ einen k-konstruierbaren Subgraphen H_e.

Fall 1: $\forall x, y, z \in V : (\{x, y\} \notin E \wedge \{x, y\} \notin E \Rightarrow \{x, z\} \notin E)$

In diesem Fall ist also Nicht-Adjazenz eine Äquivalenzrelation auf V. Wegen $\chi(G) \geq k$ gibt es mindestens k Äquivalenzklassen. Dann enthält G aber einen K_k-Subgraphen.

Fall 2: $\exists x, y, z \in V : (\{x, y\} \notin E \wedge \{x, y\} \notin E \wedge \{x, z\} \in E)$

Nach Wahl von G enthalten $G + \{x, y\}$ und $G + \{y, z\}$ k-konstruierbare Subgraphen H_{xy} bzw. H_{yz}. Offenbar enthält der Graph H_{xy} eine Kopie der Kante $\{x, y\}$ und H_{yz} eine Kopie von $\{y, z\}$. Lösche aus der disjunkten Vereinigung der Graphen H_{xy} und H_{yz} diese beiden Kanten, identifiziere die Kopie von y in H_{xy} mit der in H_{yz}, füge zwischen den Kopien der Knoten x und z in H_{xy} bzw. H_{yz} eine Kante ein und erhalte so einen k-konstruierbaren Graphen H'. Wenn man nun noch für jeden Knoten $v \in V(G) - y$, der sowohl in H_{xy} als auch in H_{yz} eine Kopie besitzt, diese beiden Kopien identifiziert, erhält man einen k-konstruierbaren Subgraphen H von G. □

Der folgende Satz verallgemeinert eine Bemerkung aus [JT95, Abschnitt 10.3].

Satz 5.4.7 *Sei $k \geq 3$ fest und $c := 1/(2 + 3\log(k+1))$. Dann gilt: k-*COLORABILITY, *eingeschränkt auf die Menge aller Graphen G mit Taillenweite $g(G) \geq c \log n$ ist* NP-*vollständig.*

Beweis. Wir reduzieren von k-COLORABILITY. Sei G ein Graph der Ordnung $n = n(G)$. Setze $g = \log n$. Wie weiter oben bereits angemerkt, gibt es einen kritisch $(k+1)$-färbbaren Graphen H der Ordnung $n(H) \leq (k+1)^{3g}$ mit Taillenweite mindestens g [LPS88, Bol91]. Für jede Kante $\{x_1, y_1\} \in E(G)$ nehme man eine Kopie von H her, wähle eine beliebige Kante $\{x_2, y_2\} \in E(H)$ aus und führe die HAJÓS Konstruktion mit diesen beiden Kanten durch. Der entstehende Graph G' hat offensichtlich Taillenweite $g(G') \geq g$ und ist k-färbbar genau dann, wenn es G ist (Übung). Wegen $n(H) \leq (k+1)^{3\log n} = n^{3\log(k+1)}$ ist

die Reduktion in polynomieller Zeit berechenbar, $n(G') \leq n^2 \cdot n(H) \leq n^{2+3\log(k+1)}$ und $g(G') \geq \log n \geq c \log n(G')$. □

Wie Übung 5.3.4 zeigt, ist der letzte Satz hinsichtlich der Taillenweite bis auf die Konstante c bestmöglich.

Überall dichte Graphen. Obige NP-Vollständigkeitsresultate machen Graphenklassen interessant, für die das Färbungsproblem polynomiell lösbar wird. In Kapitel 7 werden in diesem Zusammenhang die perfekten Graphen, in Kapitel 9 Graphen von beschränkter Baumweite behandelt.

Das Problem k-COLORABILITY (d.h. k ist fest) wird auf überall dichten (gemessen am Minimalgrad) Graphen zugänglich.[2] Der zugehörige Algorithmus benutzt (für $k = 3$) als Subroutine die untige Prozedur GREEDY_DS, die in einem Graphen G eine dominierende Menge D berechnet. (Hinsichtlich des DOMINATING SET Problems vergleiche Übung 1.4.22.) Für die Analyse des Färbungsalgorithmus werden wir die folgende, auch für sich allein genommen interessante Abschätzung für die Größe von D verwenden.

Lemma 5.4.8 *Die Prozedur* GREEDY_DS *konstruiert in einem Graphen G vom Minimalgrad $\delta := \delta(G)$ eine dominierende Menge D der Größe*

$$|D| \leq \left\lceil n \frac{1 + \ln(\delta + 1)}{\delta + 1} \right\rceil. \tag{5.6}$$

```
PROCEDURE GREEDY_DS;

BEGIN
  D := ∅;
  U := V;                {die Menge der noch nicht dominierten Knoten}
  WHILE U ≠ ∅ DO BEGIN
    v := argmax{|(u + Γ(u)) ∩ U| : u ∈ V \ D};
    D := D + v;
    U := U \ (v + Γ(v));
  END;
END;
```

Beweis. Offenbar produziert GREEDY_DS eine dominierende Knotenmenge D für G. O.B.d.A. sei $\delta < n - 1 \Leftrightarrow \frac{1+\delta}{n} < 1$. Wir zeigen zunächst:

> *In jedem Schleifendurchlauf sinkt die Zahl der von D noch nicht dominierten Knoten U um mindestens einen Faktor $1 - \frac{1+\delta}{n}$.*

Da kein Knoten aus U zu einem Knoten in D benachbart ist, gilt

$$\sum_{u \in V \setminus D} |(u + \Gamma(u)) \cap U| = \sum_{u \in U} (d(u) + 1).$$

Also existiert nach dem Schubfachprinzip ein Knoten $v \in V \setminus D$ mit

$$|(v + \Gamma(v)) \cap U| \geq \frac{1}{|V \setminus D|} \sum_{u \in V \setminus D} |(u + \Gamma(u)) \cap U| \geq |U| \cdot \frac{1 + \delta}{n}, \tag{5.7}$$

[2] Dies ist kein singuläres Phänomen, vgl. z.B. Korollar 3.2.9, Übung 4.1.14 und [AKK95]. Auch etliche approximative Zählprobleme werden auf dichten Graphen zugänglich, vgl. [AFW95, JS89b, DFJ94].

und (5.7) gilt folglich auch für den von GREEDY_DS in einem Schleifendurchlauf gewählten Knoten v. Für die nach dem Einfügen von v in D verbleibende Menge von D noch nicht dominierter Knoten $U' := U \setminus (v + \Gamma(v))$ gilt somit:

$$|U'| \;=\; |U| - |(v + \Gamma(v)) \cap U| \;\leq\; |U| \cdot (1 - \frac{1+\delta}{n}), \tag{5.8}$$

was zu zeigen war.

Bezeichne nun $n_0 := n$ und n_i, $i \in \mathbb{N}$, die Anzahl der nach dem i-ten Schritt noch nicht dominierten Knoten. Dann gilt nach (5.8) für $i = 1, \ldots$

$$\begin{array}{rrcl} & n_i & \leq & (1 - \frac{1+\delta}{n}) \quad n_{i-1} \\ \Rightarrow & n_i & \leq & (1 - \frac{1+\delta}{n})^i \quad n. \end{array}$$

Schätzen wir die Anzahl der Schleifendurchläufe ab, nach denen U nur noch höchstens $\frac{n}{\delta+1}$ Knoten enthält. Unter Verwendung der Abschätzung $\ln(1-x) \leq -x$ für $0 \leq x < 1$ gilt

$$\begin{array}{rrcl} & (1 - \frac{1+\delta}{n})^k \; n & \leq & \frac{n}{\delta+1} \\ \Leftrightarrow & k & \geq & \frac{-\ln(1+\delta)}{\ln(1-\frac{1+\delta}{n})} \\ \Leftarrow & k & \geq & n \cdot \frac{\ln(1+\delta)}{1+\delta}, \end{array}$$

d.h. spätestens, wenn D eine Größe von $\left\lceil n \cdot \frac{\ln(1+\delta)}{1+\delta} \right\rceil$ erreicht hat, dominiert D nur noch höchstens $\frac{n}{\delta+1}$ viele Knoten *nicht*. Also werden nur noch höchstens $\lfloor \frac{n}{\delta+1} \rfloor$ weitere Knoten in D aufgenommen. □

Aus Ergebnissen von ALON [Alo90] folgt, daß die Abschätzung (5.6) fast bestmöglich ist.

Satz 5.4.9 (EDWARDS 1986) *Seien $k \geq 3$ und $\epsilon > 0$ beliebig, aber fest. Dann besitzt das Problem k-*COLORABILITY*, eingeschränkt auf die Klasse aller Graphen G vom Minimalgrad $\delta(G) \geq (1 - \frac{1}{k-2} + \epsilon) \cdot n$, einen polynomiellen Algorithmus.*

Beweis. Wir zeigen dies hier nur für den Fall $k = 3$. Sei $G = (V, E)$ ein Graph. Sei o.B.d.A. $a := 1 - \frac{1}{k-2} + \epsilon < \frac{n-1}{n}$. Sei D die dominierende Menge in G, die GREEDY_DS konstruiert. Wegen $\delta(G) \geq a\, n$ gilt nach (5.6) $|D| \leq C \log n$ für eine Konstante $C \in \mathbb{R}_+$ (die nur von k und ϵ abhängt). Mithin gibt es höchstens $3^{C \log n} = n^{C \log 3}$, also polynomiell viele verschiedene 3-Färbungen von $G[D]$. Diese testen wir nun alle sukzessive durch und versuchen jeweils, die Färbung von $G[D]$ auf ganz G fortzusetzen. Offenbar ist G 3-färbbar genau dann, wenn mindestens eine der 3-Färbungen von $G[D]$ auf ganz G fortsetzbar ist. Für jeden der Knoten $r \in R := V \setminus D$ ist mindestens eine der drei Farben $\{1, 2, 3\}$ verboten (möglicherweise sogar alle, dann ist die vorliegende Färbung nicht fortsetzbar); bezeichne $S_r \subseteq \{1, 2, 3\}$, $|S_r| \leq 2$, die Menge der für einen Knoten $r \in R$ noch zulässigen Farben. Falls nun $|S_r| \geq 1$ für alle $r \in R$, so läßt sich die Fortsetzbarkeit der 3-Färbung von $G[D]$ auf R durch eine 2-SAT-Formel ausdrücken. Hierzu führen wir Boolesche Variablen $\{x_{rs} : r \in R,\ s \in S_r\}$ ein, die sich interpretieren lassen als „Knoten r trägt Farbe s“.

(i)	$\bigvee_{s \in S_r} x_{rs}$	für alle $r \in R$
(ii)	$\overline{x_{rs_1}} \vee \overline{x_{rs_2}}$	für alle $r \in R$ mit 2-elementigem $S_r = \{s_1, s_2\}$
(iii)	$\overline{x_{r_1 s}} \vee \overline{x_{r_2 s}}$	für alle $\{r_1, r_2\} \in E(G[R])$ und $s \in S_{r_1} \cap S_{r_2}$

Offenbar ist eine 3-Färbung von $G[D]$ auf R fortsetzbar genau dann, wenn die hierzu konstruierte 2-SAT-Formel erfüllbar ist. Da 2-SAT in polynomieller Zeit entschieden werden

kann (vergleiche Satz 1.4.8), erhalten wir einen insgesamt polynomiellen Algorithmus für 3-COLORABILITY auf überall dichten Graphen. □

Leider ist die Laufzeit des Algorithmus aus Satz 5.4.9 für allgemeines k exponentiell in k, so daß er für das Optimierungsproblem GRAPH COLORING nicht dienlich ist. Wie Übung 5.4.10 zeigt, ist Satz 5.4.9 in gewisser Hinsicht bestmöglich.

Übungen

Die HAJÓS-Konstruktion wurde im Zusammenhang mit der 4-Farben-Vermutung formuliert. Zum Beweis der 4-Farben-Vermutung „genügt" es, zu zeigen, daß alle 5-konstruierbaren Graphen nicht planar sind. Mehr zur HAJÓS-Konstruktion in [HRT86, PU95].

Übung 5.4.10 [Edw86] **a)** *Sei $k \geq 3$ beliebig, aber fest. Dann ist das Entscheidungsproblem k-*COLORABILITY*, eingeschränkt auf die Klasse aller Graphen G vom Minimalgrad $\delta(G) \geq (1 - \frac{1}{k-2}) \cdot n$,* NP*-vollständig.*
b) *Sei $0 < \epsilon \leq 1$ fest. Dann ist das Entscheidungsproblem 3-*COLORABILITY*, eingeschränkt auf die Klasse aller Graphen G vom Minimalgrad $\delta(G) \geq n^{1-\epsilon}$,* NP*-vollständig.*

Übung 5.4.11 *Sei $k \in \mathbb{N}$ fest.* **a)** *Vollständig k-partite Graphen färbt der Algorithmus* GREEDY_FÄRBUNG *unter jeder Anordnung der Knoten mit optimal vielen Farben* [MMI72].
b) *Vollständig k-partite Graphen können in Zeit $\mathcal{O}(n+m)$ erkannt werden.*
c) *Das (Such-) Problem, für einen Eingabegraphen G eine $\chi(G)$-Färbung zu konstruieren, ist NP-äquivalent.* [*Hinweis:* versuche, Kanten einzufügen. Benutze nicht Satz 1.4.29.]

Das Problem PARTITION INTO INDEPENDENT SETS OF SIZE $\leq k$, $k \in \mathbb{N}$ fest, ist eine Ressourcenbeschränkte Variante des Graphen-Färbungsproblems: hier ist eine Färbung eines Graphen mit minimal vielen Farben gesucht, ohne daß eine Farbklasse mehr als k Knoten enthält.

Übung 5.4.12 a) *Für $k \geq 3$ ist* PARTITION INTO INDEPENDENT SETS OF SIZE $\leq k$ NP*-vollständig. (Was ist mit $k = 2$?)*
b) PARTITION INTO INDEPENDENT SETS OF SIZE $\leq k$ *liegt in* P *für bipartite Graphen* [BJ95, KGS95].
c) *Das Problem* EQUITABLE COLORING*, die minimale Anzahl von Farben zu berechnen, mit der ein Graph gefärbt werden kann, so daß sich die Kardinalitäten der Farbklassen nur um höchstens 1 unterscheiden, ist* NP*-vollständig.*
[*Hinweis zu a) und c):* Benutze die NP-Vollständigkeit von PARTITION INTO TRIANGLES auf K_4-freien Graphen (siehe [GJ79]).]

PARTITION INTO INDEPENDENT SETS OF SIZE $\leq k$ ist selbst auf Intervallgraphen NP-vollständig [BJ95], wo das gewöhnliche Graphenfärbungsproblem polynomiell lösbar ist. Zur Approximation dieses Problems siehe Übung 12.4.7.

Ein Mengensystem $\mathcal{F} \subseteq \binom{V}{3}$ auf einer Knotenmenge V heißt *3-uniformer Hypergraph*, die Mengen $f \in F$ *Hyperkanten*. $\mathcal{H} = (V, \mathcal{F})$ heißt *2-färbbar*, falls es eine Partition $V = V_1 \dot\cup V_2$ gibt, so daß keine Hyperkante $f \in F$ ganz in V_1 oder ganz in V_2 enthalten ist. Man zeige:

Übung 5.4.13 (LOVÁSZ 1973) *Das Entscheidungsproblem „gegeben ein 3-uniformer Hypergraph $\mathcal{H}$, ist $\mathcal{H}$ 2-färbbar?" ist* NP*-vollständig.* [*Hinweis:* Reduktion von 3-COLORABILITY. Für einen Graphen $G = (V, E)$ betrachte den Hypergraphen $\mathcal{H}(G)$ mit Knotenmenge $(\{1,2,3\} \times V) \cup \{\infty\}$ und Kantenmenge $\{\{(i,u),(i,v),\infty\} : i \in \{1,2,3\}, \{u,v\} \in E\} \cup \{\{(1,v),(2,v),(3,v)\} : v \in V\}$]

5.5 Färbungsalgorithmen

In diesem Abschnitt werden die wichtigsten exakten Graph-Färbungsalgorithmen und einige weitere Heuristiken behandelt.

5.5.1 Exakte Färbungsalgorithmen

Der Greedy-Färbungsalgorithmus und seine Derivate finden i.a. keine optimale Knotenfärbung. Wie konstruiert man aber nun eine optimale, d.h. $\chi(G)$-Färbung eines Graphen? Aufgrund der NP-Vollständigkeit des Färbungsproblems ist nicht zu erwarten, daß es für dieses Problem einen polynomiellen Algorithmus gibt.

Man könnte den Greedy-Färbungsalgorithmus z.B. auf alle $n!$ verschiedenen Anordnungen der Knotenmenge anwenden und so wegen Lemma 5.2.4 eine $\chi(G)$-Färbung von G finden. Nach der STIRLING*schen Formel* (vergleiche Gleichung (A.2) im Anhang)

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \tag{5.9}$$

hätte dieser Algorithmus eine Laufzeit von

$$\Omega\left(\left(\frac{n}{e}\right)^n\right) = \Omega\left(2^{n(\log n - \log e)}\right).$$

Vergleichen wir das mit dem folgenden Ansatz.

Der Algorithmus von Zykov. Sei $G = (V, E)$ ein Graph und $e = \{u, v\} \notin E$. Es sei daran erinnert, daß $G + e$ den Graphen bezeichnet, den man aus G durch Hinzufügen der Kante e erhält, und G/e den Graphen, den man aus G erhält, wenn man die Knoten u und v identifiziert.

Lemma 5.5.1 (ZYKOV 1949) *Sei G ein Graph. Dann gilt für zwei nicht-adjazente Knoten x und y in G:* $\chi(G) = \min\{\chi(G + \{x, y\}), \chi(G/\{x, y\})\}$.

Beweis. „$\leq$“: Jede Färbung von $G + \{x, y\}$ ist auch eine Färbung von G. Also gilt $\chi(G) \leq \chi(G + \{x, y\})$. Jede Färbung des Graphen $G/\{x, y\}$ induziert eine Färbung von G mit genauso vielen Farben: man färbt x und y mit der Farbe, in der der Knoten gefärbt ist, der x und y repräsentiert, und übernimmt die restliche Färbung. Also gilt auch $\chi(G) \leq \chi(G/\{x, y\})$.
„$\geq$“: Sei $c : V \to \{1, \ldots, \chi(G)\}$ eine optimale Knotenfärbung von G. Sind die Knoten x und y in c verschieden gefärbt, dann ist c auch eine zulässige Färbung von $G + \{x, y\}$, d.h. es gilt $\chi(G + \{x, y\}) \leq \chi(G)$. Ist andernfalls $c(x) = c(y)$, so stellt c eine zulässige Färbung von $G/\{x, y\}$ dar, so daß $\chi(G/\{x, y\}) \leq \chi(G)$ gilt. □

Daraus ergibt sich unmittelbar der folgende rekursive Algorithmus, um die chromatische Zahl eines Graphen zu berechnen [Rea69]. Man stellt zunächst fest, ob der vorliegende Graph G vollständig ist. Falls ja, gibt die Prozedur seine Knotenzahl als chromatische Zahl zurück, andernfalls werden zwei nicht benachbarte Knoten x und y in G ausgewählt, rekursiv die Werte $\chi(G + \{x, y\})$ und $\chi(G/\{x, y\})$ berechnet und der kleinere der beiden Werte (rsp. die Färbung mit weniger Farben) als Funktionswert zurückgegeben. Dieser Algorithmus läuft (gemäß einer Tiefensuche) die Knoten des Binärbaums ab, der G als Wurzel hat und in dem jeder Knoten H, der kein vollständiger Graph ist, die beiden Söhne

$H + \{x, y\}$ und $H/\{x, y\}$ hat für zwei in H nicht benachbarte Knoten x und y. Die Blätter dieses Binärbaums korrespondieren zu vollständigen Graphen, und die chromatische Zahl von G ist das Minimum der Knotenzahlen der Graphen in diesen Blättern.

Um die Laufzeit dieses Algorithmus abzuschätzen, bezeichne $f(n)$ die maximale Anzahl von Prozeduraufrufen für einen Graphen auf n Knoten. Es ist also $f(1) = 1$. Betrachte nun einen Prozeduraufruf für einen Graphen G auf n Knoten. Für die rekursiven Prozeduraufrufe werden entweder zwei Knoten von G identifiziert (womit die Knotenzahl sinkt) oder eine Kante eingefügt. Es können G aber höchstens $\binom{n}{2} \leq n^2/2$ Kanten hinzugefügt werden. Mithin veranlaßt der Prozeduraufruf für G höchstens $n^2/2$ viele Prozeduraufrufe für Graphen auf $n - 1$ Knoten, und wir erhalten die Rekursionsformel

$$f(n) \leq n^2 \cdot f(n-1).$$

Da $f(n) = (n!)^2$ die Rekursionsformel und die Anfangsbedingung erfüllt, gilt $f(n) = \mathcal{O}((n!)^2)$. Da der Aufwand in jedem Knoten des Binärbaums (um H auf Vollständigkeit zu testen und die beiden Graphen $H + \{x, y\}$ und $H/\{x, y\}$ zu berechnen) durch $\mathcal{O}(n^2)$ beschränkt ist, ergibt sich die Abschätzung $\mathcal{O}(n^2 \cdot 2^{2n(\log n - \log e)})$ für die gesamte Laufzeit des Algorithmus von ZYKOV.

Man beachte, daß sowohl der Greedy-Permutationen-Färbungsalgorithmus wie auch der ZYKOV-Algorithmus viel Raum für Verbesserungen lassen. MCDIARMID [McD79] bewies jedoch, daß der vom ZYKOV-Algorithmus untersuchte Binärbaum mit Wurzel G für fast alle Graphen mindestens $c^n\sqrt{\log n}$ viele Knoten enthält für eine Konstante $c > 1$.

Der Independent-Set-Ansatz von Christofides. Ein auf CHRISTOFIDES [Chr71] zurückgehender, exakter Färbungsalgorithmus mit einer Laufzeit von „nur“ $\mathcal{O}(mn\ c^n)$ für eine Konstante $c \approx 2,445$, wendet rekursiv die Beobachtung an, daß es für jeden Graphen eine optimale Färbung gibt, in der eine (o.B.d.A. die erste) Farbklasse S eine *maximale* stabile Knotenmenge darstellt. Folglich gilt

$$\chi(G) \ = \ 1 + \min_S \{\chi(G - S)\},$$

wobei über alle (inklusions-) maximalen stabilen Knotenmengen S in G minimiert wird. Damit kann die chromatische Zahl eines Graphen in Dynamischer-Programmierungs-Manier bestimmt werden:

```
FOR k := 1 TO n DO
  FOR ALL U ∈ (V k) DO
    χ(G[U]) := 1 + min_S χ(G[U \ S]);
```

wobei das Minimum über alle maximalen stabilen Knotenmengen in $G[U]$ genommen wird. Für die Laufzeitabschätzung (vgl. [Law76b]) beachte man:

(1) es gibt einen Algorithmus, der alle maximalen stabilen Mengen eines Graphen in Zeit $\mathcal{O}(mnK)$ auflistet, wobei K die Anzahl seiner maximalen stabilen Mengen bezeichnet ([TIAS77], siehe auch [Law79, JYP88]);

(2) ein Graph besitzt höchstens $3^{n/3}$ viele maximale stabile Knotenmengen [MM65], vergleiche Satz 14.3.2.

Die Zeit, die obiger Algorithmus aufwendet, um $\chi(G[U])$ zu berechnen, läßt sich somit, da die Zahlen $\chi(G[U \setminus S])$ ja bereits alle vorliegen, abschätzen durch $\mathcal{O}(mk3^{k/3})$. Insgesamt ist die Laufzeit daher beschränkt durch

$$\sum_{k=1}^{n} \binom{n}{k} mk3^{k/3} \;\leq\; mn \sum_{k=0}^{n} \binom{n}{k} 3^{k/3} \;=\; mn\,(1+\sqrt[3]{3})^n.$$

In der Praxis ist dieser Algorithmus jedoch aufgrund seines exponentiellen Speicherbedarfs für die Zahlen $\chi(G[U])$, $U \subseteq V$, unbrauchbar.

Backtracking nach Brown. Tatsächlich erweist sich ein auf BROWN [Bro72] zurückgehendes Backtracking-Verfahren in der Praxis für kleine Graphen (bis etwa 50 Knoten) als der effizienteste Algorithmus. Bei der Anwendung des Greedy-Färbungsalgorithmus auf alle $n!$ Permutationen der Knotenmenge wird eine bestimmte Partition der Knotenmenge in stabile Mengen u.U. sehr oft erzeugt – jeweils nur mit verschiedenen Farben auf den Partitionsklassen. Das Verfahren von BROWN erzeugt dagegen jede Partition von V in stabile Mengen nur höchstens einmal. Die Anzahl $\mathcal{B}_n$ (die sogenannten BELL-Zahlen) der Partitionen einer n-elementigen Menge ist jedoch vergleichsweise nur

$$\mathcal{B}_n \;=\; 2^{n\log n - n\log\ln n + \mathcal{O}(n)},$$

vergleiche [Bru58]. Sei eine Anordnung $v_1, \ldots, v_n$ der Knoten des Graphen vorgegeben. Der Knoten v_1 kann offenbar o.B.d.A. mit der Farbe 1 gefärbt werden. Seien die Knoten $v_1, \ldots, v_{k-1}$ bereits mit den Farben $\{1, \ldots, AnzFarben[k-1]\}$ gefärbt. Dann sind nun alle Partitionen von V zu bilden, deren Partitionsklassen stabil sind und mit der Färbung der ersten $k-1$ Knoten kompatibel sind. Welche Möglichkeiten gibt es hierbei für den Knoten v_k? Einerseits kann man v_k zu jeder der bereits bestehenden Farbklasse hinzufügen, zu der er nicht adjazent ist; andererseits kann man eine neue Farbklasse erzeugen, die o.B.d.A. $AnzFarben[k-1]+1$ heißen darf und die genau aus dem Knoten v_k besteht. Die restlichen Knoten $v_{k+1}, \ldots, v_n$ werden dann jeweils durch einen rekursiven Aufruf der Färbungsprozedur gefärbt.

Wenn man von den durch Adjazenzen verbotenen Farben absieht, hat man für den k-ten Knoten k Farben zur Auswahl, so daß man als worst-case Zeitkomplexität wieder $n! = \mathcal{O}(2^{n\log n})$ erhält. Durch einige einfache Modifikationen läßt sich die Laufzeit allerdings in der Regel noch stark reduzieren. Wenn z.B. eine Färbung mit *chi* Farben gefunden wurde, so braucht man in der Folge keinen Berechnungszweig mehr zu betrachten, bei dem ein Knoten mit der Farbe *chi* gefärbt wurde – denn nunmehr ist lediglich noch festzustellen, ob der Graph nicht vielleicht auch eine $(chi-1)$–Färbung besitzt. Die einfachste Version dieses Backtracking-Algorithmus hat somit die folgende Gestalt ($A[1..n; 1..n]$ ist die Adjazenzmatrix, der Vektor $c[1..n]$ enthält in den ersten k Positionen die aktuelle Färbung, $AnzFarben[k] := |\{c[1], \ldots, c[k]\}|$ und das Boolesche Feld $ZulFarbe[k; 1..n]$ gibt in den ersten $AnzFarben[k]$ Positionen die für den Knoten $k \geq 2$ zulässigen der bisher verwendeten Farben $\{1, \ldots, AnzFarben[k]\}$ an):

```
PROCEDURE Färbe (k);
  BEGIN
    {1. Möglichkeit: v_k nacheinander jeder der bereits bestehenden
                                   Farbklassen zuordnen, wenn zulässig:}
    AnzFarben[k] := AnzFarben[k − 1];
    {die für v_k zulässigen Farben berechnen:}
    FOR i := 1 TO AnzFarben[k] DO
      ZulFarbe[k, i] := TRUE;
    FOR i : v_i ∈ Γ(v_k) DO
      IF i < k THEN   ZulFarbe[k, c[i]] := FALSE;
    i := 1;
    WHILE (i ≤ AnzFarben[k]) AND (AnzFarben[k] < chi) DO BEGIN
      IF ZulFarbe[k][i] THEN BEGIN
        c[k] := i;
        IF k < n THEN
          Färbe (k + 1)
        ELSE {bessere Färbung gefunden!}
          chi := AnzFarben[k];
      END;
      i := i + 1;
    END; {while}
    {2. Möglichkeit: mit v_k eine neue Farbklasse bilden:}
    IF (AnzFarben[k] + 1 < chi) THEN BEGIN
      AnzFarben[k] := AnzFarben[k] + 1;
      c[k] := AnzFarben[k];
      IF k < n THEN
        Färbe (k + 1)
      ELSE {bessere Färbung gefunden!}
        chi := AnzFarben[k];
    END;
  END; {Färbe}

BEGIN {Hauptprogramm}
  chi := n;                {Initialisierung der oberen Schranke an χ(G)}
  ordne die Knoten in einer günstigen Reihenfolge v_1, ..., v_n an;
  c[1] := 1;                        {o.B.d.A. erhält v_1 stets die Farbe 1}
  AnzFarben[1] := 1;
  Färbe (2);                                          {o.B.d.A. sei n ≥ 2}
  gib chi aus;
END;
```

Die allererste Rekursion bis zur Tiefe $k = n$ tut also nichts anderes als der Greedy-Färbungsalgorithmus auf der Knotenordnung $v_1, \ldots, v_n$. Die Schranke *chi* ist nun bereits höchstens $\Delta(G) + 1$. Im Folgenden wird dann sukzessive *chi* zu vermindern versucht. Tatsächlich verwendet der Algorithmus allerdings fast die gesamte Rechenzeit darauf, zu überprüfen, daß G keine $(\chi(G) - 1)$–Färbung besitzt. Es ist daher unnötig, zu Beginn der Berechnung beispielsweise durch die Smallest-Last-Färbungsheuristik eine bessere obere

Schranke *chi* an $\chi(G)$ zu bestimmen. Ein paar weitere Tricks verbessern i.a. die Laufzeit dieses Algorithmus. Beispielsweise kann man in dem Augenblick, in dem man die k-te Farbe vergibt, alle späteren Knoten vom Grad $< k$ aus G eliminieren, denn sie können in jeder solchen Färbung stets zulässig gefärbt werden. Weiter kann man die Anordnung $v_1, \ldots, v_n$ durch eine Maximum-Adjazenz-Suche bestimmen. Dadurch sind, so hofft man, für den Knoten v_k in dem Augenblick, in dem er gefärbt werden soll, viele Farben bereits verboten. Die noch ungefärbten Knoten kann man schließlich auch dynamisch, d.h. nach jedem Färben eines Knotens, umordnen; hier empfiehlt sich die Saturation-smallest-last-Heuristik [KJ85] oder die Heuristik, als nächsten stets einen ungefärbten Knoten zu färben, für den am wenigsten zulässige Farben verblieben sind [Kor79].

Übung und Anmerkung

Übung 5.5.2 *Man implementiere den Backtracking-Algorithmus mit smallest-last- und maximum-Adjazenz-Anordnung (jeweils vorwärts und rückwärts) und vergleiche die Laufzeit auf dem C_{29}^5 (i.e. einem Kreis auf 29 Knoten, bei dem zwischen allen Knoten, die Abstand höchstens 5 von einander haben, noch eine Kante eingefügt wird), auf dem* Mycielski-*Graph M_6 und auf zufälligen Graphen verschiedener Dichte der Ordnung 30-50.*

Übung 5.5.3 *Man entwerfe einen $\mathcal{O}(m2^n)$-Algorithmus für* 3-Colorability.

Der Rekord für dieses Problem liegt bei $\mathcal{O}(1.3446^n)$ [BE95].

Es gibt einen einfachen Backtracking-Algorithmus für das Entscheidungsproblem k-Colorability, $k \geq 3$ fest, der erwartet *konstante* Laufzeit hat (d.h. gemittelt über alle Graphen) [Wil84, BW85]. Der Grund für dieses auf den ersten Blick überraschende Ergebnis ist jedoch der, daß fast alle Graphen einen K_{k+1} enthalten. Es gibt jedoch auch einen Algorithmus, der jeden k-färbbaren Graphen optimal färbt und dessen erwartete Laufzeit linear ist [DF89, PS92b].

5.5.2 Färbungsheuristiken

Wie wir in Kapitel 10 sehen werden, ist i.a. selbst das Problem NP-schwer, für ein festes $k \in \mathbb{N}$ jeden Graphen mit höchstens $k \cdot \chi(G)$ vielen Farben zu färben. Daher gewinnen Heuristiken an Bedeutung, die Graphen in polynomieller Zeit färben und immerhin „in der Regel“ gute Färbungen liefern. Solchen Heuristiken kann man beispielsweise danach beurteilen, welche Graphenfamilien sie optimal färben. Die Smallest-last-Heuristik (s.o.) färbte z.B. alle Wälder optimal. Da das Färbungsproblem für bipartite Graphen in P liegt, ist es naheliegend, von einer Heuristik zu fordern, daß sie zumindest die bipartiten Graphen optimal färbt.

Die *Saturation-Largest-First* Heuristik nach Brélaz [Bré79] führt eine Greedy-Färbung des Eingabegraphen durch, bei der die Anordnung σ der Knoten *dynamisch* berechnet wird. Zunächst wird ein Knoten $v_1 \in V$ mit Farbe $c(v_1) := 1$ gefärbt. Wenn nun die Knoten $v_1, \ldots, v_{i-1}$, $1 < i \leq n$, bereits gefärbt sind, so färbt diese Heuristik als nächstes einen Knoten, für den schon am meisten Farben verboten sind: definiere für jeden noch nicht gefärbten Knoten $v \in V$ den sogenannten Sättigungsgrad

$$dsatur(v) := |\{c(u) : u \text{ ist ein bereits gefärbter Nachbar von } v\}|;$$

dann wird also als nächster Knoten ein Knoten $v_i := argmax\ \{dsatur(v) : v \text{ ungefärbt}\}$ gewählt und mit der kleinsten zulässigen Farbe gefärbt.

Proposition 5.5.4 *Die Saturation-Largest-First Heuristik färbt bipartite Graphen optimal.*

Beweis. Da die Saturation-Largest-First Heuristik die Zusammenhangskomponenten eines Graphen nacheinander färbt, sei G o.B.d.A. ein zusammenhängender bipartiter Graph. Da dann jeder von dieser Heuristik gefärbte Knoten $v_i \neq v_1$ zu einem bereits früher gefärbten Knoten adjazent ist, gibt es für ihn einen $v_1 - v_i$–Pfad P_i. Angenommen, die Heuristik benutzt nicht nur die Farben 1 und 2 für G und sei v_j der erste mit Farbe 3 gefärbte Knoten. Dann hat v_j einen Nachbarn v_{i_1} mit $c(v_{i_1}) = 1$ und einen Nachbarn v_{i_2} mit $c(v_{i_2}) = 2$. Sei v_s der letzte gemeinsame Knoten der Pfade P_{i_1} und P_{i_2}. Dann bildet die Vereinigung der Teilpfade von P_{i_1} und P_{i_2} zwischen v_s und v_j einen ungeraden Kreis, da dessen Knoten bis auf v_j abwechselnd mit Farbe 1 und 2 gefärbt sind – Widerspruch. □

Die Saturation-Largest-First Heuristik kann mit linearer Laufzeit implementiert werden. Eine Variante dieses Algorithmus färbt *fast alle* Graphen G mit chromatischer Zahl $\chi(G) \leq (1-\epsilon)\log n$, $0 < \epsilon < 1$ fest, optimal [Tur88].

Die folgende Heuristik *Similarity-Merge* stammt von DUTTON und BRIGHAM [DB81]. Setze $H := G$. Solange es nicht-adjazente Knoten in H gibt, identifiziere zwei nicht-adjazente Knoten, die am meisten gemeinsame Nachbarn haben. Zwei Knoten v_1 und v_2 zu identifizieren heißt dabei, die beiden Knoten durch einen Superknoten v_{12} zu ersetzen, der zu einem Knoten u im Restgraphen genau dann adjazent ist, wenn mindestens einer der Knoten v_1 oder v_2 vorher mit u adjazent war. Ein Superknoten vertritt somit stets eine stabile Knotenmenge in G. Auf diese Art endet man schließlich bei einem vollständigen Graphen H. Diesen färbt man mit $|H|$ Farben und überträgt die Farbe eines jeden Superknotens auf die stabile Knotenmenge, die er vertritt. Offensichtlich ist die erhaltene Färbung von G zulässig.

Proposition 5.5.5 *Similarity-Merge färbt bipartite Graphen optimal.*

Beweis. Sei $G = (U, V, E)$ ein bipartiter Graph. Da Similarity-Merge stets Knoten aus einer Zusammenhangskomponente identifiziert, sei G o.B.d.A. zusammenhängend. Solange H mindestens drei Knoten hat, gibt es daher stets nicht-adjazente Knoten mit mindestens einem gemeinsamen Nachbarn. Da zwei nichtadjazente Knoten aus verschiedenen Teilen von H keinen gemeinsamen Nachbarn haben, werden ausschließlich Knoten, die in derselben Bipartitionsmenge liegen, identifiziert, bis schließlich $H = K_2$. □

Eine Variante dieses Algorithmus färbt *fast alle* Graphen G mit chromatischer Zahl $\chi(G) \leq \sqrt{\frac{n}{196 \log n}}$ optimal [Kuc89].

Ein gänzlich anderer Ansatz besteht darin, sukzessive stabile Mengen aus G herauszuholen und jede der stabilen Mengen mit einer neuen Farbe zu färben. Die erste derartige Heuristik, die wir vorstellen, versucht im k-ten Färbungsschritt möglichst viele der bisher noch ungefärbten Knoten mit Farbe k zu färben. Da das Teilproblem, eine maximum stabile Menge in einem Graphen zu konstruieren, jedoch NP-schwer ist, begnügt sich diese Heuristik jeweils mit einer *maximalen* stabilen Knotenmenge, wie sie beispielsweise die Heuristik GREEDY_MIN_IS liefert (vgl. Abschnitt 1.5).

```
FUNCTION MIS_Color (G) : INTEGER;

BEGIN
  k := 0;                                              {aktuelle Farbe}
  U := V;                                  {Menge der ungefärbten Knoten}
  WHILE U ≠ ∅ DO BEGIN
    k := k + 1;
    S := Greedy_Min_IS (G[U]);
    färbe alle Knoten s ∈ S mit Farbe k;
    U := U − S;
  END;
  MIS_Color:= k;
END; {MIS_Color}
```

Die Güte dieser Heuristik wird in Abschnitt 11.3 näher besprochen.

Eine der besten Färbungsheuristiken (vgl. [SDK83, JAMS91]) ist die sogenannte *Recursive-Largest-First* Heuristik nach Leighton [Lei79]. Sie entfernt in jedem Schritt eine maximale stabile Menge $S \subseteq U$ aus $G[U]$, so daß der Schnitt $\langle S, U \setminus S \rangle$ möglichst groß wird. Dann ist – so die Intuition – der Restgraph bald leer und kann mit einer letzten Farbe gefärbt werden. $d(u, A) := |\Gamma(u) \cap A|$ bezeichne die Anzahl der Nachbarn von u in einer Knotenmenge $A \subseteq V$.

```
PROCEDURE Recursive_Largest_First (G) : INTEGER;

BEGIN
  k := 0;  U := V;          {aktuelle Farbe bzw. ungefärbte Knoten}
  WHILE U ≠ ∅ DO BEGIN
    k := k + 1;
    {eine stabile Knotenmenge in G[U] mit Farbe k färben:}
    Γ := ∅; {ungefärbte Nachbarn der k-ten Farbklasse}
    REPEAT
      U_max := {u ∈ U : d(u, Γ) = max_{x∈U}{d(x, Γ)}};
      wähle einen Knoten u ∈ U_max mit d(u, U) = min_{x∈U_max}{d(x, U)};
      c[u] := k;
      Γ := Γ + (Γ(u) ∩ U);
      U := U − u − Γ(u);
    UNTIL U = ∅;
    U := Γ;
  END; {while}
  Recursive_Largest_First:= k;
END; {Recursive_Largest_First}
```

Übung 5.5.6 *Die Recursive-Largest-First Heuristik kann in Zeit $\mathcal{O}(nm)$ implementiert werden und färbt alle bipartiten Graphen optimal.*

Übung 5.5.7 *Man implementiere und vergleiche die Heuristiken Smallest-Last, Saturation-Largest-First, Similarity-Merge und Recursive-Largest-First im Hinblick auf die Anzahl der verwendeten Farben sowie die Laufzeit auf einem $G_{1000,0.5}$ unter jeweils hundert verschiedenen Permutationen der Knotenmenge.*

Weitere Färbungsheuristiken findet man in [PW89, Wer90, JAMS91].

5.6 Kantenfärbung: der Satz von Vizing

In diesem Abschnitt werden Färbungen der *Kanten* eines Graphen G untersucht. Die kleinste Zahl $k \in \mathbb{N}$, so daß es eine Färbung der Kanten von G mit k Farben gibt, so daß keine zwei inzidenten Kanten dieselbe Farbe tragen, heißt der *chromatische Index* $\chi'(G)$. Offenbar ist mit dieser Definition der chromatische Index eines Graphen G gleich der chromatischen Zahl seines Linegraphen: $\chi'(G) := \chi(L(G))$; Wir haben es also mit einem Knotenfärbungsproblem für eine spezielle Graphenklasse zu tun, den Linegraphen.

Beispiele und Abschätzungen. Offenbar haben die Kreise C_n chromatischen Index $\chi'(C_n)$ gleich zwei oder drei, jenachdem ob n gerade oder ungerade ist. Da die mit einem Knoten von G inzidierenden Kanten eine Clique im Linegraphen $L(G)$ bilden, gilt:

$$\chi'(G) \;\geq\; \omega(L(G)) \;\geq\; \Delta(G).$$

Jede Farbklasse einer Kantenfärbung ist ein Matching in G. Daher gilt analog zur Abschätzung (5.2):

$$\chi'(G) \;\geq\; \left\lceil \frac{m}{\nu(G)} \right\rceil. \tag{5.10}$$

Damit haben auch die vollständigen Graphen K_n (wie schon die Kreise) je nach Parität von n chromatischen Index $\chi'(K_n)$ gleich $\Delta(K_n)$ oder $\Delta(K_n)+1$:

Proposition 5.6.1 (VIZING 1965b) $$\chi'(K_n) = \begin{cases} n-1, & \textit{falls } n \textit{ gerade,} \\ n, & \textit{falls } n \textit{ ungerade.} \end{cases}$$

Beweis. Für n gerade ist dies äquivalent zur 1-Faktorisierbarkeit des K_n, vgl. Übung 4.1.1c. Für n ungerade kann der K_{n+1} mit n Farben kantengefärbt werden, was auf dem K_n als Subgraphen eine Kantenfärbung mit $n = \Delta(K_n)+1$ Farben induziert. Dies ist nach (5.10) auch schon bestmöglich:

$$\chi'(K_n) \;\geq\; \left\lceil \frac{m}{\nu(K_n)} \right\rceil \;=\; \left\lceil \frac{n(n-1)/2}{(n-1)/2} \right\rceil \;=\; n \qquad \square$$

Anwendungen. Auch dieses Färbungsproblem, insbesondere für bipartite Graphen, tritt bei Scheduling-Problemen auf. Man stelle sich beispielsweise vor, eine Menge U von Studenten muß in einem in Zeiteinheiten geteilten Prüfungszeitraum von Professoren V geprüft werden, wobei jeder Student $u \in U$ eine bestimmte Anzahl Fächer bei einer entsprechenden Menge $\Gamma(u) \subseteq V$ von Professoren belegt hat; oder eine Menge U von Aufträgen muß von einer Menge V von Maschinen in möglichst wenigen Zeiteinheiten erledigt werden, wobei jede Aufgabe $u \in U$ eine Menge $\Gamma(u) \subseteq V$ von Maschinen (je eine Zeiteinheit) belegt.

Der Greedy-Färbungsalgorithmus, angewandt auf den Linegraphen $L(G)$, liefert die Schranke

$$\chi'(G) \;\leq\; 2\Delta(G) - 1.$$

Übung 5.6.2 a) *Man konstruiere Graphen, auf denen der Greedy-Färbungsalgorithmus i.a.* $2\Delta(G)-1$ *viele Farben benötigt.* **b)** *Für* $\Delta(G) \geq 3$ *gilt:* $\chi'(G) \;\leq\; 2\Delta(G) - 2$. [*Hinweis:* Satz 5.2.9 von BROOKS]

Diese Schranke erscheint zwar plausibel, doch wie der nächste Satz zeigt, wird man dem Kantenfärbungsproblem nicht gerecht, wenn man es lediglich als Färbungsproblem für den Linegraphen betrachtet. Denn tatsächlich kann der chromatische Index eines Graphen (im drastischen Gegensatz zur chromatischen Zahl) nur zwei Werte annehmen, wie VIZING in einer Arbeit, die das Studium von Kantenfärbungen stark beeinflußt hat, bewies:

Satz 5.6.3 (VIZING 1964) $\Delta(G) \leq \chi'(G) \leq \Delta(G)+1.$

Bevor wir diesen Satz beweisen, ist es hilfreich, das Problem zunächst in bipartiten Graphen zu betrachten. Der Satz 4.2.14 von KÖNIG besagt, daß sich jeder bipartite Graph G mit $\Delta(G)$ Farben kantenfärben läßt, was wie oben gesehen optimal ist. Wir beweisen dies nun mit Hilfe eines einfachen Färbungsalgorithmus noch einmal.

Satz 5.6.4 (KÖNIG 1916) *Für jeden bipartiten Graphen G gilt $\chi'(G) = \Delta(G)$. Eine solche Kantenfärbung von G kann in Zeit $\mathcal{O}(nm)$ konstruiert werden.*

Beweis. O.B.d.A. sei G zusammenhängend und damit $m = \Omega(n)$. Sei eine teilweise, aber zulässige Färbung der Kanten von G gegeben (zu Beginn sind alle Kanten noch ungefärbt). Sei nun $e = \{x, y\} \in E$ eine noch ungefärbte Kante von $G = (U, V, E)$, $x \in U$, $y \in V$. Der Satz ist gezeigt, wenn die Kante e in Zeit $\mathcal{O}(n)$ gefärbt werden kann. Da am Knoten x höchstens $\Delta(G) - 1$ viele gefärbte Kanten inzidieren, „fehlt" an x mindestens eine Farbe; analog fehlt auch am Knoten y mindestens eine Farbe.

Falls es eine Farbe $f_e \in \{1, \ldots, \Delta(G)\}$ gibt, die sowohl an x als auch an y fehlt, so kann e in Farbe f_e zulässig gefärbt werden.

Andernfalls fehlt am Knoten x eine Farbe a und am Knoten y eine Farbe b, so daß am Knoten x eine Kante in Farbe b und am Knoten y eine Kante in Farbe a inzidiert. Betrachte den Subgraphen $H(a, b)$ von G, der von den Kanten in Farbe a oder b gebildet wird. Da jeder Knoten in G mit höchstens einer Kante in jeder der beiden Farben inzidiert, hat der Graph $H(a, b)$ Maximalgrad zwei und ist daher die disjunkte Vereinigung von Pfaden und geraden Kreisen, die alternierend in den Farben a und b gefärbt sind. Sowohl x als auch y sind Endpunkt eines solchen Pfades. Sie sind aber nicht die Endpunkte ein und desselben Pfades, denn der Pfad, der im Knoten x beginnt, erreicht die Partitionsmenge V von G stets mit einer Kante in Farbe b, am Knoten y liegt jedoch nur eine Kante in Farbe a an (anders ausgedrückt: der Pfad würde zusammen mit der Kante e einen ungeraden Kreis in G bilden). Wenn man nun die Farben a und b in dem Pfad $P_{ab}(x)$ von $H(a, b)$, der im Knoten x beginnt, vertauscht, so bleibt die Kantenfärbung weiterhin zulässig: an den inneren Knoten des Pfades $P_{ab}(x)$ wurden lediglich die Farben a und b vertauscht, an jedem der Endknoten von $P_{ab}(x)$ ist eine Kante in eine Farbe umgefärbt worden, die dort vorher fehlte; schließlich inzidiert keine andere Kante von $H(a, b)$ mit den Kanten des Pfades. Nach Vertauschen der Farben a und b in dem Pfad $P_{ab}(x)$ fehlt nun die Farbe b auch am Knoten x, so daß die Kante e mit der Farbe b gefärbt werden kann.

Es bleibt, zu zeigen, daß pro Kante e höchstens $\mathcal{O}(n)$ Operationen notwendig sind. Die Farben f_e, a und b lassen sich in Zeit $\mathcal{O}(d(x) + d(y) + 1) = \mathcal{O}(n)$ berechnen. Um auch den Pfad $P_{ab}(x)$ in Zeit $\mathcal{O}(n)$ berechnen zu können, unterhält man ein Feld $Nachbar[v, f]$, $v \in V$, $1 \leq f \leq \Delta(G)$. Der Eintrag $Nachbar[v, f]$ sei hierbei gleich 0, falls mit dem Knoten v keine Kante in Farbe f inzidiert, und gebe andernfalls denjenigen Nachbarn $u \in \Gamma(v)$ an, so daß die Kante $\{u, v\}$ die Farbe f trägt. Zu Beginn wird $Nachbar[\cdot, \cdot]$ auf 0 initialisiert. Die Aktualisierung von $Nachbar[\cdot, \cdot]$ kostet nur $\mathcal{O}(n)$ Zeit pro Kante. □

Übung 5.6.5 *Warum funktioniert obiger Beweis für den Satz von* VIZING *nicht?*

Beweis des Satzes 5.6.3 von VIZING *(nach* [BF91]*):*
Es ist nur die obere Schranke zu zeigen. Wir induzieren nach der Kantenzahl $m(G)$. Für leere Graphen ist die Aussage trivial. Sei also G ein Graph, $\Delta := \Delta(G)$ und $e = \{x, y\} \in E$ eine beliebige Kante von G. Nach Induktionsvoraussetzung gibt es eine zulässige Kantenfärbung $c : E - e \to \{1, \ldots, \Delta + 1\}$ von $G - e$. Wir geben nun eine Prozedur an, um auch die Kante e mit einer dieser Farben zu färben.

Wir sagen, eine Farbe *fehlt* an einem Knoten $v \in V$, falls in der Kantenfärbung c keine der mit v inzidierenden Kanten mit dieser Farbe gefärbt ist. Offensichtlich fehlt an jedem Knoten v mindestens $\Delta + 1 - d(v) \geq 1$ Farbe. Für einen Knoten $v \in V$ bezeichne F_v die Menge der am Knoten v fehlenden Farben.

Sei $b \in F_y$ eine Farbe, die am Knoten y fehlt. Falls $b \in F_x$, so können wir e mit der Farbe b färben. Andernfalls gibt es genau eine Kante $\{x, y'\}$ in Farbe b, und das Problem ist vom Knoten y auf den Knoten y' verlagert: wenn wir $\{x, y'\}$ in einer anderen Farbe als b zulässig färben können, so darf $\{x, y\}$ in Farbe b gefärbt werden. Wir werden daher eine Folge $\{y_j : 1 \leq j \leq k\} \subseteq \Gamma(x)$ von Nachbarn von x und eine Folge $\{b_j : 1 \leq j \leq k\} \subseteq \{1, \ldots, \Delta + 1\}$ von Farben konstruieren, so daß

- $y_1 = y$;
- $b_j \in F_{y_j}$ für $1 \leq j \leq k$;
- für $1 \leq j < k$ die Kante $\{x, y_{j+1}\}$ mit Farbe b_j gefärbt ist und
- die Knoten $y_1, \ldots, y_k$ alle verschieden sind.

Wir werden in einem Moment sehen, wie $y_1, b_1, y_2, b_2, \ldots, y_k, b_k$ sukzessive konstruiert werden. Betrachten wir jedoch zunächst eine Situation, in der wir die Konstruktion der Folge abbrechen können. Wenn wir in der k-ten Runde feststellen sollen, daß die Kante $\{x, y_k\}$ in eine Farbe $f \in F_x \cap F_{y_k}$, $f \notin \{b_1, \ldots, b_{k-1}\}$ umgefärbt werden darf, so können wir wie folgt (zulässig!) umfärben:

```
PROCEDURE FARBSHIFT (k, f);

BEGIN
  c({x, y_k}) := f;
  FOR j := k - 1 DOWNTO 1 DO
    c({x, y_j}) := b_j;
END;
```

Damit ist die Kante $e = \{x, y_1\}$ erfolgreich gefärbt worden, und das Verfahren bricht ab. Die folgende Prozedur konstruiert die Folgen $\{y_j : 1 \leq j \leq k\}$ und $\{b_j : 1 \leq j \leq k\}$.

```
k := 0;   y_1 := y;
REPEAT
  {es gilt: die Knoten y_1,...,y_{k+1} sind alle verschieden}
  k := k + 1;
  wähle b_k ∈ F_{y_k};
  IF b_k ∈ F_x THEN BEGIN
    {es gilt b_k ∉ {b_1,...,b_{k-1}}}
    FARBSHIFT (k, b_k);
    EXIT;
  END
  ELSE BEGIN
    {b_k fehlt nicht an x}
    y_{k+1} := derjenige Nachbar von x mit c({x, y_{k+1}}) = b_k;
  END;
UNTIL y_{k+1} ∈ {y_1,...,y_k};
```

Beachte, daß die REPEAT-Schleife höchstens $d(x)$-mal durchlaufen wird, weil die Knoten $\{y_1, \ldots, y_k\}$ paarweise verschiedene Nachbarn von x sind. Angenommen nun, die REPEAT-Schleife wird nicht durch EXIT verlassen (in welchem Fall wir fertig wären), sondern mit $y_{k+1} = y_i \Leftrightarrow b_k = b_{i-1}$ für ein $i \in \{2, \ldots, k-1\}$ (offensichtlich gilt $i \neq 1$ und $i \neq k$). Sei $a \in F_x$ eine Farbe, die am Knoten x fehlt, und bezeichne $H(a, b_k)$ den Subgraphen von G, der von der Menge aller Kanten gebildet wird, die mit Farbe a oder b_k gefärbt sind. Da in $H(a, b_k)$ jeder Knoten vom Grad höchstens zwei ist, ist $H(a, b_k)$ eine disjunkte Vereinigung von (farblich alternierenden) Pfaden und geraden Kreisen. Offenbar erhält man eine neue, wiederum zulässige Kantenfärbung von G, wenn man die Farben a und b_k in einer Komponente von $H(a, b_k)$ vertauscht (switcht).
Die Farbe b_k fehlt am Knoten y_k, so daß im Graphen $H(a, b_k)$ im Knoten y_k ein Pfad P beginnt (der ggf. nur aus dem Knoten y_k besteht, falls auch a an y_k fehlt). Sei z der andere Endpunkt des Pfades P. Da am Knoten x die Farbe a und am Knoten y_{i-1} die Farbe b_k fehlt, kommen insbesondere diese beiden Knoten für z in Frage. Beachte: P enthält keine der Kanten $\{x, y_j\}$, $1 \leq j \leq k$, bis auf $\{x, y_i\}$ im Fall $z = x$.

Fall 1: $z = y_{i-1}$
Beide Endkanten des Pfades P sind also mit a gefärbt. Switche die Farben a und b_k auf den Kanten des Pfades P. In der erhaltenen Färbung c' liegt dann die Farbe a an y_{i-1} nicht mehr an. Wir vollenden die Färbung durch FARBSHIFT $(i-1, a)$.

Fall 2: $z = x$
Nach Switchen von a und b_k entlang des Pfades P liegt die Farbe $b_k = b_{i-1}$ an x nicht mehr an. Wir vollenden die Färbung durch FARBSHIFT $(i-1, b_{i-1})$.

Fall 3: $z \notin \{y_{i-1}, x\}$
Nach Switchen der Farben a und b_k entlang P (sofern $E(P) \neq \emptyset$) liegt die Farbe a an y_k nicht mehr an. Wir vollenden die Färbung durch FARBSHIFT (k, a). □

Der Beweis des Satzes von VIZING war konstruktiv. Wir finden:

Korollar 5.6.6 *Es gibt einen $\mathcal{O}(mn)$-Algorithmus, um die Kanten eines Graphen G mit höchstens $\Delta(G) + 1 \leq \chi'(G) + 1$ vielen Farben zu färben.*

Beweis. Beachte, daß es bei einer Laufzeit von $\mathcal{O}(mn)$ nicht möglich ist, für alle $i = 1, \ldots, k$

jeweils zu testen, ob eventuell $F_x \cap F_{y_i} \neq \emptyset$ gilt.

Sei wieder $\Delta := \Delta(G)$. Es sind m Kanten zu färben. Pro Kante wird ein FARBSHIFT durchgeführt (Aufwand $\mathcal{O}(\Delta)$, wenn man sich neben den Knoten $y_1, \ldots, y_k$ auch die zugehörigen Kanten $\{x, y_1\}, \ldots, \{x, y_k\}$ merkt), und es müssen höchstens einmal der Pfad P berechnet und die Farben entlang P geswitcht werden (Aufwand $\mathcal{O}(n\Delta)$). Da die `REPEAT`-Schleife (pro Kante) höchstens Δ-mal durchlaufen wird, ist der Aufwand für alle restlichen Operationen (pro Kante) $\mathcal{O}(\Delta^2)$. Aus obigem Beweis des Satzes von VIZING ergibt sich also unmittelbar ein $\mathcal{O}(mn\Delta)$-Algorithmus, um die Kanten eines Graphen G mit höchstens $\Delta(G) + 1$ vielen Farben zu färben.

Mit Hilfe geeigneter Datenstrukturen gewinnt man eine $\mathcal{O}(mn)$-Implementation dieses Algorithmus. Dazu berechnet man eingangs ein Feld $Fehlfarbe : V \to \{1, \ldots, \Delta + 1\}$, das zu einem Knoten $v \in V$ eine Farbe $f \in F_v$ zurückliefert, und ein zweidimensionales Feld $Nachbar[v, f]$, $v \in V$, $1 \leq f \leq \Delta(G) + 1$, definiert durch

$$Nachbar[v, f] := \begin{cases} 0 & \text{falls } f \in F_v \\ u & \text{falls } u \in \Gamma(v) \text{ und } c(\{u, v\}) = f. \end{cases}$$

Beide Felder können in Zeit $\mathcal{O}(n^2)$ initialisiert werden. Ein weiteres Feld $Folgenglied : V \to \{0, \ldots, \Delta\}$ ist definiert durch:

$$Folgenglied[v] := \begin{cases} 0 & \text{falls } v \notin \{y_1, \ldots, y_k\} \\ i & \text{falls } v = y_i \end{cases}$$

und wird vor jedem Färben einer Kante auf 0 gesetzt. Mit Hilfe dieser drei Felder lassen sich alle Operationen der `REPEAT`-Schleife bis auf den (einmaligen) FARBSHIFT in konstanter Zeit erledigen und der Pfad P in Zeit $\mathcal{O}(n)$ berechnen und switchen. Beachte, daß auch das Feld $Fehlfarbe$ in Zeit $\mathcal{O}(n)$ pro Kante aktualisiert werden kann, da nur jeweils drei Werte von $Fehlfarbe[\cdot]$ wirklich neu berechnet werden müssen. □

Nach dem Satz von VIZING zerfällt die Menge aller Graphen in zwei Klassen.

Definition 5.6.7 *Ein Graph G heißt* „Klasse 1“, *falls* $\chi'(G) = \Delta(G)$, *sonst* „Klasse 2“.

Bipartite Graphen sind also „Klasse 1“ und vollständige Graphen und Kreise je nach Parität von n. Eine einfache Beobachtung ist:

Lemma 5.6.8 *Reguläre Graphen sind „Klasse 1“ genau dann, wenn sie 1-faktorisierbar sind.* □

Aus Gleichung (5.10) ergibt sich sofort das folgende nützliche Kriterium.

Lemma 5.6.9 (BEINEKE, WILSON 1973)
Ein Graph G mit $m > \Delta(G) \cdot \nu(G)$ vielen Kanten ist „Klasse 2“. □

Ein Graph G ungerader Ordnung mit $m > \Delta(G)\frac{n-1}{2}$ vielen Kanten heißt *übervoll*. Nach Lemma 5.6.9 ist ein solcher Graph trivialerweise „Klasse 2“. Vollständig multipartite Graphen sind „Klasse 2“ genau dann, wenn sie übervoll sind [HR92a]. Auch für die Klasse der kreisartig planaren Graphen können die Graphen der Klasse 2 charakterisiert werden, vergleiche Übung 6.1.35f.

Der Satz von VIZING hat die folgende Erweiterung.

Korollar 5.6.10 (VIZING 1964; FOURNIER 1973) *Sei $G = (V, E)$ ein Graph und $S \subseteq V$ die Menge der Knoten maximalen Grades in G, d.h. $S := \{v \in V \mid d(v) = \Delta(G)\}$. Falls der von S induzierte Subgraph $G[S]$ von G ein Wald (d.h. kreisfrei) ist, so ist G „Klasse 1". Eine entsprechende Kantenfärbung von G mit höchstens $\Delta(G)$ vielen Farben kann in Zeit $\mathcal{O}(mn)$ konstruiert werden.*

Beweis. Im Beweis zu Satz 5.6.3 wurde die $(\Delta(G)+1)$−te Farbe lediglich dazu benötigt, an den Knoten $x + \Gamma(x)$ stets mindestens eine Fehlfarbe zu garantieren. Die angegebene Bedingung garantiert diese Fehlfarbe, wie wir gleich sehen werden, auch im Fall von nur $\Delta(G)$ vielen Farben. Sei F die (kreisfreie) Kantenmenge von $G[S]$. Wir induzieren über $|F|$.
Induktionsverankerung: $F = \emptyset$
In diesem Fall ist die Knotenmenge S stabil in G. Wähle für jeden Knoten $x \in S$ eine mit ihm inzidierende Kante $\{x, y\} \in E$; Sei M, $|M| = |S|$, die Menge dieser Kanten. Nach dem Satz von VIZING können die Kanten des Graphen $G - M$ mit $\Delta(G - M) + 1 = \Delta(G)$ vielen Farben gefärbt werden. Füge nun sukzessive die Kanten $\{x, y\} \in M$ wieder hinzu, und färbe sie wie im Beweis vom Satz von VIZING: Da nach Voraussetzung keine zwei Knoten vom Grad $\Delta(G)$ benachbart sind, fehlt an x eine Farbe (da $\{x, y\}$ noch ungefärbt ist) und an allen Nachbarn von x (da sie Grad $< \Delta(G)$ haben).
Induktionsschluß: $|F| \geq 1$.
Sei $x \in S$ ein Blatt im Wald $G[S]$ und $\{x, y\} \in F$ die mit x inzidierende Waldkante. Nach Induktionsvoraussetzung kann $G - \{x, y\}$ mit $\Delta(G)$ vielen Farben kantengefärbt werden. Auch diese Kantenfärbung kann wieder wie im Beweis vom Satz von VIZING auf $\{x, y\}$ fortgesetzt werden, denn an x und y fehlt (genau) eine Farbe (da $\{x, y\}$ noch ungefärbt ist) und an allen übrigen Nachbarn von x (da sie in $V \setminus S$ liegen und somit Grad $< \Delta(G)$ haben).
Aus dem Beweis ergibt sich unmittelbar ein Algorithmus, um die $\Delta(G)$-Kantenfärbung zu konstruieren. Zunächst wird die Menge S bestimmt. Durch eine leicht modifizierte Breitensuche in den Komponenten von $G[S]$ berechnet man dann die Kantenmenge $F \cup M$ (bzw. überprüft, ob $G[F]$ überhaupt ein Wald ist). Nun wird in Zeit $\mathcal{O}(|E \setminus (F \cup M)| \cdot n)$ die $\Delta(G)$-Kantenfärbung von $G - (F \cup M)$ nach Korollar 5.6.6 konstruiert. Anschließend färbt man in Zeit $\mathcal{O}(|F \cup M| \cdot n)$ sukzessive die Kanten aus $F \cup M$ (in der Reihenfolge, wie sie gefunden und der Menge $F \cup M$ hinzugefügt wurden). Bis auf die zusätzlichen $\mathcal{O}(n+m)$ Schritte für die Bestimmung von S und $F \cup M$ ist also die Laufzeit dieses Algorithmus dieselbe wie die aus Korollar 5.6.6. □

Ohne Beweis zitieren wir in diesem Zusammenhang die beiden folgenden Ergebnisse. ERDŐS und WILSON [EW77] bewiesen, daß fast alle Graphen genau einen Knoten vom Grad $\Delta(G)$ besitzen, d.h. daß der Anteil der Graphen mit mehreren Knoten vom Grad $\Delta(G)$ unter allen Graphen auf n (numerierten) Knoten für $n \to \infty$ gegen 0 strebt. Der Kantenfärbungsalgorithmus aus Korollar 5.6.10 färbt aber alle Graphen mit genau einem Knoten vom Grad $\Delta(G)$ mit $\Delta(G)$ vielen Farben. Folglich färbt dieser Algorithmus die Kanten fast aller Graphen mit $\Delta(G)$ vielen Farben, und wir schließen insbesondere, daß fast alle Graphen „Klasse 1" sind. D.h. wenn wir den Anteil der „Klasse 2"-Graphen unter allen Graphen auf n (numerierten) Knoten mit p_n bezeichnen, so gilt $p_n \to 0$ für $n \to \infty$. Tatsächlich sind „Klasse 2"-Graphen sogar außerordentlich selten.

Satz 5.6.11 (FRIEZE, JACKSON, MCDIARMID, REED 1988) *Sei $\epsilon > 0$ beliebig. Dann gilt*

für n groß genug

$$n^{-(1/2+\epsilon)n} \;<\; p_n \;<\; n^{-(1/8-\epsilon)n}.$$

□

Dabei folgt die obere Schranke aus einer Verfeinerung des Kanten-Färbungsalgorithmus aus Korollar 5.6.10. Im Kontrast dazu zeigt das folgende Ergebnis, daß es NP-schwer ist, die wenigen restlichen Graphen, die der Algorithmus nicht mit $\Delta(G)$ vielen Farben kantenfärben konnte, kantenzufärben.

Satz 5.6.12 (HOLYER 1981b) *Das Problem* CHROMATIC INDEX, *zu einem gegebenen Graphen zu entscheiden, ob er „Klasse 1" oder „Klasse 2" ist, ist schon für kubische, brückenlose Graphen* NP*-vollständig.* □

Das Resultat läßt sich auf r-reguläre Graphen, $r \geq 3$ fest, verallgemeinern [GL83].

Gleichmäßige Kantenfärbung. In den Anwendungen sind Probleme oft mit zahlreichen Nebenbedingungen behaftet. Bei dem Problem, einen möglichst kurzen Prüfungszeitplan zu erstellen (siehe dazu die Einleitung dieses Abschnitts), könnte beispielsweise die Anzahl der zur Verfügung stehenden Räume begrenzt sein, so daß man möglichst wenige Prüfungen gleichzeitig abhalten möchte. Ins graphentheoretsiche Modell übersetzt bedeutet dies, daß man eine $\chi'(G)$-Färbung des Graphen sucht, in der alle Farbklassen möglichst gleich groß sind. Im Gegensatz zu dem analogen Problem bei Knotenfärbungen gibt es hier eine sehr einfache Lösung.

Proposition 5.6.13 (DE WERRA 1970; MCDIARMID 1972) *Sei G ein Graph und c eine (zulässige) Färbung seiner Kanten mit k Farben. Dann gibt es einen $\mathcal{O}(nm)$-Algorithmus, der eine Kantenfärbung c' von G mit k Farben konstruiert, in der sich die Farbklassen in ihrer Größe um höchstens 1 unterscheiden.*

Beweis. Seien $C_{\min} \subseteq E$ bzw. $C_{\max} \subseteq E$ zwei Farbklassen von c mit minimal vielen bzw. maximal vielen Kanten unter allen Farbklassen in der Kantenfärbung von G. Angenommen, es gilt $|C_{\max}| > |C_{\min}| + 1$. Dann enthält der von den Kanten $C_{\min} \cup C_{\max}$ gebildete Subgraph von G einen (alternierenden) Pfad ungerader Länge, der mehr $C_{\max}$−Kanten als $C_{\min}$−Kanten enthält. Vertausche die beiden Farben auf den Kanten eines solchen Pfades. Nach höchstens $m/2$ vielen solchen Schritten ist man bei der gesuchten Färbung c' angelangt.
Jeder Schritt läßt sich mit Hilfe eines Feldes $Buckets[1..n/2]$, dessen i-te Liste die Farbklassen mit genau i Kanten enthält, und der Datenstruktur $Nachbar[v,i]$ (s.o.) in Zeit $\mathcal{O}(n)$ bewerkstelligen. □

Übungen

Übung 5.6.14 *Man bestimme den chromatischen Index des* PETERSEN*-Graphen.*

Übung 5.6.15 *Sei G ein Δ-regulärer Graph, $\Delta \geq 2$. Hat G (i) ungerade Ordnung oder besitzt G (ii) eine Brücke oder (iii) einen Artikulationsknoten, so ist G „Klasse 2": $\chi'(G) = \Delta(G) + 1$.*

Übung 5.6.16 *Es nehmen n Spieler an einem Schachturnier teil, bei dem jeder Spieler eine Partie gegen jeden anderen spielt. Wenn kein Spieler mehr als eine Partie täglich spielt, in wievielen Tagen kann man das Turnier dann abhalten?*

Übung 5.6.17 *Für jeden Graph G gilt* $\nu(G) \geq m/(\Delta(G)+1)$.

Übung 5.6.18 *Für jedes maximale Matching M in einem Graphen G gibt es eine $(\Delta(G)+1)$−Kantenfärbung von G, die alle Kanten von M in derselben Farbe färbt.*

Ein Graph G heißt *eindeutig* $\chi'(G)$-kantenfärbbar, falls jede $\chi'(G)$-Kantenfärbung von G dieselbe Partition von E induziert.

Übung 5.6.19 (Greenwell, Kronk 1973)
a) *Ein kubischer, Hamiltonscher Graph ist „Klasse 1“.*
b) *Sei G kubisch und eindeutig $\chi'(G)$-kantenfärbbar. Man bestimme $\chi'(G)$ und* **c)** *die Anzahl der Hamiltonkreise in G.*

Übung 5.6.20 *In einer eindeutigen $\chi'(G)$-Kantenfärbung*
a) *inzidiert jede Kante mit Kanten in allen anderen Farben;*
b) *unterscheiden sich die Größen verschiedener Farbklassen um höchstens 1.*

Übung 5.6.21 Der Greedy-Kantenfärbungsalgorithmus, der $\Delta(G)+1$ viele disjunkte maximale Matchings in G konstruiert und alle Kanten je eines maximalen Matchings in einer neuen Farbe färbt, und der lokale Verbesserungsalgorithmus, der sukzessive jede Kante $e = \{u, v\} \in E$ mit einer Farbe $c(e) \in F_u \cap F_v$ färbt, falls dies möglich ist, können mit Laufzeit $\mathcal{O}(\Delta(G)m)$ implementiert werden. Man zeige: *Beide Algorithmen färben mindestens $m/2$ viele Kanten von G zulässig.* Bei Graphen G mit $\Delta(G) = O(1)$ kann man also die ersten $m/2$ vielen Kanten in konstanter Zeit pro Kante färben.

5.7 ⋆ Kantenfärben in bipartiten Graphen

Ähnlich wie beim Matching-Problem gestaltet sich auch das Problem des Kantenfärbens einfacher auf der Klasse der bipartiten Graphen. Oben hatten wir einen $\mathcal{O}(mn)$-Algorithmus für das bipartite Kantenfärbungsproblem kennengelernt. In diesem Abschnitt wird der effizientere $\mathcal{O}(m(\log n)^2)$-Algorithmus von Gabow und Kariv [GK82] für das bipartite Kantenfärbungsproblem dargestellt. Seine Technik, einen Graphen rekursiv in Subgraphen zu zerlegen, ist ebenso instruktiv wie seine Analyse. Zudem ist dieser Algorithmus dadurch von Bedeutung, daß er leicht parallelisierbar ist, siehe [LPV81, GR88]. Im Anschluß daran studieren wir ein brandneues Ergebnis von Galvin [Gal95] über eine Verallgemeinerung des Kantenfärbungsproblems: bipartite Graphen sind sogar $\Delta(G)$-*edge-choosable*, d.h. für jedwede Vorgabe von Listen $L(e)$, $e \in E$, von je mindestens $\Delta(G)$ Farben an den Kanten des Graphen gibt es eine Kantenfärbung $\ell : E \to \bigcup_{e \in E} L(e)$ von G, so daß $\ell(e) \in L(e)$ für alle $e \in E$.

Der Algorithmus von Gabow und Kariv. Dieser Kantenfärbungsalgorithmus basiert darauf, daß bipartite Graphen, deren Maximalgrad eine Potenz von 2 ist, besonders effizient kantengefärbt werden können. Wir betrachten daher zunächst diesen Spezialfall. Die Aufgabe besteht also nun darin, einen bipartiten Graphen G mit $\Delta(G) = 2^k$, $k \in \mathbb{N}$, vielen Farben kantenzufärben. Die Idee des Algorithmus ist ein divide-et-impera Ansatz: Der Graph (bzw. seine Kantenmenge) wird in zwei Subgraphen partitioniert, so daß sich der Maximalgrad in jedem der beiden Subgraphen gegenüber dem Originalgraphen halbiert, die Menge der zur Verfügung stehenden Farben wird in zwei gleichgroße Mengen partitioniert und dann wird jeder der beiden entstandenen Subgraphen rekursiv mit den Farben aus je einer der beiden Mengen gefärbt. Da $\Delta(G)$ gerade eine Zweierpotenz ist,

ist dieses Halbieren der Knotengrade und der Menge der Farben stets möglich, und die schließlich konstruierte Kantenfärbung ist automatisch zulässig.

Die Frage ist nun, wie man die Partition des Graphen in zwei derartige Subgraphen findet. Dies ist aber weiter auch kein großes Problem: Falls G Eulersch ist, so berechnet man leicht in einer Laufzeit, die linear in der Anzahl der Kanten des Graphen ist, eine Zerlegung der Kantenmenge in (da G bipartit ist) *gerade* Zykeln. Falls G hingegen $2q$ Knoten ungeraden Grades enthält, so berechnet man analog eine Zerlegung der Kantenmenge in q Wege, die genau die $2q$ Knoten ungeraden Grades als Endknoten haben, und gerade Zykeln. Knoten vom Grad $\Delta(G)$ treten offenbar in den Wegen nur als innere Knoten auf; wenn man daher die Kanten der q Wege und der Zykeln abwechselnd dem einen bzw. dem anderen Subgraphen zuschlägt, so halbiert sich der Maximalgrad in beiden Subgraphen genau. Eine Zerlegung eines bipartiten Graphen in zwei solche Subgraphen heißt auch Euler-Partition. Der Algorithmus hat damit die folgende Gestalt, wenn man die Farbe c_e einer Kante $e \in E$ als k-stelligen Bitvektor $(c_e[k], \ldots, c_e[1]) \in \{0,1\}^k$ darstellt (o.B.d.A. sei der Maximalgrad $\Delta(G) = 2^k > 1$):

```
PROCEDURE EULER_KANTENFÄRBUNG (G, k);

BEGIN
  · zerlege die Kantenmenge von G in Zykeln und q (offene) Wege;
  · schlage die Kanten in den Wegen und Zykeln abwechselnd
    dem Subgraphen G_0 bzw. G_1 von G zu, indem man für
    jede Kante e ∈ E das k-te Bit von c_e auf 0 bzw. 1 setzt;
  IF k > 1 THEN BEGIN
    EULER_KANTENFÄRBUNG (G_0, k − 1);
    EULER_KANTENFÄRBUNG (G_1, k − 1);
  END;
END; {Euler_Kantenfärbung}
```

Lemma 5.7.1 (GABOW 1976) *Der Algorithmus* EULER_KANTENFÄRBUNG *kantenfärbt einen bipartiten Graphen G vom Maximalgrad $\Delta(G) = 2^k$, $k \in \mathbb{N}$, in Zeit $\mathcal{O}(m \log n)$.*

Beweis. Wenn man einen Graphen jeweils als eine Liste von Kanten (aus der man sich dann wiederum eine Adjazenzlisten-Darstellung berechnen kann) codiert und an die Prozedur EULER_KANTENFÄRBUNG übergibt, so kann jeder Aufruf der Prozedur EULER_KANTENFÄRBUNG offenbar in Zeit linear in der Anzahl der Kanten des Graphen G bearbeitet werden.
Die verschiedenen rekursiven Aufrufe der Prozedur EULER_KANTENFÄRBUNG bilden einen Binärbaum mit $k = \log \Delta(G) \leq \log n$ Ebenen. Auch wenn ein Aufruf des Algorithmus die Knoten des Binärbaums tiefensuchen-artig abarbeitet, zählen wir die insgesamt durchgeführten Operationen ebenenweise. Jede Ebene des Binärbaums enthält die Kantenmenge von G auf verschiedene Subgraphen verteilt, die den Knoten des Binärbaums auf dieser Ebene entsprechen. Wenn also die Operationen für jeden Knoten linear in der Anzahl der Kanten des zugehörigen Subgraphen sind, so ist die Anzahl aller Operationen für eine Ebene linear in m. Da höchstens $\log n$ Ebenen zu bearbeiten sind, ist die Gesamtzahl von Operationen $\mathcal{O}(m \log n)$. □

Man beachte, daß die rekursive Programmierung relativ speicheraufwendig ist; tatsächlich kommt man mit linear viel Speicherplatz aus (Übung).

Der Kantenfärbungsalgorithmus für einen allgemeinen bipartiten Graphen $G = (V, E)$ (d.h. $\Delta(G)$ ist nicht notwendig eine Zweierpotenz) benutzt die Prozedur EULER_KANTENFÄRBUNG als Subroutine. Sei 2^k die größte Zweierpotenz, die in $\Delta(G)$ enthalten ist, d.h. sei $k \in \mathbb{N}$ derart, daß $2^k \leq \Delta(G) < 2^{k+1}$ oder $k = \lfloor \log \Delta(G) \rfloor$. Die Kunst besteht darin, jeweils einen Subgraphen S von G zu finden, der viele noch ungefärbte Kanten enthält (d.h. genauer: mindestens einen stets konstanten Anteil aller noch ungefärbten Kanten) und Maximalgrad höchstens 2^k hat. Wenn man dann EULER_KANTENFÄRBUNG 2^k Farben zur Verfügung stellt, mit denen EULER_KANTENFÄRBUNG die Kanten von S unabhängig vom restlichen Graphen färben kann, so ist, wie wir sehen werden, schon nach nur logarithmisch vielen solchen Schritten der gesamte Graph kantengefärbt. Beachte: der konstruierte Subgraph S von G hat nicht notwendig Maximalgrad genau 2^k – wenn der Maximalgrad jedoch tatsächlich kleiner als 2^k ist, so kann EULER_KANTENFÄRBUNG den Subgraphen S „erst recht“ mit 2^k Farben zulässig färben – zwar nicht optimal, aber zulässig. Der Algorithmus hat folgende Struktur:

```
PROCEDURE GK_KANTENFÄRBUNG (G);

BEGIN
  U := E; {Menge der ungefärbten Kanten}
  WHILE U ≠ ∅ DO BEGIN
    FOR v ∈ V DO BEGIN
      · markiere alle mit v inzidenten, noch ungefärbten Kanten
        mit einer Farbe, so daß unter allen mit v inzidenten Kanten
        keine Farbe mehrfach (als Farbe oder Marke) auftritt;
    END;
    {jede ungefärbte Kante trägt nun zwei Marken α und β}
    · finde eine Menge C von 2^k Farben, so daß für
      mindestens 1/6 aller ungefärbten Kanten gilt: {α,β} ⊆ C;
    · sei S = (V,F) der Subgraph von G, der alle bereits in einer
      Farbe c ∈ C gefärbten Kanten sowie alle ungefärbten Kanten
      mit {α,β} ⊆ C enthält;
    {nach Konstruktion hat S Maximalgrad Δ(S) ≤ 2^k}
    · entfärbe alle in einer Farbe c ∈ C gefärbten Kanten e ∈ F;
    {Euler-färbe S mit den Farben aus C:}
    EULER_KANTENFÄRBUNG (S,C);
    U := U \ F;
  END; {while}
END; {GK_Kantenfärbung}
```

Satz 5.7.2 (GABOW, KARIV 1982) *Der Algorithmus* GK_KANTENFÄRBUNG (G) *kantenfärbt einen bipartiten Graphen G zulässig in Zeit $\mathcal{O}(m(\log n)^2)$.*

Beweis. Wir zeigen zunächst, daß jeder Schritt innerhalb der `WHILE`-Schleife in Zeit $\mathcal{O}(m \log n)$ bearbeitet werden kann, und dann, daß die `WHILE`-Schleife nur $\mathcal{O}(\log n)$-mal durchlaufen wird.

Für die Markierung der noch ungefärbten Kanten hält man sich ein Boolesches Feld $Farben[1, \ldots, \Delta(G)]$, das die zur Verfügung stehenden Farben repräsentiert. Das Feld *Farben* wird zu Anfang mit 1 initialisiert. Für jeden Knoten $v \in V$ geht man nun die Adjazenzliste von v einmal durch und streicht alle Farben, die an mit v inzidierenden

Kanten auftreten, aus *Farben*. Dann geht man die Adjazenzliste von v ein weiteres Mal durch und markiert die noch nicht gefärbten, mit v inzidierenden Kanten sukzessive mit noch zur Verfügung stehenden Farben aus dem Feld *Farben*. Schließlich geht man die Adjazenzliste ein letztes Mal durch und setzt das Feld *Farben* wieder auf 1 zurück. Der Aufwand pro Knoten ist also $\mathcal{O}(d(v))$, was sich über alle Knoten zu $\mathcal{O}(m)$ summiert.

Die Bestimmung der Farbenmenge C. O.B.d.A. sei zunächst $\Delta(G) \geq 4$ [der Fall $\Delta(G) \leq 2$ ist trivial, im Fall $\Delta(G) = 3$ geht man analog vor]. Wir partitionieren die Menge der zur Verfügung stehenden Farben $1, \ldots, \Delta(G)$ in vier Mengen gemäß der letzten 2 Bits einer jeden Farbe (entspricht Rechnung modulo 4). Dann unterscheiden sich die so definierten vier Farbmengen C_{00}, C_{01}, C_{10} und C_{11} in ihrer Größe paarweise höchstens um 1. Aus diesen vier Farbmengen lassen sich durch Vereinigung von jeweils zweien $\binom{4}{2} = 6$ verschiedene Vereinigungmengen $C_1, \ldots, C_6$ von Farben bilden. Wir behaupten nun, daß jede dieser Mengen C_i, $1 \leq i \leq 6$, höchstens 2^k viele Farben enthalten kann. Dies gilt aber wegen

$$\begin{aligned} 2|C_i| &\leq \Delta(G) + 2 \\ \Rightarrow\ |C_i| &\leq \frac{\Delta(G)}{2} + 1 < \frac{2^{k+1}}{2} + 1 = 2^k + 1 \end{aligned}$$

Wir ordnen nun jede noch ungefärbte Kante genau einer dieser 6 Farbmengen C_i zu, so daß die beiden Marken $\{\alpha, \beta\}$ der Kante in der zugeordneten Farbmenge enthalten sind. Falls die beiden Marken α und β einer ungefärbten Kante e in zwei verschiedenen der 4 Farbmengen C_{00}, C_{01}, C_{10} und C_{11} liegen, also $\alpha \in C_{i_1,j_1}$ und $\beta \in C_{i_2,j_2}$ mit $i_1 \neq i_2$ oder $j_1 \neq j_2$, so ordnen wir e der Vereinigungsmenge $C_{i_1,j_1} \cup C_{i_2,j_2}$ zu; falls andernfalls $\{\alpha, \beta\} \subseteq C_{ij}$ gilt, so ordnen wir e der Vereinigungsmenge $C_{i,0} \cup C_{i,1}$ zu. Damit sind die noch ungefärbten Kanten von G auf die 6 Mengen $C_1, \ldots, C_6$ aufgeteilt worden, von denen mithin mindestens eine mindestens 1/6 aller noch ungefärbten Kanten enthält. Wir wählen also diejenige Farbenmenge C' unter $C_1, \ldots, C_6$ aus, die die meisten noch ungefärbten Kanten erhalten hat, ergänzen gegebenenfalls beliebig weitere Farben, bis die Menge genau 2^k Farben enthält und verwenden die so erhaltene Farbenmenge als die Menge C im Algorithmus. Offensichtlich benötigt die Bestimmung der Menge C' lediglich ein Scannen der noch ungefärbten Kanten in G (Aufwand $\mathcal{O}(n+m)$) und die Bestimmung der Menge C höchstens $\mathcal{O}(n)$ weitere Operationen.

Die Berechnung des Subgraphen S sowie das Entfärben der Kanten von G, die in einer Farbe $c \in C$ gefärbt sind, benötigt offenbar nur Zeit $\mathcal{O}(n+m)$.

Die Prozedur EULER_KANTENFÄRBUNG ist lediglich dahingehend zu ändern, daß auf die zur Verfügung stehenden Farben via des Feldes $C[1, \ldots, 2^k]$ zugegriffen wird und die rekursiven Aufrufe (statt von der Abfrage $k > 1$?) davon abhängig gemacht werden, ob $\Delta(G_0)$ bzw. $\Delta(G_1)$ größer als 1 ist. Dieser Schritt benötigt, wie in obigem Lemma gesehen, $\mathcal{O}(m \log n)$ Zeit.

Um einzusehen, daß die `WHILE`-Schleife nur $\mathcal{O}(\log n)$-mal durchlaufen wird, bemerken wir, daß in jedem (Um-) Färbungsschritt EULER_KANTENFÄRBUNG (S, C) nach Wahl von C' mindestens 1/6 aller bis dahin noch ungefärbten Kanten von G kantengefärbt werden. (Die Zulässigkeit dieser neuen Färbung liegt darin begründet, daß nach Definition von S außerhalb von F keine Kante eine Farbe $c \in C$ trägt.) Bezeichne $U_0 := E$ und U_i die Menge der ungefärbten Kanten nach $i \in \mathbb{N}$ Durchläufen der `WHILE`-Schleife. Dann gilt also

$|U_0| = m$, $|U_1| < \frac{5}{6}m$ und allgemein $|U_i| < \left(\frac{5}{6}\right)^i m$. Wegen

$$\begin{array}{rcl} \left(\frac{5}{6}\right)^i m & < & 1 \\ \Leftrightarrow \quad i & > & \frac{\log m}{\log 6/5} \end{array}$$

gilt nach spätestens

$$i_0 \;=\; \left\lceil \frac{\log m}{\log 6/5} \right\rceil \;\leq\; 8 \log n$$

Durchläufen durch die `WHILE`-Schleife $|U_{i_0}| = 0 \Leftrightarrow U_{i_0} = \emptyset$. □

Einen $\mathcal{O}(m \log n)$-Kantenfärbungsalgorithmus für bipartite Graphen findet man in [CH82].

Edge-Choosability. Eine Listen-Kantenfärbung eines Graphen $G = (V, E)$ ist eine (zulässige) Färbung seiner Kanten, wobei für jede Kante $e \in E$ eine Liste $L(e)$ von Farben vorgegeben ist, in der die Kante e gefärbt werden darf. Beachte, daß das Problem trivial wird, wenn die Farblisten paarweise disjunkt sind; sind die Listen andererseits alle identisch, so haben wir es mit der gewöhnlichen Kantenfärbung zu tun. Ein Graph $G = (V, E)$ heißt *k-edge-choosable*, wenn für *jede* Vorgabe von Listen $L(e)$ an den Kanten $e \in E$ des Graphen, die nur jeweils mindestens k viele (verschiedene) Farben enthalten, eine Listen-Kantenfärbung von G existiert. Trivialerweise ist jeder Graph $(2\Delta(G)-1)$-edge-choosable. Die kleinste Zahl k, so daß G k-edge-choosable ist, heißt der *Listen-chromatische Index* von G, symbolisch $\chi'_\ell(G)$. Eine Vermutung von Vizing [Viz76] besagt, daß stets $\chi'_\ell(G) = \chi'(G)$ gilt. Sie ist trivialerweise erfüllt für Bäume und Graphen vom Maximalgrad ≤ 2. Trotz großer Anstrengungen konnten im allgemeinen nur starke Abschwächungen dieser Vermutung gezeigt werden. Selbst für den vollständig bipartiten Graphen $K_{n,n}$ (die sogenannte Dinitz-Vermutung [Din79]) konnten über die Jahre nur Abschwächungen gezeigt werden, bis Galvin 1995 die Vermutung von Vizing gleich für alle bipartiten Graphen beweisen konnte.

Satz 5.7.3 (Galvin 1995) *Bipartite Graphen sind $\Delta(G)$-edge-choosable.*

Wir orientieren uns hier an Ideen von Slivnik [Sli96]. Statt obigen Satz direkt zu zeigen, beweisen wir eine etwas allgemeinere Aussage, die für eine Induktion besser geeignet ist. Gegeben ein bipartiter Graph $G = (L \dot\cup R, E)$ und eine Funktion $f : E \to \mathbb{N}$, so sagen wir, G sei *f-edge-choosable*, wenn für jede Vorgabe von Listen $L(e)$ an den Kanten $e \in E$ des Graphen, so daß für alle $e \in E$ gilt: $|L(e)| \geq f(e)$, eine Listen-Kantenfärbung von G existiert. e_L bzw. e_R bezeichne den Endknoten der Kante $e \in E$ in L bzw. R.

Lemma 5.7.4 *Sei $G = (L, R, E)$ ein bipartiter Graph und $c : E \to \mathbb{N}$ eine (zulässige) Kantenfärbung von G. Definiere eine Funktion $T_{G,c} : E \to \mathbb{N}_0$ durch:*

$$T_{G,c}(e) \;=\; |\{f \in E | (f_L = e_L) \wedge (c(f) > c(e))\}| + |\{f \in E | (f_R = e_R) \wedge (c(f) < c(e))\}|.$$

Dann ist G $(T_{G,c} + 1)$-edge-choosable.

Der Satz 5.7.3 von Galvin leitet sich daraus wie folgt ab:

Beweis von Satz 5.7.3:
Sei $c : E \to \{1, \ldots, \chi'(G)\}$ eine optimale (zulässige) Kantenfärbung des bipartiten Graphen G. Wegen

$$T_{G,c}(e) \;\leq\; (\chi'(G) - c(e)) \;+\; (c(e) - 1) \;\leq\; \chi'(G) - 1$$

ist G nach dem Lemma ($T_{G,c} + 1 \leq \chi'(G) = \Delta(G)$)-edge-choosable. □

Beweis von Lemma 5.7.4:
Die (zulässige) Kantenfärbung c von G aus Lemma 5.7.4 wird lediglich benötigt, um auf allen mit einem Knoten $v \in L \dot\cup R$ inzidenten Kanten eine Ordnung zu definieren; sie muß daher nicht notwendig optimal sein.
Wir induzieren nach $|E(G)|$. Für $|E(G)| = 0$ ist offenbar nichts zu zeigen. Sei also $E(G) \neq \emptyset$. Gegeben Listen $L(e)$ von Farben an den Kanten $e \in E$ von G mit $|L(e)| \geq T_{G,c}(e)+1$, so ist nun zu zeigen, daß es eine Listen-Kantenfärbung $\ell : E \to \bigcup L(e)$ von G mit $\ell(e) \in L(e)$ für alle $e \in E$ gibt.
Wähle eine Farbe $a \in \bigcup_{e \in E} L(e)$ und setze $E_a := \{e \in E : \ a \in L(e)\}$. Wir werden weiter unten ein Matching M im Subgraphen (V, E_a) von G konstruieren, so daß für jede Kante $e \in E_a \setminus M$ gilt:

- entweder wird $e \in E_a \setminus M$ durch eine Kante $f \in M$ *links-dominiert*, d.h. es gilt $f_L = e_L$ und $c(f) > c(e)$;
- oder aber $e \in E_a \setminus M$ wird durch eine Kante $f \in M$ *rechts-dominiert*, d.h. es gilt $f_R = e_R$ und $c(f) < c(e)$.

Mit Hilfe des Matchings M geht der Beweis wie folgt zu Ende: Wir färben die Kanten $e \in M$ mit der Farbe a, entfernen die Kanten von M aus E, entfernen die Farbe a aus allen verbleibenden Listen $L(e)$ und wenden die Induktionsvoraussetzung an, um die restlichen Kanten zu färben. Offensichtlich erhält man eine zulässige Listen-Kantenfärbung von G. Es bleibt, zu prüfen, daß die Induktionsvoraussetzung auch tatsächlich anwendbar ist. Setze $G' := (V, E \setminus M)$ und $L'(e) := L(e) - a$ für alle $e \in E(G')$. Dann gilt

$$\begin{array}{lcllllll}
|L'(e)| & = & |L(e)| & \geq & T_{G,c}(e) + 1 & \geq & T_{G',c}(e) + 1 & \text{falls } e \in E \setminus E_a, \\
|L'(e)| & = & |L(e)| - 1 & \geq & T_{G,c}(e) & \geq & T_{G',c}(e) + 1 & \text{falls } e \in E_a \setminus M,
\end{array}$$

wobei die letzte Ungleichung gilt, weil für eine Kante $e \in E_a \setminus M$ die e links- oder rechts-dominierende Kante $f \in M$ nun bei $T_{G',c}(e)$ nicht mehr mitzählt.

Es bleibt, das Matching $M \subseteq E_a$ zu konstruieren. Eine Kante $e \in E_a$ heiße *links-maximal* (bezüglich c), falls es keine Kante $f \in E_a$ gibt mit $f_L = e_L$ und $c(f) > c(e)$. Eine links-maximale Kante $e \in E_a$ links-dominiert offenbar alle anderen mit dem Knoten e_L inzidierenden Kanten in E_a. Das folgende Programmstück wählt daher sukzessive links-maximale Kanten aus E_a aus und fügt sie M hinzu unter Rücksichtname darauf, daß M ein Matching bleibt.

```
M := ∅;
WHILE ∃ links-maximale Kante e ∈ E_a \ M DO BEGIN
  IF ∃f ∈ M : (f_R = e_R) ∧ (c(f) < c(e)) THEN BEGIN
    {e wird durch f ∈ M rechts-dominiert}
    E_a := E_a − e;
  END
  ELSE BEGIN
    {e wird noch durch keine Kante f ∈ M dominiert}
    M := M + e;
    {ist M noch ein Matching?}
    IF ∃f ∈ M : (f_R = e_R) ∧ (c(f) > c(e)) THEN BEGIN
      {f wird nun durch e rechts-dominiert}
      E_a := E_a − f;
      M := M − f;
    END;
  END; {else}
END; {while}
```

Da die Relation „e wird von f rechts-dominiert“ transitiv ist, wird eine Kante e, die zu einem Zeitpunkt von einer anderen Kante $f \in M$ rechts-dominiert wird, für immer von irgendeiner Kante rechts-dominiert werden, denn eine Kante wird nur aus M entfernt, wenn sie von einer anderen Kante aus M rechts-dominiert wird. Also werden nach Verlassen der WHILE-Schleife alle bis dahin aus E_a entfernten Kanten von Kanten aus M rechts-dominiert. Da aber nach Verlassen der WHILE-Schleife alle links-maximalen Kanten zu M gehören, werden alle noch in $E_a \setminus M$ verbliebenen Kanten links-dominiert.
Die WHILE-Schleife terminiert, weil der Wert $|E_a| - |M|$ in jedem Durchlauf um mindestens 1 abnimmt. Da zu Beginn $|E_a| - |M| \leq m$, andererseits jedoch $|E_a| - |M| \geq 0$ gilt, wird die WHILE-Schleife nach spätestens m Durchläufen verlassen. □

Kapitel 6

Topologische Graphentheorie

6.1 Planare Graphen

Die topologische Graphentheorie nahm ihren Ursprung im Jahre 1750, als EULER seine berühmte Polyederformel entdeckte. Sie ruhte dann wieder für 180 Jahre und erwachte erst wieder zu neuem Leben mit KURATOWSKIs Charakterisierung planarer Graphen durch verbotene (verallgemeinerte) Subgraphen. Seitdem sind vielfältige Resultate über planare Graphen erzielt worden. Das berühmteste ist wohl der Vier-Farben-Satz: jede Landkarte kann so mit vier Farben gefärbt werden, daß Länder (als zusammenhängend vorausgesetzt) mit einer gemeinsamen Grenzlinie verschiedene Farben tragen. Die mannigfaltigen Ansätze, diesen Satz zu beweisen, haben der Graphentheorie wichtige Impulse gegeben.

In letzter Zeit galt das Interesse verstärkt dem algorithmischen Aspekt der planaren Graphen und Graphen höheren Geschlechts, siehe z.B. [WKC88]. So gibt es eine Reihe von NP-schweren Problemen, die für planare Graphen polynomiell lösbar werden. Hierzu zählen beispielsweise die folgenden Probleme:

- einen maximalen Schnitt in einem Kanten-gewichteten Graphen zu finden (siehe Abschnitt 10.6);
- einen minimum Multiterminal-Cut (d.h. eine k vorgegebene Knoten paarweise trennende Kantenmenge) zu finden für $k \geq 3$ fest (siehe [DJPSY94]);
- das EDGE-DISJOINT PATHS Problem (gegeben ein Graph G mit Knotenpaaren $(s_1, t_1), \ldots, (s_k, t_k)$ – enthält G eine Menge paarweise kantendisjunkter $s_i - t_i$–Pfade, $i = 1, \ldots, k$?) ist in planaren Eulerschen Graphen polynomiell lösbar [Sey81], während es für allgemeine Eulersche Graphen NP-vollständig ist [MP93a];
- die Anzahl der perfekten Matchings in einem Graphen zu bestimmen (siehe [Val79] bzw. [Kas67, Lit74] und [Ber85, Chapter 7.5]);
- eine Approximation an eine maximum stabile Menge in einem Graphen zu finden. INDEPENDENT SET bleibt auch für planare Graphen NP-schwer [GJS76, Lich82], doch während es für allgemeine Graphen nicht einmal approximierbar ist, falls P $\neq$ NP (vgl. Satz 11.2.1), läßt es sich im Fall planarer Graphen beliebig gut approximieren (vgl. Satz 10.8.5);

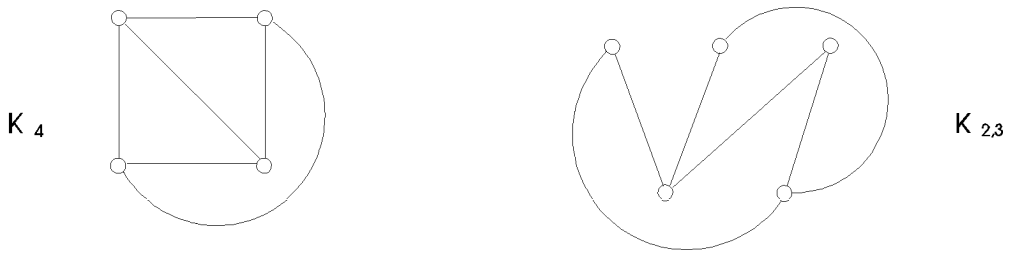
- das Färbungsproblem ist zwar auch auf planaren Graphen NP-schwer, doch kann man einen planaren Graphen in linearer Zeit mit höchstens zwei Farben mehr als nötig färben (vgl. Übung 6.3.6).

Ferner lassen sich planare Graphen [HW74] (bzw. allgemeiner Graphen beschränkten Geschlechts [Mil80]) in polynomieller Zeit auf Isomorphie testen.
Voraussetzung für solche Algorithmen ist in der Regel, daß eine Einbettung des planaren Graphen in die Ebene vorliegt. Hierauf gehen wir in Abschnitt 6.2 ein.

6.1.1 Ebene Graphen

Definition 6.1.1 *Ein Graph G heißt* planar *(oder plättbar), falls es eine* Einbettung *von G in die Ebene gibt, d.h. eine Darstellung von G in der Ebene, in der die Knoten von G durch Punkte und die Kanten von G durch Jordankurven repräsentiert werden, so daß jede Jordankurve genau ihre Endpunkte mit der Menge der Knotenpunkte gemeinsam hat und sich zwei verschiedene Kanten höchstens in einem gemeinsamen Knoten schneiden. Ein* ebener *Graph ist ein in die Ebene eingebetteter, planarer Graph.*

Offensichtlich sind beispielsweise Bäume planar. Die folgende Abbildung zeigt, daß die Graphen K_4 und $K_{2,3}$ planar sind:



Da die Ebene vermittels stereographischer Projektion homöomorph ist zur Oberfläche einer Kugel, aus der ein Punkt (der Nordpol als Projektionszentrum) entfernt wurde, ist Planarität gleichbedeutend mit Einbettbarkeit in die Kugeloberfläche.
Bei Herausnahme der Kanten eines ebenen Graphen zerfällt die Ebene in einfach-zusammenhängende *Gebiete* R (für engl. regions), von denen genau eines unbeschränkt ist.[1] Dieses wird das äußere Gebiet genannt. Der *Rand eines Gebietes* sind alle die Kanten, die im Abschluß des Gebietes liegen. Offensichtlich kann man einen ebenen Graphen stets so in die Ebene einbetten, daß ein vorgegebenes Gebiet zum äußeren wird (man drehe die Kugel, bis der Nordpol in dieses Gebiet fällt). Wir schließen daraus:

Proposition 6.1.2
Ein Graph ist planar genau dann, wenn jeder seiner Blöcke planar ist.

Beweis. Man paßt die Einbettungen eines jeden Blockes von G gemäß des Block-Artikulations-Baumes $bc(G)$ (vgl. Satz 2.3.8) aneinander an, indem man zunächst einen beliebigen Block einbettet und dann $bc(G)$ gemäß einer Breiten- oder Tiefensuche durchläuft und

[1] Hier geht essentiell der sogenannte Jordansche Kurvensatz ein, der besagt, daß für eine geschlossene Kurve C in der Ebene, $\mathbb{R}^2 \setminus C$ genau zwei einfach-zusammenhängende Gebiete hat, von denen genau eines unbeschränkt ist. Obwohl diese einfache Aussage als ein recht schwieriger Satz in der Topologie gilt, gibt es einen knappen graphentheoretischen Beweis, siehe [Tho92].

jeden Block von G, auf den man stößt, so einbettet, daß der bereits eingebettete Artikulationsknoten, über den man den Block betrat, auf den Rand des äußeren Gebietes zu liegen kommt. □

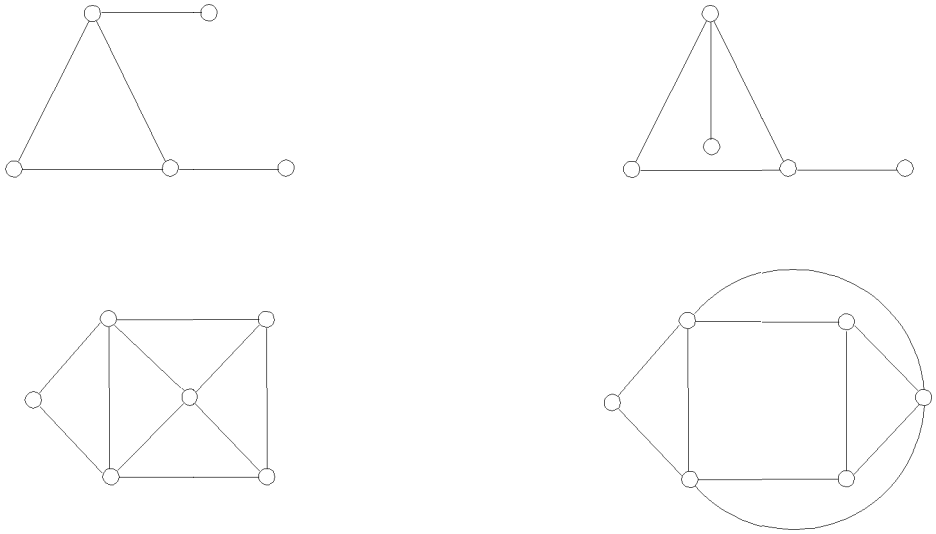
Zweifach zusammenhängende, planare Graphen lassen sich hübsch charakterisieren:

Übung 6.1.3 *Ein ebener Graph ist genau dann zweifach zusammenhängend, wenn jedes seiner Gebiete durch einen Kreis berandet wird.*

Im allgemeinen gibt es viele verschiedene Möglichkeiten, einen planaren Graphen in die Ebene einzubetten. Wir definieren auf der Menge der Einbettungen eines planaren Graphen eine Äquivalenzenrelation:

Definition 6.1.4 *Es seien φ und φ' zwei Einbettungen eines planaren Graphen $G = (V, E)$ in die Ebene, d.h. Graphen-Isomorphismen von G auf zwei ebene Graphen H und H'. φ und φ' heißen* äquivalent, *falls $\varphi' \circ \varphi^{-1}$ ein Isomorphismus von H auf H' derart ist, daß für alle Kantenfolgen $R \in E^*$ gilt: $\varphi(R)$ ist Rand eines Gebiets in H genau dann, wenn $\varphi'(R)$ in derselben oder in umgekehrter Reihenfolge Rand eines Gebiets in H' ist.*

D.h. der Isomorphismus $\varphi' \circ \varphi^{-1}$ bezieht auch die Gebiete der beiden ebenen Graphen H und H' bijektiv aufeinander, so daß ihre Ränder modulo des Drehsinns übereinstimmen. Die Äquivalenzklassen unter den Einbettungen eines planaren Graphen G bestehen dann gerade aus den Einbettungen, die durch einen Homöomorphismus der Kugeloberfläche ineinander überführbar sind.
Nachfolgend sind für zwei Graphen nicht äquivalente Einbettungen angegeben.



Daß die beiden abgebildeten Einbettungen jeweils nicht äquivalent sind, erkennt man leicht daran, daß die linke Einbettung jeweils ein Gebiet besitzt, das durch fünf Kanten berandet ist, während dies bei der rechten Einbettung nicht der Fall ist. Der Graph auf Seite 178 unten besitzt genau drei nicht äquivalente Einbettungen.

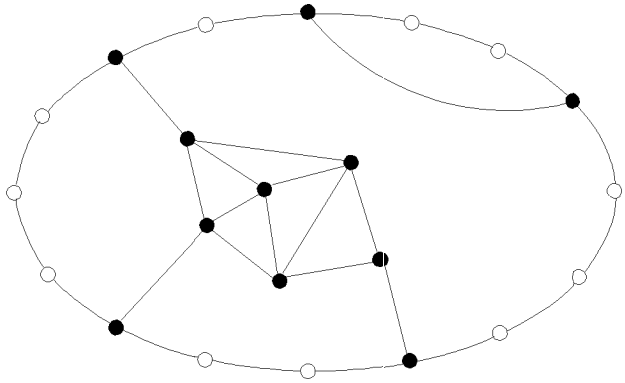
Das folgende Resultat wird in Übung 6.1.32b bewiesen.

Satz 6.1.5 (Wagner 1936; Fáry 1948) *Zu jeder Einbettung eines planaren Graphen in die Ebene gibt es eine äquivalente gradlinige, d.h. die Jordankurven in einer Einbettung eines planaren Graphen in die Ebene können als Geradenstücke gewählt werden.* □

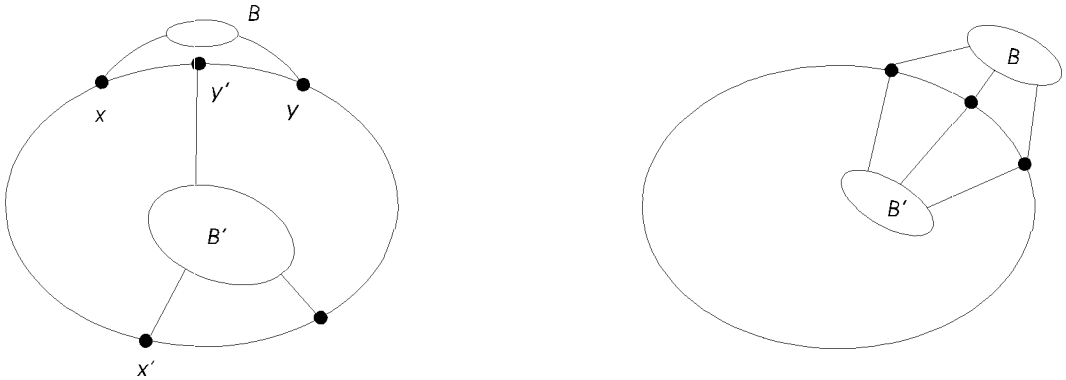
Die Einbettungen eines Pfads P_n, eines Kreises C_n oder auch des $K_{2,3}$ sind offenbar jeweils alle äquivalent - man sagt, solche Graphen sind *eindeutig einbettbar*. Der nachfolgende Satz des Topologen WHITNEY [Whi32b] zeigt, daß die Einbettung eines planaren Graphen in die Ebene bei genügend hohem Zusammenhang stets eindeutig ist.

Satz 6.1.6 (WHITNEY 1932) *Ein 3-fach zusammenhängender, planarer Graph kann nur auf eine Weise in die Ebene eingebettet werden.*

Beweis. Sei $G = (V, E)$ 3-fach zusammenhängend. Dann ist der Rand eines jeden Gebietes in einer Einbettung von G ein Kreis. Angenommen wir hätten zwei nicht äquivalente Einbettungen $G_1 = (E_1, V_1, R_1)$ und $G_2 = (E_2, V_2, R_2)$ von G. Dann muß es einen Kreis C in G geben, der Rand eines Gebietes von G_1 ist, der aber nicht Rand eines Gebietes von G_2 ist. Ohne Einschränkung können wir annehmen, daß C der Rand des äußeren Gebietes von G_1 ist.
Als eine *C-Brücke* von G bezeichnen wir zum einen jede Sehne von C (d.h. jede Kante von G, deren Endpunkte auf C liegen, die aber selbst nicht zu C gehört), und zum anderen jede Komponente H von $G - C$ zuzüglich der von H nach C führenden Kanten und deren Endpunkte. Die nachfolgende Abbildung zeigt die beiden Möglichkeiten für eine C-Brücke.



Wir sagen, daß zwei C-Brücken B und B' trennend sind, wenn es entweder zwei Knotenpaare $x, y \in B \cap C$ und $x', y' \in B' \cap C$ gibt, die sich auf C trennen, oder aber wenn B und B' genau drei Knoten gemeinsam haben. Die folgende Abbildung zeigt diese beiden Möglichkeiten von zwei sich trennenden C-Brücken:



Offensichtlich liegen zwei sich trennende C-Brücken in jeder Einbettung auf verschiedenen Seiten des Kreises C. Da C Rand des äußeren Gebietes in G_1 ist, gibt es also keine zwei sich trennenden C-Brücken in G. Da C nun nicht Rand eines Gebietes in der Einbettung G_2 ist, muß es in G_2 mindestens zwei C-Brücken B und B' geben, von denen eine außerhalb

von C und eine innerhalb von C liegt. Mithin liegen die Knoten in $B \cap C$ auf einem Teilweg von C, dessen Inneres keinen Knoten aus B' enthält. Entfernen wir nun aber die beiden Endpunkte dieses Teilweges aus dem Graphen G, so haben wir diesen damit in mindestens zwei Komponenten zerlegt - im Widerspruch zu der Voraussetzung, daß G 3-fach zusammenhängend ist. Es kann also keine Einbettung von G existieren, in der C nicht der Rand eines Gebietes ist. □

Übung 6.1.7 *Ein zweifach zusammenhängender, planarer Graph ist eindeutig in die Ebene einbettbar genau dann, wenn er sich nicht als kantendisjunkte Vereinigung zweier Graphen darstellen läßt, die beide keine Wege sind, beide mindestens drei Knoten enthalten und die genau zwei Knoten gemeinsam haben.*

Übung 6.1.8 a) *Ein Eulerscher, zusammenhängender planarer Graph besitzt einen Eulerzug, bei dem sich Wegabschnitte in Knoten höchstens „berühren", nicht aber überkreuzen.*
b) *Ein zusammenhängender planarer Graph vom Maximalgrad ≤ 4 besitzt einen Eulerzug, bei dem aufeinanderfolgende Kanten auf dem Rand eines Gebietes liegen.*

6.1.2 Die Eulersche Formel

Die Eulersche Formel stellt einen einfachen Zusammenhang zwischen der Anzahl der Knoten, Kanten und Gebiete eines zusammenhängenden, planaren Graphen her.

Satz 6.1.9 (EULER 1750) *Sei $G = (V, E, R)$ ein zusammenhängender, ebener Graph. Dann gilt:*

$$|V| - |E| + |R| \;=\; 2.$$

Beweis. Wir induzieren nach $|E|$. Die Fälle $|E| = 0$ und $|E| = 1$ sind offensichtlich trivial. Sei nun die Behauptung wahr für alle Graphen mit weniger als $|E|$ Kanten, und sei G ein Graph mit $|E|$ Kanten. Wir betrachten zunächst den Fall, daß G ein Baum ist. Dann besitzt G einen Knoten v vom Grad 1. Der zusammenhängende Graph $G - v$ besitzt $|V| - 1$ Knoten, $|E| - 1$ Kanten und $|R|$ Gebiete, so daß nach der Induktionsannahme $(|V| - 1) - (|E| - 1) + |R| = 2$ gilt. Daraus ergibt sich aber sofort die Behauptung.
Wir betrachten nun den Fall, daß G kein Baum ist. Enfernt man eine Kante aus einem Kreis von G, so verschmelzen genau zwei verschiedene Gebiete zu einem (hier geht via Jordanscher Kurvensatz ein, daß G in die Ebene eingebettet ist, denn auf dem Torus beispielsweise ist ein solcher Schluß i.a. falsch) und der verbleibende Graph hat $|V|$ Knoten, $|E| - 1$ Kanten und $|R| - 1$ Gebiete, so daß nach Induktionsvoraussetzung $|V| - (|E| - 1) + (|R| - 1) = 2$ gilt. Hieraus folgt wiederum sofort die Behauptung. □

Jede Einbettung eines planaren Graphen in die Ebene hat also gleich viele Gebiete. Aus der Eulerschen Formel leitet sich eine Reihe von Korollaren ab. Wir wollen zunächst zeigen, daß ein planarer Graph nur linear viele Kanten besitzen kann.

Korollar 6.1.10 *Für einen planaren Graphen $G = (V, E)$ auf mindestens drei Knoten gilt*

$$|E| \;\leq\; 3 \cdot |V| - 6.$$

Beweis. Sei (V, E, R) eine Einbettung von G in die Ebene. Wir zählen die Kanten-Gebiete-Inzidenzen auf zwei Weisen. Da jede Kante mit höchstens zwei Gebieten inzidiert und andererseits jedes Gebiet aus R von mindestens drei Kanten berandet wird, gilt: $3 \cdot |R| \leq$

$2 \cdot |E|$. Aus der Eulerschen Formel folgt sodann:

$$|V| - |E| + \frac{2}{3} \cdot |E| \geq 2 \;\Leftrightarrow\; 3 \cdot |V| - 6 \geq |E|. \qquad \square$$

Definition 6.1.11 *Ein planarer Graph $G = (V, E)$ heißt* maximal planar, *falls für alle nicht benachbarten $u, v \in V$ der Graph $G + \{u, v\}$ nicht mehr planar ist. Eine* Triangulation *der Ebene (auch Dreiecksgraph) ist ein planarer Graph, in dessen Einbettungen jedes Gebiet (auch das äußere) von einem (nicht notwendig konvexen) Dreieck berandet wird.*[2]

Proposition 6.1.12 *Für einen planaren Graphen $G = (V, E)$ sind äquivalent:*

(i) G ist maximal planar,

(ii) G ist eine Triangulation,

(iii) $|E| = 3|V| - 6$.

Beweis. In einem maximal planaren Graphen ist der Rand eines jeden Gebietes offensichtlich ein Dreieck. Für eine Triangulation gilt in den Ungleichungen des Beweises zu Korollar 6.1.10 Gleichheit. $(iii) \Rightarrow (i)$ folgt direkt aus Korollar 6.1.10. $\square$

Korollar 6.1.13 *Der K_5 ist nicht planar.*

Beweis. Es gilt $|E(K_5)| = \binom{5}{2} = 10 > 9 = 3 \cdot 5 - 6 = 3 \cdot |V(K_5)| - 6$, womit der K_5 nach Korollar 6.1.10 nicht planar ist. $\square$

Korollar 6.1.14 *Es sei G ein planarer Graph mit mindestens fünf Knoten. Dann gibt es mindestens vier Knoten in G vom Grad höchstens 5.*

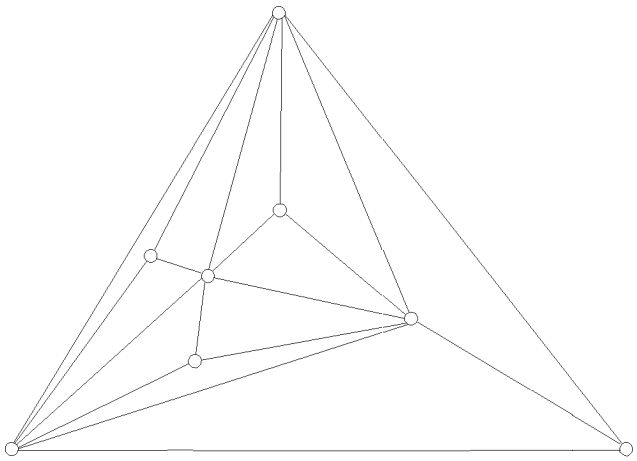
Beweis. Da jeder planare Graph Subgraph eines maximal planaren Graphen ist, genügt es, den Beweis für maximal planare Graphen zu führen.
Wir nehmen nun an, daß wir einen maximal planaren Graphen mit mindestens fünf Knoten hätten, in dem höchstens drei Knoten einen Grad kleiner als 6 haben. Da in einem maximal planaren Graphen jeder Knoten mindestens Grad drei hat, gilt:

$$|E| = \frac{1}{2} \cdot \sum_{v \in V(G)} deg(v) \geq \frac{1}{2}\{(|V| - 3) \cdot 6 + 3 \cdot 3\} = 3 \cdot |V| - \frac{9}{2} > 3 \cdot |V| - 6$$

im Widerspruch zu Korollar 6.1.10. $\square$

Der folgende Graph zeigt, daß das vorangegangene Korollar bestmöglich ist, da er nur vier Knoten enthält, die einen Grad kleiner als 6 haben.

[2] nicht zu verwechseln mit den chordalen Graphen, die auch als trianguliert bezeichnet werden, vgl. Abschnitt 7.2!



Übung 6.1.15 *Ein planarer Graph G ist 6-färbbar: $\chi(G) \leq 6$.*

Korollar 6.1.14 impliziert insbesondere, daß ein planarer Graph G wegen $\delta(G) \leq 5$ höchstens 5-fach zusammenhängend sein kann. Das Ikosaeder (s. Seite 181) liefert ein Beispiel eines 5-fach zusammenhängenden, planaren Graphen.

Übung 6.1.16 *Man finde einen 6-fach zh., (abzählbar) unendlichen, planaren Graphen.*

Für den Fall eines dreiecksfreien Graphen können wir die Aussage des Korollars 6.1.10 noch verschärfen.

Korollar 6.1.17 *Es sei G ein dreiecksfreier, zusammenhängender, planarer Graph auf mindestens 3 Knoten. Dann gilt:*

$$|E| \ \leq \ 2 \cdot |V| - 4.$$

Beweis. In einem dreiecksfreien Graphen, der kein Wald ist, ist der Rand eines jeden Gebietes mindestens ein Viereck. Somit gilt: $4 \cdot |R| \leq 2 \cdot |E|$. Aus der Eulerschen Formel folgt wieder:

$$|V| - |E| + \frac{1}{2}|E| \geq 2 \ \Leftrightarrow \ |E| \leq 2 \cdot |V| - 4. \qquad \square$$

Korollar 6.1.18 *Der $K_{3,3}$ ist nicht planar.*

Beweis. Ein bipartiter Graph ist insbesondere dreiecksfrei. Wegen $|E(K_{3,3})| = 9 > 8 = 2 \cdot 6 - 4 = 2 \cdot |V(K_{3,3})| - 4$ ist der $K_{3,3}$ also nicht planar. $\square$

Übungen

Übung 6.1.19 a) *Es gibt eine Konstante $c \in \mathbb{R}^+$, so daß jeder planare Graph mindestens $c \cdot n$ Knoten vom Grad höchstens 6 besitzt.*
b) *Ein dreiecksfreier planarer Graph besitzt einen Knoten vom Grad höchstens 3.*
c) *Ob ein (planarer) Graph dreiecksfrei ist, läßt sich in Zeit $\mathcal{O}(nm)$ ($\mathcal{O}(n)$) testen.*

Übung 6.1.20 *Für einen planaren Graphen $G = (V, E)$ der Taillenweite $g := g(G)$ gilt:*

$$|E| \ \leq \ \frac{g}{g-2}(|V| - 2).$$

Wie die Graphen $K_{\delta,n}$ zeigen, kann eine kardinalitätsmaximale stabile Menge in einem zusammenhängenden, planaren Graphen vom Minimalgrad $\delta \in \{1,2\}$ fast ganz V ausschöpfen.

Übung 6.1.21 *Für einen zusammenhängenden, planaren Graphen G mit Minimalgrad $\delta(G) \geq 3$ (beachte $\delta(G) \leq 5$) gilt:*

$$\alpha(G) \leq \frac{2n-4}{\delta(G)}.$$

Übung 6.1.22 [Jur94] *Sei G ein zusammenhängender, maximal planarer Graph der Ordnung $n > 4$.* **a)** *G enthält keine adjazenten Knoten vom Grad jeweils 3.* **b)** *G besitzt eine maximum stabile Menge S (d.h. $|S| = \alpha(G)$), die alle Knoten vom Grad 3 enthält.*

6.1.3 Die Sätze von Kuratowski und Wagner

Definition 6.1.23 *Ein Graph G gehe aus einem Graphen H durch sukzessives Einfügen neuer Knoten (vom Grad 2) auf Kanten hervor (d.h. die Kanten von H werden durch Pfade der Länge ≥ 1 ersetzt); Dann heißt G* Unterteilung *von H. Zwei Graphen heißen* homöomorph, *falls sie beide Unterteilungen eines dritten Graphen sind.*

Wir sagen, ein Graph G enthält einen Graphen H als Unterteilung (oder topologischen Subgraphen), falls er einen (schwachen) Subgraphen H' enthält, der eine Unterteilung von H ist. Offenbar ist ein Graph planar genau dann, wenn jede seiner Unterteilungen planar ist. Da jeder Subgraph eines planaren Graphen planar ist, besitzt ein planarer Graph nach den Ergebnissen des letzten Abschnitts sicher keine Unterteilung des K_5 oder des $K_{3,3}$.

Daraus folgt beispielsweise, daß wir das Problem, in einem planaren Graphen G die Cliquenzahl $\omega(G)$ zu bestimmen und eine $\omega(G)$-Clique anzugeben, trivial durch Enumeration aller 4-Teilmengen der Knotenmenge in Zeit $\mathcal{O}(|V|^4)$ lösen können.[3]

Überraschenderweise gilt nun auch die Umkehrung der obigen Beobachtung, d.h. wenn ein Graph weder eine Unterteilung des $K_{3,3}$ noch eine Unterteilung des K_5 enthält, so ist er auch schon planar.

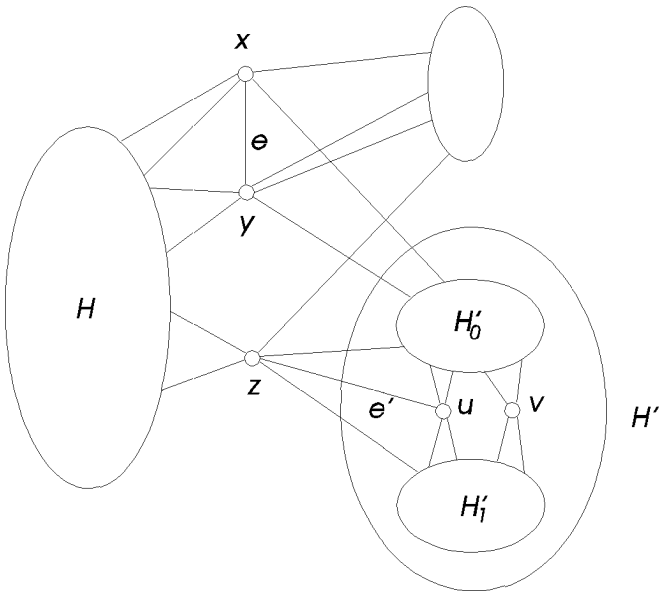
Satz 6.1.24 (Kuratowski 1930) *Ein Graph ist genau dann planar, wenn er keine Unterteilung des $K_{3,3}$ oder des K_5 enthält.*

Der elegante Beweis von Thomassen [Tho80, Tho81] benutzt folgende zwei Hilfsaussagen.

Lemma 6.1.25 *Jeder 3-fach zusammenhängende Graph G auf mindestens fünf Knoten enthält eine Kante e, so daß G/e 3-fach zusammenhängend ist.*

Beweis. Angenommen für jede Kante $e = (x,y)$ von G ist G/e nicht 3-fach zusammenhängend. Da G 3-fach zusammenhängend ist, ist G/e mindestens 2-fach zusammenhängend. Nach unserer Annahme existiert daher ein trennendes Knotenpaar in G/e, wobei einer der beiden Knoten x bzw. y sein muß. Somit wissen wir, daß G eine trennende Knotenmenge der Form $\{x,y,z\}$ besitzt.
Es seien nun $e = \{x,y\}$ und z so gewählt, daß die größte Komponente H in $G\backslash\{x,y,z\}$ größtmöglich ist. Siehe hierzu die folgende Abbildung:

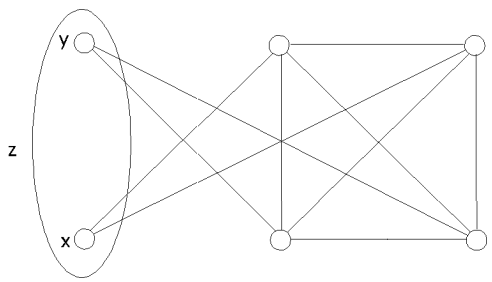
[3] Durch Enumeration aller disjunkter Kantenpaare erhält man einen $\mathcal{O}(|V|^2)$-Algorithmus. In [PY81] findet sich sogar ein linearer Algorithmus. Independent Set bleibt dagegen selbst für planare Graphen vom Maximalgrad 3 NP-vollständig, vgl. Korollar 8.3.2.



Nun wähle man eine beliebige andere Komponente H' in $G\backslash\{x,y,z\}$. Es sei $e' = \{z,u\}$ eine Kante, die z und einen Knoten $u \in H'$ verbindet. Eine solche existiert, da ansonsten $\{x,y\}$ bereits trennend wäre. Nun ist nach unserer Annahme G/e' nicht 3-fach zusammenhängend, so daß G eine trennende Knotenmenge der Gestalt $\{z,u,v\}$ besitzt. Der Knoten v muß dann in $H' \cup \{x,y\}$ liegen, da sonst bereits $\{z,v\}$ eine trennende Knotenmenge in G bildet.
Ist nun v in H', so besitzt $G\backslash\{z,u,v\}$ eine Komponente, die H und $\{x,y\}$ enthält, was im Widerspruch zur Maximalität von H steht.
Ist $v = x$ (oder $v = y$), so ist $H \cup \{y\}$ in einer Komponente von $G - \{z,u,v\}$ enthalten, was wiederum einen Widerspruch zur Maximalität von H darstellt. □

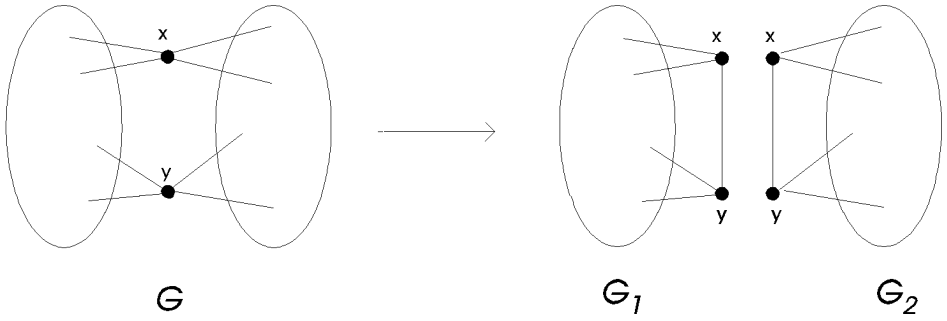
Lemma 6.1.26 *Es sei e eine Kante in G. Wenn G/e eine Unterteilung des K_5 oder des $K_{3,3}$ enthält, so enthält auch G eine solche Unterteilung.*

Beweis. Es sei $e = (x,y)$ und z der durch Kontraktion der Kante e neu hervorgegangene Knoten in G/e. Bezeichne S die Unterteilung des K_5 bzw. $K_{3,3}$ in G/e. Jede Kante (z,v) in G/e hat mindestens eine der beiden Kanten (x,v) und (y,v) als Gegenstück in G. Wir wählen für jede Kante (z,v) beliebig genau eine der höchstens zwei Kanten (x,v) und (y,v) in G und erhalten zusammen mit der Kante e einen Teilgraphen T in G. Dann enthält schon T/e die Unterteilung des K_5 bzw. $K_{3,3}$.
Falls z den Grad zwei in S hat, so enthält offenbar auch T diese Unterteilung. Ebenso, falls einer der Knoten u oder v den Grad 2 in T hat. Haben aber x und y beide mindestens den Grad 3 in T (d.h. S ist eine Unterteilung des K_5), so enthält T eine Unterteilung des $K_{3,3}$:

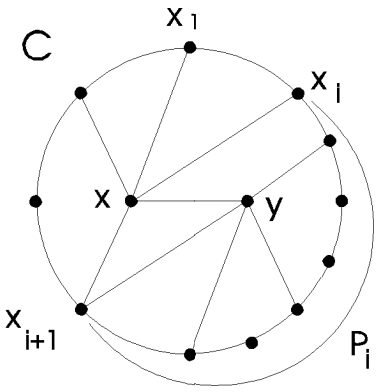


□

Beweis von Satz 6.1.24:
durch Induktion über $|V|$. Da der $K_5 - e$ planar ist, ist die Behauptung für $|V| \leq 5$ offensichtlich richtig. G habe also mindestens 6 Knoten und enthalte weder eine Unterteilung des K_5 noch des $K_{3,3}$.
Fall 1: G ist nicht 3-fach zusammenhängend.
Da ein Graph planar ist genau dann, wenn jeder seiner Blöcke planar ist, sei G ohne Einschränkung 2-fach zusammenhängend und (x, y) ein trennendes Knotenpaar:

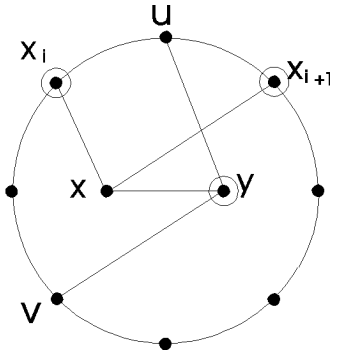


Ist H eine Komponente von $G \backslash \{x, y\}$, so bezeichne $G_1 := G[H \cup \{x, y\}]$ und $G_2 := G[V \backslash H]$, wie in der Abbildung. Sofern nicht schon in G enthalten, sei die Kante (x, y) in den beiden Teilen G_1 und G_2 hinzugefügt. Da x und y jeweils beide mit allen Komponenten von $G \setminus \{x, y\}$ verbunden sind, kann dadurch keine Unterteilung des K_5 oder $K_{3,3}$ entstehen. Da G_1 und G_2 auch jeweils weniger Knoten als G enthalten, sind sie nach Induktionsvoraussetzung also planar. Wählen wir nun eine Einbettung von G_1 und G_2, so daß die Kante (x, y) jeweils das äußere Gebiet berandet, so können wir diese beiden Einbettungen zu einer Einbettung von G zusammenfügen, d.h. G ist planar.
Fall 2: G ist 3-fach zusammenhängend.
Nach Lemma 6.1.25 besitzt G eine Kante $e = (x, y)$, so daß G/e 3-fach zusammenhängend ist. Es bezeichne z den durch Kontraktion der Kante e neu hervorgegangenen Knoten in G/e. Nach Lemma 6.1.26 besitzt G/e keine Unterteilung des $K_{3,3}$ oder K_5 und ist daher nach Induktionsannahme planar.
Betrachte eine Einbettung von G/e in die Ebene. Ohne Einschränkung ist das Gebiet in $G/e - z$, das den Punkt z im Inneren enthält, nicht das äußere. Der Rand C dieses Gebietes ist nach Proposition 6.1.3 ein Kreis. Es seien $x_1, x_2, \ldots, x_k \in C$ die von y verschiedenen Nachbarn von x in G in zyklischer Reihenfolge und P_i der Teilpfad $x_i, \ldots, x_{i+1}$ auf C, wobei $x_{k+1} := x_1$ sei. Falls alle von x verschiedenen Nachbarn von y in einem der P_i enthalten sind, so können wir die Einbettung von G/e offenbar auf G fortsetzen:

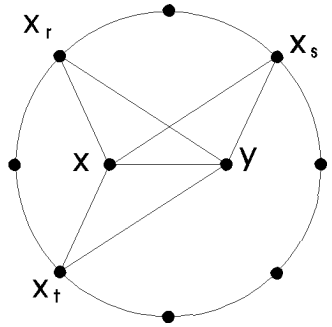


Die KURATOWSKI-Bedingung schließt nun aber gerade alle anderen Situationen aus. Wir unterscheiden zwei Fälle:

α) y besitzt einen Nachbarn $u \in P_i$ auf $C \setminus \{x_1, \ldots, x_k\}$. Aufgrund des 3-Zusammenhangs von G ist $d(y) \geq 3$, und y besitzt nach Annahme noch mindestens einen weiteren Nachbarn $v \notin P_i$ auf C. Somit enthält G eine Unterteilung des $K_{3,3}$:



β) Alle Nachbarn von y liegen in $\{x_1, \ldots, x_k\}$. Im Fall $d(y) = 3$ folgt $k \geq 4$, und man schließt wie in α). Hat y andernfalls drei Nachbarn x_r, x_s, x_t in $\{x_1, ..., x_k\}$, so findet man eine Unterteilung des K_5 in $G[C \cup \{x, y\}]$:



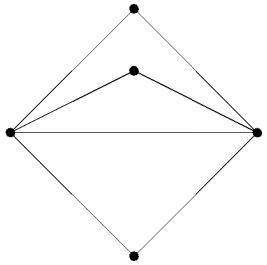
□

Aus diesem Beweis leiten wir nun ohne Mühe ab, daß 3-zusammenhängende, planare Graphen sogar eine sogenannte *konvexe* Einbettung in die Ebene besitzen, in der alle Gebiete bis auf das äußere konvex sind. Insbesondere sind in einer konvexen Einbettung Kanten, die nicht das Außengebiet beranden, schon notwendig geradlinig - Kanten auf dem Rand des Außengebiets sind es o.B.d.A.

Korollar 6.1.27 (STEIN 1951; TUTTE 1960) *Ein 3-fach zusammenhängender, planarer Graph besitzt eine konvexe Einbettung in die Ebene.*

Beweis durch Induktion über $|V|$. Für $|V| \leq 5$ überzeugt man sich leicht von der Richtigkeit der Aussage. Sei also $|V| > 5$. Wir haben beim Beweis des Satzes von KURATOWSKI (Fall 2) gesehen, daß der Knoten y nur mit x und Knoten auf C, die in einem P_i liegen, verbunden ist. Wir können daher y einfach in das Innere des von den Kanten $(x, x_i), (x, x_{i+1})$ und P_i berandeten Gebietes (nahe genug bei x) legen und geradlinig mit allen seinen Nachbarn verbinden, um eine geradlinige, konvexe Einbettung von G zu erhalten, siehe obige Abbildung. Man verifiziere lediglich, daß das von C berandete Gebiet tatsächlich ohne Einschränkung als nicht das äußere angenommen werden kann bzw. überlege sich, daß man auch in diesem Fall so vorgehen kann. □

Einen effizienten Algorithmus, um eine solche Einbettung zu konstruieren, findet man bei [NC88]. Ein nur zweifach zusammenhängender Graph besitzt nicht notwendigerweise eine konvexe Einbettung in die Ebene, wie das folgende Beispiel demonstriert:



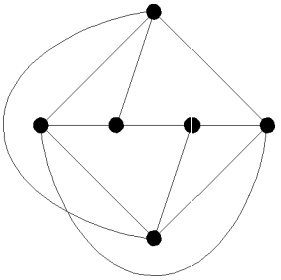
Der K_5 und der $K_{3,3}$ spielen im Satz von KURATOWSKI unterschiedliche Rollen:

Übung 6.1.28 (WAGNER 1937, HALL 1943) *Ein 3-zusammenhängender Graph $\neq K_5$ ist planar genau dann, wenn er keine Unterteilung des $K_{3,3}$ enthält.*[4]

Definition 6.1.29 *Ein Graph G enthält einen Graphen H als* Minor, *in Zeichen $G \succ H$, falls G einen Subgraphen H' enthält, der sich zu H zusammenziehen läßt, d.h. es ist möglich, H' durch sukzessives Kontrahieren von Kanten in H überzuführen.*

Die Relation „$\succ$“ zwischen Graphen ist offenbar transitiv ($G_1 \succ G_2 \wedge G_2 \succ G_3 \Rightarrow G_1 \succ G_3$) und antisymmetrisch ($G_1 \succ G_2 \wedge G_2 \succ G_1 \Rightarrow G_1 \cong G_2$).

Enthält ein Graph G eine Unterteilung eines Graphen H, so enthält G den Graphen H auch als Minor. Die Umkehrung gilt i.a. jedoch nicht! So enthält beispielsweise der folgende Graph den K_5 als Minor, nicht aber als Unterteilung:



Es gilt jedoch die folgende Aussage, die uns noch öfter von Nutzen sein wird:

[4] Nach KELMANS [Kel84] und THOMASSEN [Tho84b] genügt es sogar, einen Kreis mit drei sich paarweise kreuzenden Sehnen auszuschließen.

Übung 6.1.30 a) *Seien G und H zusammenhängende Graphen. Dann enthält G den Graphen H als Minor genau dann, wenn es eine Abbildung $\phi : V(G) \to V(H)$ gibt, so daß für die Urbilder $\phi^{-1}(v)$ von Knoten $v \in H$ gilt:*

1. *$G[\phi^{-1}(v)]$ ist zusammenhängend für alle $v \in V(H)$;*
2. *Für alle $\{u, v\} \in E(H)$ ist der Schnitt $\langle \phi^{-1}(u), \phi^{-1}(v) \rangle_G \neq \emptyset$.*

b) *Gilt $\Delta(H) \leq 3$, so enthält ein Graph den Graphen H als Minor genau dann, wenn er ihn auch als Unterteilung enthält.*

In den Anwendungen ist oft das folgende Pendant zum Satz von KURATOWSKI nützlich.

Korollar 6.1.31 (WAGNER 1937) *Ein Graph ist planar genau dann, wenn er weder den K_5 noch den $K_{3,3}$ als Minor enthält.*

Beweis. Man sieht leicht ein, daß jeder Minor eines planaren Graphen planar ist. Besitzt ein Graph umgekehrt weder den K_5 noch den $K_{3,3}$ als Minor, so sicher auch nicht als Unterteilung, so daß das Korollar aus dem Satz von KURATOWSKI folgt. □

Übungen

Übung 6.1.32 a) (WHITNEY 1931) *Eine Triangulation der Ordnung $n \geq 4$ ist 3-zusammenhängend.*
b) *Man beweise Satz 6.1.5.*

Übung 6.1.33 *Man überlege sich auf drei Arten, daß der* PETERSEN*-Graph nicht planar ist.*

Übung 6.1.34 (ORE 1951) *Ein von einem Knoten $v \in V$ aus willkürlich durchlaufbarer Graph* (vgl. Übung 3.1.8) *ist planar.*

Ein planarer Graph heißt *kreisartig planar* (engl. outerplanar), falls er so in die Ebene eingebettet werden kann, daß alle Knoten auf dem Rand genau eines Gebietes liegen – o.B.d.A. des äußeren. Wälder sind also beispielsweise kreisartig planar. Offenbar ist ein Graph G kreisartig planar genau dann, wenn $G * 1$ (G mit einem zusätzlichen Knoten, der zu allen Knoten von G verbunden wird) planar ist.

Übung 6.1.35 a) *Ein (kanten-) maximaler kreisartig planarer Graph ($n \geq 3$) ist 2-zusammenhängend.*
b) *Ein 2-zusammenhängender, kreisartig planarer Graph ist Hamiltonsch.*
c) (KÖNIG 1905) *Ein kreisartig planarer Graph ist 3-färbbar.*
d) *Ein kreisartig planarer Graph hat höchstens $2n - 3$ Kanten.*
e) (CHARTRAND, HARARY 1967) *Ein Graph ist kreisartig planar genau dann, wenn er keine Unterteilung des K_4 oder $K_{2,3}$ enthält.*
f) (FIORINI 1975) *Ein zusammenhängender, kreisartig planarer Graph ist Klasse 2 genau dann, wenn er ein ungerader Kreis ist.*
[*Hinweis zu „⇒“:* man nehme die Existenz eines (knoten- und kantenminimalen) zusammenhängenden, kreisartig planaren Graphen der Klasse 2 an, der kein ungerader Kreis ist.]

Übung 6.1.36 (CHVÁTAL's Art Gallery Theorem, 1975) *Sei K eine Teilmenge der Ebene, die durch einen (nicht notwendig konvexen) Polygonzug über n Punkte berandet ist. Dann gibt es eine Menge S von höchstens $\lfloor n/3 \rfloor$ vielen Punkten aus K, die zusammen ganz K überblicken, d.h. für jeden Punkt $x \in K$ gibt es ein $s \in S$, so daß die Strecke xs vollständig in K enthalten ist. Für jedes $n \in \mathbb{N}$ gibt es einen Polygonzug, so daß jede solche Menge S $\lfloor n/3 \rfloor$ viele Punkte aus K enthält.*

Weitere Art-Gallery-Theorems finden sich in [Rou87].

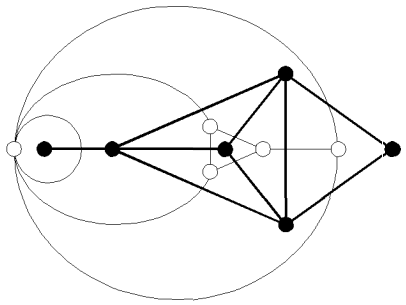
Übung 6.1.37 *Ist ein Graph G nicht planar, so ist auch sein Linegraph $L(G)$ nicht planar.*

6.1.4 Dualität bei planaren Graphen

Mit Hilfe des Begriffs des kombinatorischen Duals werden wir eine weitere Charakterisierung (endlicher) planarer Graphen geben können.

Definition 6.1.38 *Es sei $G = (V, E, R)$ ein ebener Graph. Das* geometrische Dual $G^* = (V^* = R, E^*, R^*)$ *von G erhält man auf folgende Weise: Man wähle im Inneren eines jeden Gebietes von G einen Punkt. Für jede Kante $e \in E$ verbinde man die beiden ausgewählten Punkte in den an e angrenzenden Gebieten durch eine Jordankurve $e^* \in E^*$, die e im Inneren schneidet und sonst ganz innerhalb der beiden Gebiete verläuft - ist e eine Brücke, so zeichne man eine Schlinge.*

Das folgende Beispiel möge den Begriff des geometrischen Duals veranschaulichen:



Die Kanten von G und G^* stehen also in bijektiver Korrespondenz. Offenbar ist das geometrische Dual eines ebenen Graphen wiederum eben. Man überlegt sich unschwer: e ist eine Brücke in G genau dann, wenn e^* Schlinge in G^* ist, und Mehrfachkanten treten in G^* genau dort auf, wo Gebiete von G mehrere gemeinsame Grenzkanten besitzen (eine sog. *mehrfache Nachbarschaft*).

G^* hängt natürlich wesentlich von der Einbettung von G ab. Äquivalente Einbettungen führen jedoch zu isomorphen Dualen. Der Dualgraph charakterisiert umgekehrt aber nicht die Einbettung, denn es gibt planare Graphen mit nicht äquivalenten Einbettungen in die Ebene, deren Duale isomorph sind. Das Dual eines ebenen Graphen ist stets zusammenhängend - für zusammenhängende Graphen ist Dualisierung involutorisch:

Übung 6.1.39 *Es sei G eben und zusammenhängend. Dann ist $(G^*)^*$ isomorph zu G.*

Ein Graph G heißt *selbstdual*, falls er isomorph zu G^* ist. Das Tetraeder K_4 ist beispielsweise die einzige selbstduale Triangulation (Beweis?).

Übung 6.1.40 *Man gebe eine unendliche Familie selbstdualer Graphen an.*

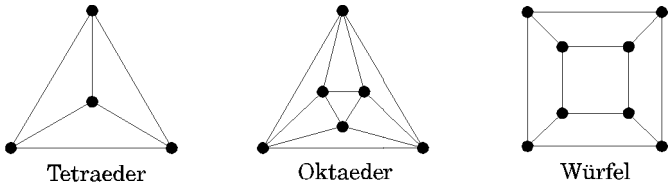
Zusammenhang und polyedrische Graphen. Wir wollen hier kurz die Beziehung zwischen Zusammenhang und Dualität in planaren Graphen untersuchen. Ein 2-Kantenzusammenhängender Graph hat einen schlingenfreien Multigraphen zum Dual. Das Dual eines Graphen ist zwar stets zusammenhängend, das Dual eines Kreises beispielsweise aber nicht 2-zusammenhängend. Man überlege sich zunächst anschaulich:

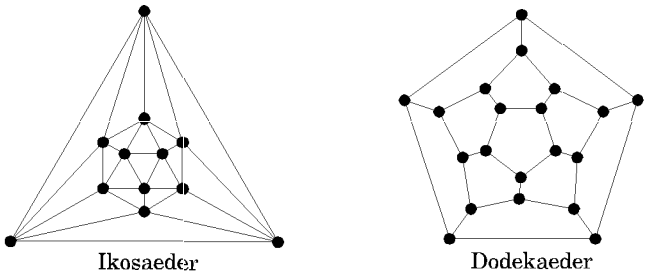
Proposition 6.1.41 *Ein zweifach zusammenhängender, ebener Graph $G = (V, E, R)$ mit mindestens drei Gebieten hat ein zweifach zusammenhängendes Dual G^*.*

Beweis. Hat G genau drei Gebiete, so haben nach Proposition 6.1.3 je zwei eine gemeinsame Kante, d.h. $G^* = K_3$. Andernfalls genügt es zu zeigen, daß je zwei nicht-parallele

Kanten $e^*, f^* \in E(G^*)$ auf einem Kreis liegen. Im Fall $e^* \cap f^* = \{x\}$, $x \in R$ läuft man einfach die Nachbarländer von x im Uhrzeigersinn ab. Sonst liegen die dualen Kanten $e, f \in E$ in einem Kreis C von G, der die Ebene in Inneres und Äußeres teilt. Im Inneren wie im Äußeren von C findet man je eine Gebietskette von G, die e^* und f^* verbinden. □

Hat der Dualgraph G^* aber beispielsweise Gebiete, die nur von einer Schlinge berandet sind, so hat $(G^*)^* \cong G$ Blätter, auch wenn G^* zweifach zusammenhängt. Die Umkehrung gilt also nicht.
Das geometrische Dual eines ebenen Graphen G habe eine Mehrfachkante zwischen zwei Gebieten R_1 und R_2 von G. Je zwei Kanten dieser Mehrfachkante bilden einen Kreis, der in G einen Subgraph (eventuell nur einen einzigen Knoten) einschließt, der nur über die beiden zugehörigen gemeinsamen Grenzkanten von R_1 und R_2 mit dem Rest des Graphen verbunden ist. Daher ist G dann also nicht 3-zusammenhängend. Ist G hingegen 3-zusammenhängend, so ist G^* ein einfacher Graph. Beispiele: die *Platonischen Körper*





Das Tetraeder ist selbstdual, der Würfel ist dual zum Oktaeder und das Dodekaeder zum Ikosaeder. Daß das Dual wieder ein Graph ist, gilt allgemein für sogenannte polyedrische Graphen. Man erinnere sich, daß ein n-dimensionales Polyeder der Schnitt endlich vieler Halbräume des $\mathbb{R}^n$ ist.

Definition 6.1.42 *Ein Graph heißt* polyedrisch, *falls er isomorph zum Gerüstgraphen eines dreidimensionalen, endlichen Polyeders (Polytops) ist.*

Offenbar ist ein polyedrischer Graph planar (man projiziere das Polyeder auf eine maximale, im Polyeder befindliche Kugel) und der minimale Knotengrad $\delta(G) \geq 3$. Die Knoten des polyedrischen Graphen entsprechen den Ecken des Polyeders, die Gebiete den Seitenflächen. Da diese durch Kreise berandet sind, ist ein solcher Graph nach Proposition 6.1.3 2-zusammenhängend. Aus geometrischen Überlegungen folgt, daß G sogar 3-zusammenhängend ist (siehe z.B. [Bar95]), und tatsächlich charakterisiert dies polyedrische Graphen bereits (siehe z.B. [SR34, Grü67, Zie95]):

Satz 6.1.43 (Steinitz 1922)
Ein Graph ist genau dann polyedrisch, wenn er planar und 3-zusammenhängend ist. □

Mittels der Konvexität von Polyedern folgert man aus diesem Satz leicht auch wieder das Korollar 6.1.27. Wir erinnern uns ferner, daß wir von polyedrischen Graphen zeigen konnten, daß sie nur auf eine Weise in die Sphäre einbettbar sind, vgl. Satz 6.1.6. Mehr zu polyedrischen Graphen in [Mal88]. Für diese Klasse von Graphen können wir sogar zeigen, daß sie abgeschlossen bezüglich Dualisierung ist, d.h.

Proposition 6.1.44
Ist ein ebener Graph G 3-zusammenhängend, so auch sein geometrisches Dual G^*.

Beweis. Seien S und T zwei Gebiete in $G = (V, E, R)$. Es ist zu zeigen, daß es in G^* drei knotendisjunkte S-T-Wege gibt. Zu je drei Knoten auf dem Rand des Gebietes S bzw. T gibt es nach dem Fächersatz von jedem Knoten $v \in V$ aus drei knotendisjunkte Wege W_i, $i = 1, 2, 3$. Daher ist auch der Graph G' 3-zusammenhängend, der aus G entsteht, indem man in jedes der Gebiete S bzw. T noch einen Knoten s bzw. t setzt und ihn mit den drei Knoten auf dem Gebietsrand durch eine Kante verbindet. Drei knotendisjunkte s-t-Wege in G' zerlegen aber die Ebene in drei Gebietsketten von S nach T. Aufgrund des 3-Zusammenhangs von G kann nur das Innere *einer* solchen Gebietskette leer sein, d.h. der zugehörige Weg in G^* eine einzelne Kante. □

Das Oktaeder zeigt, daß eine solche Beziehung bei 4-Zusammenhang nicht mehr gilt!

Schließlich wollen wir noch ein klassisches Resultat erwähnen. Dazu folgende

Definition 6.1.45 *Ein Polyeder heißt* regulär *oder* platonischer Körper*, falls jede seiner Seitenflächen von einem regulären k-Eck berandet wird für ein $k \in \mathbb{N}$ (d.h. alle Kanten gleichlang und alle Winkel gleich) und die in einer Ecke zusammenlaufenden Kanten kongruente Eckenfiguren bilden.*

Die Gerüstgraphen von regulären Polyedern sind also samt ihrer geometrischen Duale regulär (von konstantem Knotengrad). Schon die „alten Griechen“ wußten:

Satz 6.1.46 *Es gibt genau 5 reguläre Polyeder (platonische Körper).*

Beweisskizze. Sei $G = (V, E, F)$ der ebene Gerüstgraph eines regulären Polyeders. G sei regulär vom Grade k, und die Gebiete in F seien durch h-Ecke berandet. Dann gilt offenbar:

$$h|F| \;=\; 2|E| \;=\; k|V|.$$

Die EULERsche Formel

$$2 \;=\; |V| - |E| + |F| \;=\; \frac{2|E|}{k} - |E| + \frac{2|E|}{h}$$

liefert

$$\frac{1}{k} + \frac{1}{h} \;=\; \frac{2 + |E|}{2|E|} \;>\; \frac{1}{2}.$$

Wegen $k \geq 3$ und $h \geq 3$ folgt daraus

$$\begin{aligned} 3 \;\leq\; h \;&\leq\; 5 \\ 3 \;\leq\; k \;&\leq\; 5. \end{aligned}$$

Die platonischen Körper ergeben sich nun durch Unterscheidung der neun möglichen Fälle, ihre Eindeutigkeit folgt aus geometrischen Überlegungen. □

Korollar 6.1.47 *Die Gerüstgraphen platonischer Körper sind genau diejenigen polyedrischen Graphen, die zugleich mit ihrem geometrischen Dual regulär sind.*

Beweis. Siehe den Beweis des obigen Satzes. □

Der Satz von Whitney. Das folgende Lemma stellt einen Zusammenhang zwischen den Kreisen in G und den minimal trennenden Kantenmengen in G^* her.

Lemma 6.1.48 *Es sei G ein ebener Graph und G^* das geometrische Dual von G. Die Menge $C \subseteq E(G)$ ist ein Kreis genau dann, wenn die entsprechende Kantenmenge $C^* \subseteq E(G^*)$ eine (bezüglich Inklusion) minimal trennende Kantenmenge in G^* ist.*

Beweis. Es sei $C \subseteq E(G)$ ein Kreis in G. Dann ist C^* eine trennende Kantenmenge in G^*, da sowohl im Inneren als auch im Äußeren von C ein Knoten von G^* liegt. Da sowohl das Innere des Kreises C in G^*, als auch das Äußere zusammenhängend ist, ist C^* minimal.
Es sei nun C^* eine minimal trennende Kantenmenge in G^*. Dann besitzt $G^* - C^*$ genau zwei zusammenhängende Komponenten. Kontrahiert man jede dieser beiden Komponenten zu einem Punkt, so erhält man zwei Punkte, die durch die trennenden Kanten von C^* verbunden sind. Das Dual dieses Graphen ist ein Kreis in G und bleibt auch einer, wenn man die Kontraktion wieder rückgängig macht. □

Dieser Zusammenhang legt die folgende Verallgemeinerung des Begriffs des geometrischen Duals auf beliebige, nicht notwendig planare Graphen nahe.

Definition 6.1.49 *Es sei $G = (V, E)$ ein Graph. Dann heißt ein Graph $G^\star = (V^\star, E^\star)$* kombinatorisches Dual *zu G, falls es eine Bijektion $\varphi : E \to E^\star$ gibt, so daß $C \subseteq E$ genau dann ein Kreis in G ist, wenn $C^\star = \varphi(C) \subseteq E^\star$ eine minimal trennende Kantenmenge in $G^\star$ ist.*

Es stellt sich nun die Frage, welche Graphen ein kombinatorisches Dual besitzen. Aufgrund von Lemma 6.1.48 wissen wir bereits, daß zumindestens die planaren Graphen ein solches besitzen:

Korollar 6.1.50 *Es sei G ein planarer Graph. Dann ist jedes geometrische Dual von G auch ein kombinatorisches Dual von G.* □

WHITNEY zeigte nun, daß die Graphen, die ein kombinatorisches Dual besitzen, genau die planaren Graphen sind. Mit Hilfe der folgende vier Lemmata führen wir dieses Resultat auf den Satz von KURATOWSKI zurück.

Lemma 6.1.51 *Besitzt ein Graph G ein kombinatorisches Dual, so auch jeder Subgraph von G.*

Beweis. Es genügt offensichtlich zu zeigen, daß der Graph $G - \{e\}$ für jede beliebige Kante e in G ein kombinatorischen Dual besitzt, wenn nur G ein kombinatorisches Dual besitzt.
Es sei nun $G^\star$ ein kombinatorisches Dual von G. Wir zeigen, daß dann $G^\star/e^\star$ ein kombinatorisches Dual von $G - \{e\}$ ist.
Es sei $C \subseteq E - \{e\}$ ein Kreis in $G - \{e\}$. Da $C^\star$ in $G^\star$ trennend ist, ist $C^\star$ auch in $G^\star/e^\star$ trennend. Da der Zusammenhang der Komponenten in $G^\star - C^\star$ durch Kontraktion der Kante $e^\star$ nicht beeinflußt wird, ist $C^\star$ auch in $G^\star/e^\star$ minimal trennend. Ist nun umgekehrt $C^\star$ eine minimale trennende Kantenmenge in $G^\star/e^\star$, so ist sie dies auch in $G^\star$, und damit ist C ein Kreis in $G - \{e\}$. □

Lemma 6.1.52 *Es sei G ein Graph mit kombinatorischem Dual $G^\star$. Ferner seien e_o und e_1 Kanten in G. Dann sind $e_0^\star$ und $e_1^\star$ genau dann parallel, wenn jeder Kreis in G, der e_0 enthält, auch e_1 enthält und umgekehrt.*

Beweis. Angenommen jeder Kreis in G, der e_i enthält, enthält auch $e_{1-i}, i = 0, 1$. Dann enthält jede minimal trennende Menge in G*, die $e_i^\star$ enthält auch $e_{1-i}^\star$. Wären nun $e_0^\star$ und $e_1^\star$ nicht parallel, so gäbe es einen spannenden Baum T von G^*, der sowohl $e_0^\star$ als auch $e_1^\star$ enthält. Das Entfernen der Kante $e_0^\star$ zerlegt diesen Baum in zwei Komponenten T_0 und T_1. Die Menge der Kanten zwischen T_0 und T_1 in $G^\star$ ist minimal trennend und enthält $e_0^\star$, nicht jedoch $e_1^\star$. Dies ist ein Widerspruch. Die andere Richtung folgt durch Umkehrung des Beweises. □

Lemma 6.1.53 *Es sei H eine Unterteilung eines Graphen G. Besitzt H ein kombinatorisches Dual, so auch G.*

Beweis. Es reicht offensichtlich den Fall zu betrachten, daß H durch Unterteilung einer einzigen Kante in G entstanden ist.
Es seien e_0, e_1 die beiden Kanten in H, die durch Unterteilung der Kante e in G entstanden sind. Wir zeigen nun, daß $H^\star - \{e_1^\star\}$ ein kombinatorisches Dual von G ist.
Da jeder Kreis in H, der e_0 benutzt auch e_1 benutzt (und umgekehrt) wissen wir nach Lemma 6.1.52, daß $e_0^\star$ und $e_1^\star$ parallel sind. Das bedeutet, daß $H^\star - \{e_1^\star\}$ aus $H^\star$ durch Identifikation der Kanten $e_0^\star$ und $e_1^\star$ entsteht. Dann ist aber $H^\star - \{e_1^\star\}$ das kombinatorische Dual zu G. □

Lemma 6.1.54 *Weder der $K_{3,3}$ noch der K_5 besitzen ein kombinatorisches Dual.*

Beweis. Angenommen, es existiert ein kombinatorisches Dual des $K_{3,3}$. Da jeder Kreis im $K_{3,3}$ eine Länge von mindestens 4 hat, müßte jede minimal trennende Kantenmenge in $K_{3,3}^\star$ auch mindestens 4 Kanten enthalten. Nun kann man zu je zwei Kanten im $K_{3,3}$ einen Kreis finden, der die eine enthält und die andere nicht. Dies bedeutet, daß es im $K_{3,3}^\star$ keine parallelen Kanten geben kann. Wir wissen nun, daß der Grad eines jeden Knoten im $K_{3,3}^\star$ mindestens vier sein muß. Da der $K_{3,3}^\star$ 9 Kanten haben muß , muß er mindestens 5 Knoten enthalten. Dann enthält er aber mindestens $\frac{1}{2} \cdot 5 \cdot 4 = 10$ Kanten, was offensichtlich nicht möglich ist.

Angenommen, der K_5 besitzt ein kombinatorisches Dual. Dieses müßte dann 10 Kanten besitzen und dürfte keine parallelen Kanten besitzen, da es im K_5 zu je zwei Kanten einen Kreis gibt, der die eine enthält und die andere nicht. Da jeder Kreis im K_5 mindestens die Länge 3 hat, hat jede minimal trennende Kantenmenge im $K_5^\star$ mindestens 3 Elemente. Somit hat jeder Knoten im $K_5^\star$ mindestens den Grad 3. Hätte nun der $K_5^\star$ 5 Knoten, so müßte aufgrund der Kantenzahl der $K_5^\star$ isomorph zum K_5 sein. Dies ist jedoch nicht möglich, da der K_5 keine trennende Kantenmenge der Kardinalität 3 besitzt.
Hätte der $K_5^\star$ mehr als 6 Knoten, so besäße er mindestens $\frac{1}{2} \cdot 7 \cdot 3 = 10,5$ Kanten, was nicht möglich ist.
Es bleibt also nur noch der Fall zu untersuchen, daß der $K_5^\star$ 6 Knoten besitzt. Es gibt nun mindestens eine minimal trennende Kantenmenge der Kardinalität 3 in $K_5^\star$, die den $K_5^\star$ in zwei Teile S_0 und S_1 zerlegt. Hätten beide Teile nun 3 Knoten, so könnte der $K_5^\star$ nur höchstens $2 \cdot \binom{3}{2} + 3 = 9$ Kanten enthalten. Hätte ein Teil 4 Knoten und der andere 2, so gäbe es in dem Teil mit 2 Knoten einen Knoten vom Grad 2. □

Satz 6.1.55 (WHITNEY 1932b/1933)
Ein Graph ist genau dann planar, wenn er ein kombinatorisches Dual besitzt.

Beweis. Daß jeder planare Graph ein kombinatorisches Dual besitzt, wurde bereits mit Korollar 6.1.50 gezeigt. Besitzt nun ein Graph G ein kombinatorisches Dual, so besitzt auch jeder Teilgraph dieses Graphen nach Lemma 6.1.51 ein kombinatorisches Dual. Da der K_5 und der $K_{3,3}$ kein kombinatorisches Dual besitzen, besitzt nach Lemma 6.1.53 auch keine Unterteilung dieser Graphen ein kombinatorisches Dual. Somit besitzt G keine Unterteilung des $K_{3,3}$ oder K_5 und ist damit nach dem Satz von KURATOWSKI planar. □

Anmerkungen und Übung

MACLANE [MacL37a, MacL37b] gab eine algebraische Charakterisierung planarer Graphen. Memo: Sei $G = (V, E)$ ein Graph mit $|V| = n$, $|E| = m$ und k Zusammenhangskomponenten. Die Menge $\mathcal{E}(G)$ der charakteristischen Funktionen von Kantenmengen $F \subseteq E$ bildet bezüglich der symmetrischen Differenz eine Abelsche Gruppe, die wir als einen m-dimensionalen Vektorraum über $\mathbb{F}_2 = \{0, 1\}$ auffassen können. Ein *Zyklus* $Z \in \mathcal{E}(G)$ ist der Inzidenzvektor einer Kantenmenge $F_Z \subseteq E$, so daß jeder Knoten $v \in V$ mit genau zwei Kanten aus F_Z inzidiert (F_Z ist also die (nicht notwendig zusammenhängende) kantendisjunkte Vereinigung von Kreisen in G. Ein Zyklus Z ist ein Zykel genau dann, wenn F_Z zusammenhängt). Die Menge aller Zyklen $Z \in \mathcal{E}(G)$ spannt einen $[m-(n-k)]$-dimensionalen Unterraum von $\mathcal{E}(G)$ auf, den sogenannten Zyklusraum $\mathcal{Z}(G)$. Sei W ein maximaler spannender Wald in G. Dann definiert jede Kante $e \in E \setminus W$ genau einen Kreis C_e in G. Offensichtlich sind die sogenannten Elementarzyklen $\{C_e : e \in E \setminus W\}$ linear unabhängig und bilden daher eine Basis von $\mathcal{E}(G)$. $\mathcal{Z}$ ist die direkte Summe der Zyklusräume der Blöcke von G. Eine Basis $\{C_1, \ldots, C_r\}$ von $\mathcal{Z}(G)$ heiße *schlicht*, falls jede Kante $e \in E$ nur in höchstens zwei der C_i vorkommt. Existiert eine schlichte Basis von $\mathcal{Z}$, so auch eine aus lauter Kreisen. Man zeige:

Übung 6.1.56 (MACLANEsches Planaritätkriterium)
Ein Graph G ist genau dann planar, wenn $\mathcal{Z}(G)$ eine schlichte Basis besitzt.
[*Hinweis zu „⇐“:* Man nehme o.E. G als zweifach zusammenhängend an und führe die Aussage in zwei Schritten auf den Satz von KURATOWSKI zurück:
(i) für jeden Subgraphen H von G besitzt $\mathcal{Z}(H)$ eine schlichte Basis;
(ii) weder K_5 noch $K_{3,3}$ besitzen eine schlichte Basis (man zähle Kanten).]

Hinreichend ist auch das folgende Kriterium:

Satz 6.1.57 (TUTTE 1963) *Ein 3-zusammenhängender Graph ist planar genau dann, wenn jede Kante zu genau zwei nicht-trennenden Kreisen gehört.* □

Ein weiterer algebraischer Ansatz findet sich in [ABL95]. WU gab ein Gleichungssystems über GF(2) an, das genau dann lösbar ist, wenn der Graph planar ist, siehe [Xu89, Liu90].

Ein hübsches geometrisches Resultat ist die folgende, öfter wiederentdeckte Charakterisierung planarer Graphen.

Satz 6.1.58 (KOEBE 1936) *Ein Graph ist planar genau dann, wenn sich seine Knoten als sich nicht überlappende Kreise in der Ebene darstellen lassen, so daß zwei Knoten benachbart sind genau dann, wenn sich die beiden zugehörigen Kreise berühren („Münzgraph“).*

□

In Erweiterung dieses Ergebnisses konnte auch die folgende Vermutung von TUTTE 1963 bewiesen werden.

Satz 6.1.59 (BRIGHTWELL, SCHEINERMAN 1993) *Ein planarer Graph $G = (V, E)$ ist 3-zusammenhängend genau dann, wenn er und sein Dual G^* gleichzeitig so in die Ebene ($\cup\{\infty\}$) eingebettet werden können, daß alle Kanten geradlinig sind und sich Kanten $e \in E$ mit den entsprechenden Kanten $e^* \in E^*$ rechtwinklig schneiden.* □

Die Planarität eines Graphen läßt sich auch über eine Eigenschaft des zugehörigen Inzidenzposets charakterisieren. Das Inzidenzposet $P_G = (V \cup E, \preceq)$ eines Graphen $G = (V, E)$ hat die Knoten und Kanten des Graphen als Elemente, geordnet durch Inklusion: $x \preceq y$ genau dann, wenn die Kante y den Knoten x enthält. Die Dimension einer teilweisen Ordnung $(P, \preceq)$ ist die minimale Anzahl linearer Erweiterungen von P, deren Schnitt P ist.

Satz 6.1.60 (SCHNYDER 1989) *Ein Graph G ist planar genau dann, wenn die Dimension seines Inzidenzposets P_G höchstens drei ist.* □

Weitere Charakterisierungen der Planarität findet man in [SBW90, Seite 146ff] und [ABL95, CG96].

6.2 Ein Einbettungsalgorithmus

Wir skizzieren hier einen Algorithmus, der einen Graphen G in die Ebene einbettet, falls G planar ist, und andernfalls beweist, daß G nicht planar ist. (Man beachte, daß der Satz von KURATOWSKI nur einen exponentiellen Algorithmus liefert, um zu *entscheiden*, ob ein Graph planar ist.) Unter Benutzung einer solchen Einbettung eines planaren Graphen lassen sich dann etliche kombinatorische Probleme effizient lösen - darauf werden wir in späteren Kapiteln zurückkommen.

Einbettungsalgorithmen für planare Graphen haben eine lange Geschichte. Grundsätzlich gibt es zwei Ansätze für Planaritäts*tests*. Der eine fügt sukzessive Pfade hinzu und geht auf AUSLANDER und PARTER [AP61] bzw. DEMOUCRON, MALGRANGE und PERTUISET [DMP64] zurück. Der Algorithmus sucht zunächst einen Kreis C in G und überprüft anschließend sukzessive die Komponenten von $G - C$. HOPCROFT und TARJAN [HT74] fanden hierfür eine lineare Implementation mittels Tiefensuche. Darstellungen dieses Ansatzes findet man auch in [Meh84, Kuc90, BM76].

Der andere, konzeptuell einfachere Ansatz ist die Methode des sogenannte Knoten-Hinzufügens und wurde von LEMPEL, EVEN und CEDERBAUM [LEC67] entwickelt. Mit Hilfe von PQ-Bäumen haben BOOTH und LUEKER [BL76] hierfür eine lineare Implementierung angegeben (siehe auch [NC88, TS92]).

Übung 6.2.1 *Sei G ein Hamiltonscher Graph mit Hamiltonkreis $C = (v_1, \ldots, v_n)$. Den sogenannten* Sehnengraphen *$S(G,C) = (E \setminus E(C), F)$ (engl. circle graph) zu G und C erhält man, wenn man C auf einem Kreis in die Ebene einbettet und alle Kanten aus $E \setminus E(C)$ als Geradenstücke (Sehnen von C) einfügt. Zwei Sehnen s_1 und s_2 aus $E \setminus E(C)$ sind nun in $S(G,C)$ benachbart genau dann, wenn sie sich als Sehnen von C im Innern der Kreisscheibe schneiden (kreuzen).*

a) *$S(G,C)$ ist wohldefiniert, d.h. unabhängig von der Einbettung von C im Uhrzeiger- oder Gegenuhrzeigersinn in die Ebene.*

b) *Ein Sehnengraph läßt sich auf $\mathcal{O}(n)$ Speicherplatz codieren.*

c) *G ist planar genau dann, wenn $S(G,C)$ bipartit ist.*

d) *Man leite einen einfachen Planaritätstest und Einbettungsalgorithmus für Hamiltonsche Graphen mit gegebenen Hamiltonkreis ab, der Laufzeit $\mathcal{O}(n)$ hat.*

6.3 Die 4-Farben-Vermutung

Eine *Landkarte* ist ein brückenloser, ebener (Multi-) Graph $G = (V, E, R)$. Die Gebiete aus R heißen auch Länder, G selbst auch Gerüst der Landkarte. (Die Länder einer solchen Landkarte besitzen also keine Enklaven.) Färbungen einer Landkarte sind Färbungen ihrer Länder(!), so daß benachbarte Länder (das sind solche, die mindestens eine gemeinsame Grenzkante haben) verschiedene Farben erhalten. Über Jahrhunderte ein Faktum für Kartographen, vermutete GUTHRIE 1852:

4-Farben-Vermutung *Jede Landkarte ist 4-färbbar.*

Kurz nach ihrem Bekanntwerden feierte man schon 1878 ihre Bestätigung durch KEMPE, der daraufhin zum Fellow of the Royal Society gewählt wurde. HEAWOOD war es 1890 fast peinlich, einen Fehler in dessen „Beweis" gefunden zu haben. Er konnte allerdings folgende Abschwächung „retten" (vgl. Übung 6.1.15):

Satz 6.3.1 (5-Farben-Satz) (KEMPE 1879; HEAWOOD 1890)
Jeder planare Graph ist 5-färbbar.

Beweis. Sei $G = (V, E)$ ein planarer Graph. Wir konstruieren induktiv eine 5-Färbung. Für $|V| \leq 5$ ist die Behauptung offensichtlich trivial. Sei also $|V| > 5$. Wir wissen aus Korollar 6.1.14, daß G einen Knoten v vom Grad $d(v) \leq 5$ besitzt. Sei $c : V \to \{1, \ldots, 5\}$ eine 5-Färbung von $G \setminus v$ gemäß Induktionsannahme. Ist die Nachbarschaft $\Gamma(v)$ von v in G mit weniger als 5 Farben gefärbt, so läßt sich c offensichtlich auf v fortsetzen. Enthält $\Gamma(v)$ andernfalls alle 5 Farben, d.h. insbesondere $d(v) = 5$, so seien die Knoten $x_i \in \Gamma(v)$, $i = 1..5$, o.B.d.A. im Uhrzeigersinn jeweils mit Farbe i gefärbt.
Alternative 1: Die bereits mit Farbe i oder j, $1 \leq i < j \leq 5$, gefärbten Knoten von $G \setminus v$ induzieren einen bipartiten Subgraphen G_{ij}, der $\Gamma(v)$ nur in x_i und x_j trifft. Ist x_1 mit x_3 in G_{13} durch einen Pfad verbunden, so schließt sich dieser Pfad über den Knoten v zu einer geschlossene Kurve in der Ebene. Nach dem JORDANschen Kurvensatz kann daher also x_2 nicht mit x_4 durch einen G_{24}-Pfad verbunden sein. Ohne Einschränkung sei x_1 nicht mit x_3 verbunden (sonst nehme man x_2 und x_4). Vertausche die Farben 1 und 3 in der Komponente von G_{13}, die x_1 aber nicht x_3 enthält (dies ist eine sogenannte KEMPE-Kette). Die erhaltene Färbung von $G - v$ ist offenbar noch immer zulässig, aber die Farbe 1 ist nun für den Knoten v frei.
Alternative 2: Da G keinen K_5 enthält, gibt es zwei nicht benachbarte x_i und x_j, $i \neq j$, in $\Gamma(v)$. Der Graph G', der aus $G \setminus v$ durch Identifikation der Knoten x_i und x_j entsteht, ist wieder planar, hat zwei Knoten weniger als G und ist daher nach Induktionsannahme 5-färbbar. Diese 5-Färbung induzierte eine solche auf $G \setminus v$, in der x_i und x_j gleich gefärbt sind. Also verbleibt eine Farbe, um v zu färben. □

Übung 6.3.2 a) *Warum läßt sich mit obigem* KEMPE*-Ketten-Argument nicht die 4-Färbbarkeit planarer Graphen zeigen?*
b) *Man entwerfe einen (rekursiven) Algorithmus, der einen planaren Graphen in $\mathcal{O}(n^2)$ Schritten mit 5 Farben färbt.*[5]

Für Mathematiker erwies sich die 4-Farben-Vermutung als eine harte Nuß, die, obwohl sie zu einem der berühmtesten Probleme des 20. Jahrhunderts avancierte, erst 1976 von AP-

[5] Ein linearer Algorithmus wird sich durch eine Verallgemeinerung des Problems ergeben, siehe Übung 6.3.6.

PEL, HAKEN und KOCH unter massivem Einsatz von Computern geknackt werden konnte. Bis dahin hatte die 4-Farben-Vermutung allerdings wesentlich zur Entstehung einer ganzen mathematischen Disziplin beigetragen: der Graphentheorie. Sie wirkte wie ein Katalysator, initiierte oder zumindest forcierte das Studium ganz unterschiedlicher, graphentheoretischer Konzepte wie Hamiltonkreise, chromatisches Polynom, Graphen höheren Geschlechts, Kantenfärbungen, perfekte Matchings und nirgends-verschwindende Flüsse.

Wir werden in der Folge äquivalente Formulierungen der 4-Farben-Vermutung über Kantenfärbungen bzw. 1-Faktorisierungen und Kreisüberdeckungen, eine Reduktion der 4-Farben-Vermutung auf Hamiltonsche Graphen, eine äquivalente Formulierung über Minoren, die keinen Bezug mehr nimmt zur Planarität, sowie eine den 4-Farben-Satz implizierende Vermutung über nirgends-verschwindende Flüsse kennenlernen.[6]

Während man sich bei der Untersuchung der chromatischen Zahl von Graphen stets auf zweifach zusammenhängende zurückziehen kann (Proposition 5.1.1), zeigt Punkt (6.) unseres nächsten Satzes, daß es bei der Vier-Farben-Vermutung genügt, polyedrische Graphen zu betrachten (vgl. Übung 6.1.32 und Satz 6.1.43). Wegen Proposition 6.1.44 steht (4.) aus Satz 6.3.3 dem in keiner Weise nach.

Eine Landkarte heiße *normal*, wenn sie zusammenhängend ist, weder Schlingen noch Mehrfachkanten enthält und jeder Knoten mindestens Grad 3 hat. Eine Landkarte heißt *kubisch*, wenn ihr Gerüstgraph 3-regulär ist. Wir wollen uns zunächst folgende, einfache Reduktionen bzw. Umformulierungen der 4-Farben-Vermutung ansehen:

Satz 6.3.3 *Folgenden Aussagen sind äquivalent:*

1. *Jede Landkarte ist 4-färbbar.*
2. *Jede normale Landkarte ist 4-färbbar.*
3. *Jede kubische, normale Landkarte ist 4-färbbar.*
4. *Jede kubische, normale Landkarte ohne mehrfache Nachbarschaften ist 4-färbbar.*
5. *Jeder planare Graph ist 4-färbbar.*
6. *Jede Triangulation der Ebene ist 4-färbbar.*

Beweis. (1.) $\Rightarrow$ (2.) $\Rightarrow$ (3.) ist trivial. Offenbar kann eine Landkarte auch ohne Einschränkung als zusammenhängend angenommen werden: man passe ggf. die Färbungen der Zusammenhangskomponenten hinsichtlich des Außengebietes aneinander an. Schlingen und Mehrfachkanten entfernt man durch Einfügen von Knoten und Knoten vom Grad 2 mittels Ersetzen durch einen $K_4 - e$. Ist dann die so entstandene, normale Landkarte 4-färbbar, dann auch die ursprüngliche.
Knoten vom Grad ≥ 4 wiederum ersetzt man so durch ein kleines Land, daß alle mit dem Knoten inzidierenden Kanten die Grenze dieses neuen Landes an verschiedenen Stellen treffen. Dann ist die modifizierte Landkarte offenbar kubisch und normal. Ist sie darüberhinaus 4-färbbar, so sicherlich auch die ursprüngliche Landkarte.
Schließlich kann man eine solche kubische, normale Landkarte auch ohne mehrfache Nachbarschaften annehmen, indem man eine entsprechende Grenzkante einfach durch ein

[6] In [Kau90] wird eine Formulierung über das Vektor-Kreuzprodukt im R^3 hergeleitet.

„schmales, rechteckiges Land“ ersetzt, das die beiden dort vorher benachbarten Länder nun trennt.
(5.) entsteht durch Dualisierung aus (1.): das Dual eines brückenlosen, ebenen (Multi-) Graphen (einer Landkarte) ist ein schlingenfreier, ebener Graph, dessen Knoten den Ländern der Landkarte entsprechen. Ist dieser Graph (etwaige Mehrfachkanten identifiziere man) also 4-färbbar, so sicher auch die ursprüngliche Landkarte, was (5.) $\Rightarrow$ (1.) zeigt. Die Umkehrung folgt analog.
(6.) folgt direkt aus (5.) und ist nun gerade die dualisierte Fassung von (4.). Aber auch (6.) $\Rightarrow$ (5.) ist leicht einzusehen: ist doch jeder ebene Graph Subgraph einer Triangulation (man füge solange Kanten ein). □

Ein Graph ist zweifärbbar genau dann, wenn er bipartit ist. Es gibt aber schon dreiecksfreie Graphen von beliebig hoher chromatischer Zahl. Im Fall planarer, dreiecksfreier Graphen (die ja bereits recht dünn sind: $|E| \leq 2|V|-4$), bewies GRÖTZSCH den berühmten Dreifarbensatz (Beweise findet man in [Ste93, Tho94c]):

Satz 6.3.4 (GRÖTZSCH 1959) *Jeder dreiecksfreie, planare Graph ist 3-färbbar.* □

Übung und Anmerkung

Eine *Listen-Färbung* eines Graphen ist eine zulässige Knotenfärbung des Graphen, bei der jeder Knoten $v \in V$ eine Farbe aus einer vorgegebenen Liste $L(v)$ von Farben zugewiesen bekommt. Ein Graph heißt *k-choosable*, wenn eine solche Färbung für *jede* Vorgabe von Listen $L(v)$, $v \in V$, existiert, bei der jede Liste $L(v)$ mindestens k Farben enthält. Insbesondere ist ein *k-choosable* Graph also k-färbbar, Offenbar ist jeder Graph n-choosable. Die kleinste Zahl k, so daß ein Graph G k-choosable ist, heißt die *Listen-chromatische Zahl* (englisch auch choice-number) von G, bezeichnet mit $\chi_\ell(G)$ oder $ch(G)$. Der Greedy-Algorithmus zeigt $\chi_\ell(G) \leq \Delta(G)+1$. Jedoch läßt sich $\chi_\ell(G)$ nicht durch eine Funktion in $\chi(G)$ nach oben beschränken:

Übung 6.3.5 (ERDŐS, RUBIN, TAYLOR 1979) *Sei $k \in \mathbb{N}$ und $r := \binom{2k-1}{k}$. Dann ist der Graph $K_{r,r}$ nicht k-choosable.*

In letzter Zeit gab es bemerkenswerte Ergebnisse zur Choosability in planaren Graphen. ERDŐS, RUBIN und TAYLOR vermuteten, daß jeder planare Graph 5-choosable, aber nicht jeder 4-choosable ist. VOIGT [Voi93] konstruierte eine unendliche Familie planarer Graphen, die nicht 4-choosable sind. Der andere Teil der Vermutung wurde von THOMASSEN [Tho94b] bewiesen und stellt den vielleicht elementarsten Beweis des 5-Farben-Satzes dar (der Beweis erfordert weder eine Umfärbung, noch benutzt er die Eulersche Formel). Der Trick besteht darin, eine passende Verschärfung zu beweisen.

Übung 6.3.6 a) *Sei $G = (V, E, R)$ eine Semi-Triangulation der Ebene, d.h. ein ebener Graph, dessen innere Gebiete von Dreiecken und dessen Äußeres von einem Kreis $C = (v_1, \ldots, v_p)$ auf $p \geq 3$ Knoten berandet wird. Angenommen, die Knoten v_1 und v_2 sind mit den Farben 1 und 2 gefärbt und $L(v)$ ist eine Liste von mindestens 3 Farben, falls $v \in C - v_1 - v_2$, bzw. von mindestens 5 Farben, falls $v \in G - C$. Dann läßt sich die Färbung von v_1 und v_2 auf ganz G fortsetzen.* [*Hinweis:* Induktion nach n, betrachte Sehnen]
b) *Man entwickle daraus einen entsprechenden Listenfärbungsalgorithmus für planare Graphen mit Laufzeit $\mathcal{O}(n)$.* [7] [*Hinweis:* Man nehme zunächst an, der Graph sei bereits eine Semi-Triangulation]
c)[?] *Dreiecksfreie planare Graphen sind 4-choosable.*

[7] Die Eleganz dieses Algorithmus wird auch deutlich, wenn man ihn mit früheren linearen 5-Färbungsalgorithmen für planare Graphen vergleicht, siehe z.B. [NC88].

Auch Teil c) ist bestmöglich in dem Sinn, daß es dreiecksfreie planare Graphen gibt, die nicht 3-choosable sind [?]. Das Listen-Färbungsproblem ist selbst auf planaren Graphen vom Maximalgrad 3 NP-vollständig, wenn alle Listen nur höchstens 3 Elemente enthalten [KT94a].

7.3.1 Kantenfärbungen und 1-Faktorisierungen

Nach dem Satz 5.6.3 von VIZING nimmt der chromatische Index $\chi'(G)$ eines Graphen G vom Maximalgrad $\Delta(G)$ den Wert $\Delta(G)$ oder den Wert $\Delta(G)+1$ an.

Satz 6.3.7 (TAIT 1880b) *Ein kubischer, ebener Graph G ist 3-kantenfärbbar genau dann, wenn er eine 4-Färbung seiner Gebiete besitzt:*

$$\chi'(G) = 3 \quad \Leftrightarrow \quad \chi(G^*) \leq 4.$$

Eine solche Kantenfärbung heißt auch TAIT-Färbung.

Beweis. Ein kubischer Graph mit einer Brücke ist „Klasse 2“: $\chi'(G) = \Delta + 1$ (vgl. Übung 5.6.15) und daher weder zulässig 3-Kanten-färbbar noch zulässig 4-Flächen-färbbar. Sei G also zweifach kantenzusammenhängend.
„$\Leftarrow$“: Sei Δ eine Färbung der Gebiete von $G = (V, E, R)$ mit den Elementen der KLEINschen Vierergruppe $\mathbb{Z}_2 \times \mathbb{Z}_2 = \{(0,0),(0,1),(1,0),(1,1)\}$.[8] Betrachte die Funktion $\Delta' : E \to \mathbb{Z}_2 \times \mathbb{Z}_2$ definiert durch

$$\Delta'(e) = \Delta(x) - \Delta(y),$$

wobei x und y die an die Kante $e \in E$ angrenzenden Gebiete seien. Da in $\mathbb{Z}_2 \times \mathbb{Z}_2$ jedes Element sein eigenes Inverses ist, ist Δ' wohldefiniert. Die Zulässigkeit der Färbung Δ impliziert $\Delta'(e) \neq (0,0)$ für alle $e \in E$. Um zu zeigen, daß die Färbung Δ' auch eine zulässige 3-Färbung von E ist, betrachten wir die drei in einem Knoten v inzidierenden Kanten $e, f, g \in E$. Sind $x, y, z \in R$ die angrenzenden Gebiete, so gilt:

$$\begin{aligned} 0 &= (\Delta(x) - \Delta(y)) + (\Delta(y) - \Delta(z)) + (\Delta(z) - \Delta(x)) \\ &= \Delta'(e) + \Delta'(f) + \Delta'(g). \end{aligned}$$

Wären also beispielsweise e und f gleichgefärbt, so folgte $\Delta'(g) = 0$ – Widerspruch.
„$\Rightarrow$“: Sei $\Delta' : E \to \mathbb{Z}_2 \times \mathbb{Z}_2 \setminus \{0\}$ eine zulässige 3-Färbung der Kanten von G. Übertrage die Färbung Δ' auf die Kantenmenge E^* des geometrischen Duals $G^* = (R, E^*)$ von G. Wir werden nun eine Färbung $\Delta : R \to \mathbb{Z}_2 \times \mathbb{Z}_2$ der Knoten von G^* angeben. Beachte: G^* ist eine Triangulation. Da alle drei in einem Knoten $v \in V$ inzidierenden Kanten verschieden gefärbt sind, gilt für das Dreieck T in G^*, das v umschließt:

$$\sum_{e^* \in T} \Delta'(e^*) = \sum_{e \cap v \neq \emptyset} \Delta'(e) = 0. \tag{6.1}$$

Wir färben nun den Knoten $R_0 \in R$, der dem äußeren Gebiet von G entspricht, mit Farbe 0 und definieren: ist $R_k \in R$, $R_k \neq R_0$, ein Knoten und $R_0, R_1, \ldots, R_k$ ein $R_0 - R_k$–Pfad in G^*, so sei

$$\Delta(R_k) := \sum_{i=1}^{k} \Delta'(\{R_{i-1}, R_i\}).$$

[8] Die Elemente dieser Gruppe werden komponentenweise modulo 2 addiert.

Δ ist wohldefiniert: sind P und Q zwei verschiedene $R_0 - R_k$–Pfade, so triangulieren die Kanten von G^* das von P und Q eingeschlossene Gebiet int der Ebene [das sich, falls sich P und Q überschneiden, aus mehreren einfach zusammenhängenden Gebieten zusammensetzt]. Da jede in int enthaltene Kante $e^* \in E^*$ zum Rand von genau zwei Dreiecken $T_1(e^*) \subset int$ und $T_2(e^*) \subset int$ gehört, gilt nach (6.1)

$$\sum_{e^* \in P \cup Q} \Delta'(e^*) \;=\; \sum_{T \subset int} \sum_{e^* \subset T} \Delta'(e^*) \;=\; 0.$$

Also gilt $\sum_{e^* \in P} \Delta'(e^*) = \sum_{e^* \in Q} \Delta'(e^*)$. Daß Δ schließlich eine zulässige Färbung der Gebiete R darstellt, ersieht man daraus, daß sich Δ auf zwei benachbarten Gebieten R_1 und R_2 genau um den Wert $\Delta'(\{R_1, R_2\}) \neq 0$ unterscheidet. □

Die 4-Farben-Vermutung läßt sich somit auch über Kantenfärbungen formulieren:

Korollar 6.3.8 *Die 4-Farben-Vermutung ist äquivalent dazu, daß alle kubischen, brükkenlosen, planaren Graphen „Klasse 1" : $\chi'(G) = \Delta(G)$ (oder 1-faktorisierbar) sind.* □

Derjenige Leser, der den Computerbeweis des 4-Farben-Satzes nicht akzeptiert, sei an dieser Stelle ermuntert, eine TAIT-Färbung des Dodekaeders zu finden! Man beachte, daß der PETERSEN-Graph (siehe Seite 88) kubisch und brückenlos, aber nicht 1-faktorisierbar ist.

Übung und Anmerkung

Übung 6.3.9 (HEAWOOD 1898) **a)** *Die 4-Farben-Vermutung ist genau dann wahr, wenn es für jeden kubischen, brückenlosen, ebenen Graphen $G = (V, E, R)$ eine Funktion $\phi : V \to \{-1, 1\}$ gibt, so daß die Knoten auf dem Rand eines jeden Gebietes $F \in R$ erfüllen:*

$$\sum_{v \in \partial F} \phi(v) = 0 \pmod 3.$$

[*Hinweis:* interpretiere die Werte 1 und -1 als Orientierungen und benutze die TAIT-Färbbarkeit.] **b)** *Man gebe einen linearen Algorithmus an, der jede Triangulation der Ebene, deren Knotengrade sämtlich durch 3 teilbar sind, mit 4 Farben färbt.*

Es ist ein offenes Problem, ob das Entscheidungsproblem CHROMATIC INDEX – nach Satz 5.6.12 NP-vollständig für kubische, brückenlose Graphen – auch für planare Graphen schwer bleibt. VIZING [Viz65a] bewies immerhin, daß planare Graphen vom Maximalgrad $\Delta \geq 8$ Klasse 1 sind (siehe auch [FW77]). In [NC88] findet sich darüberhinaus ein quadratischer Algorithmus für dieses Problem, vgl. auch [CN90].

7.3.2 Kreis- und Zykelüberdeckungen

Ein brückenloser, kubischer Graph ist offenbar 1-faktorisierbar genau dann, wenn er eine Kreisüberdeckung der Knotenmenge aus geraden Kreisen besitzt. Aus Korollar 6.3.8 erhält man also:

Korollar 6.3.10 *Die 4-Farben-Vermutung ist äquivalent dazu, daß die Knoten eines kubischen, brückenlosen, planaren Graphen durch gerade Kreise überdeckt werden können.* □

Vergleiche auch Übung 5.6.19. Folgende Aussage ist eine einfache

Übung 6.3.11 (TAIT 1880a)
Eine Landkarte ist genau dann 2-färbbar, wenn ihr Gerüstgraph Eulersch ist.
[*Hinweis* zu „⇐“: Zeige, daß die Blöcke des Duals keine ungeraden Kreise enthalten.]

Damit verallgemeinert man Lemma 6.1.48 leicht wie folgt:

Lemma 6.3.12 *Eine Kantenmenge Z in einem ebenen Graphen $G = (V, E)$ ist genau dann ein Zykel, wenn die Z entsprechende Kantenmenge Z^* im Dual $G^* = (V^*, E^*)$ von G ein Schnitt ist, d.h. wenn es eine nichtleere Menge $S^* \subset V^*$ gibt, so daß $Z^* = \{\{x, y\} \in E^* : x \in S^*, y \in V^* \setminus S^*\}$.* □

Wir erhalten daraus unschwer die

Proposition 6.3.13 *Ein planarer Graph ist genau dann 4-Flächen-färbbar, wenn seine Kantenmenge die Vereinigung zweier Zyklen ist.*

Wie aus Übung 6.3.11 ersichtlich, dürfen die Zyklen dabei identisch sein.

Beweis. „⇒“: Sei $\Delta : V^* \to \{1, 2, 3, 4\}$ eine 4-Färbung der Gebiete des ebenen Graphen $G = (V, E, V^*)$. Die den Schnitten $S_1^* := \{x \in V^* : \Delta(x) \in \{1, 2\}\}$ und $S_2^* := \{x \in V^* : \Delta(x) \in \{1, 3\}\}$ entsprechenden Kantenmengen in E sind die gesuchten Zyklen.
„⇐“: Wir färben die Gebiete von G mit den Elementen der KLEINschen Vierergruppe. Nach Übung 6.3.11 induziert jeder der beiden Zyklen Z_i, $i = 1, 2$, eine 2-Färbung $\Delta_i : V^* \to \{0, 1\}$ der Gebiete von G. Dann ist $\Delta : V^* \to \mathbb{Z}_2 \times \mathbb{Z}_2$ mit $\Delta(x) = (\Delta_1(x), \Delta_2(x))$, $x \in V^*$, eine 4-Färbung von V^* genau dann, wenn $E = Z_1 \cup Z_2$. □

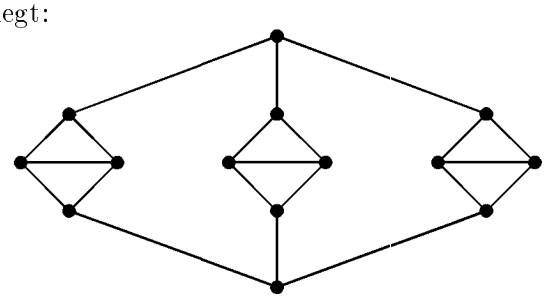
7.3.3 Hamiltonkreise

Wir beobachten zunächst:

Proposition 6.3.14 *Die Gebiete eines Hamiltonschen, ebenen Graphen sind 4-färbbar.*

Beweis. Der Hamiltonkreis C von G zerlegt die Ebene in Inneres und Äußeres. Wir überlegen uns, daß wir jeweils die inneren wie die äußeren Gebiete mit zwei Farben färben können. Das Innere von C enthält keine Knoten von G, daher hat das Dual G^* dort keinen Kreis. Ein Wald ist aber zweifärbbar. □

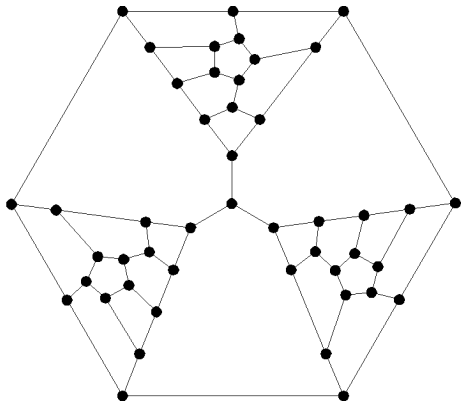
Es ist natürlich bei weitem nicht jede Landkarte Hamiltonsch, und die Schwierigkeit liegt gerade darin, Bedingungen zu finden, die einen Hamiltonkreis garantieren.
Ein Hamiltonkreis ist eine Kreisüberdeckung wie in Korollar 6.3.10. Es würde also die Richtigkeit der 4-Farben-Vermutung folgen, wenn nur jeder kubische, zusammenhängende, brückenlose, ebene Graph Hamiltonsch wäre. Leider ist dies jedoch nicht der Fall, wie schon der folgende Graph belegt:



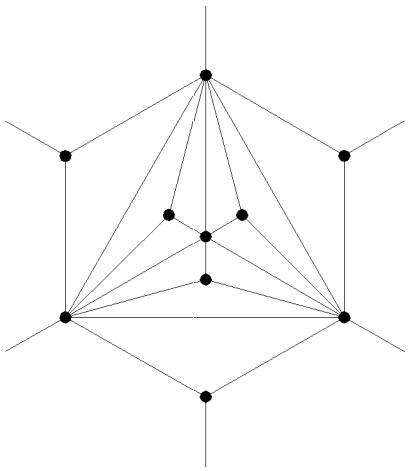
TAIT vermutete allerdings, daß es hier bereits genügte, zusätzlich dreifachen Zusammenhang zu fordern:

Vermutung (TAIT 1884)
Jeder kubische, 3-zusammenhängende, planare Graph besitzt einen Hamiltonkreis.

Wir erinnern uns, daß dies nach Satz 6.1.43 gerade die kubischen, polyedrischen Graphen sind. Man bestätige die Vermutung am Gerüstgraphen des Dodekaeders! Dies genau war die Aufgabe bei HAMILTONs Dodekaederspiel von 1859, das die Problemstellung, einen Hamiltonkreis in einem Graphen zu finden, popularisierte. Daß die Aussage der Vermutung für nicht planare Graphen dieser Art falsch wird, sieht man am PETERSEN-Graph (s. Seite 88). Er ist „Klasse 2“: $\chi'(G) = \Delta(G) + 1$ und daher sicher nicht Hamiltonsch, vgl. Übung 5.6.19.
Bedeutung gewann die Vermutung von TAIT dadurch, daß ihre Richtigkeit auch schon die der 4-Farben-Vermutung implizieren würde. Denn nach Übung 6.1.32 und Proposition 6.1.44 sind kubische, normale Landkarten ($n \geq 4$) ohne mehrfache Nachbarschaften 3-zusammenhängend und daher unter Annahme der Richtigkeit der Vermutung von TAIT Hamiltonsch. Der 4-Farben-Satz folgte aus Korollar 6.3.10 oder mit 6.3.14 aus Satz 6.3.3(4.). Leider ist die Vermutung aber falsch, wie erst 1946 TUTTE durch folgenden Graphen zeigte:



Nachzuweisen, daß er nicht Hamiltonsch ist, ist hier schon keine leichte Aufgabe mehr (vgl. Übung 6.3.20). WHITNEY gelang die Reduktion der 4-Farben-Vermutung auf Hamiltonsche Graphen aber auf dem Wege der Dualisierung der Vermutung von TAIT. Das Dual eines kubischen, 3-zusammenhängenden, ebenen Graphen ist eine Triangulierung. Eine solche braucht allerdings noch keinen Hamiltonschen Kreis zu besitzen - kleinstes Gegenbeispiel ist der folgende Graph:



Die Halbgeraden sollen sich hier im Knoten „∞“ treffen. Welches Kriterium beweist leicht, daß der Graph nicht Hamiltonsch ist? Man verifiziere, daß sein Dual *kein* Gegenbeispiel zur Vermutung von TAIT ist. Es gilt aber der

Satz 6.3.15 (WHITNEY 1931)
Eine 4-zusammenhängende Triangulation besitzt einen Hamiltonkreis. □

Aus diesem Satz können wir mit Hilfe des folgenden Lemmas eine Reduktion der 4-Farben-Vermutung ableiten.

Lemma 6.3.16 *Eine Triangulation auf mindestens 5 Knoten ist entweder 4-zusammenhängend oder besitzt ein trennendes Dreieck.*

Beweis. Sei $T \subset V$ eine minimal trennende Knotenmenge in $G = (V, E)$. Sei A eine Komponente in $G \setminus T$ und $G \setminus T = A \cup B$ mit $A \cap B = \emptyset$. Da G eine Triangulation ist, bilden die Nachbarn eines Knotens $v \in T$ einen Kreis C. Da T minimal trennend, hat v Nachbarn x_A und x_B in $A \cap C$ bzw. $B \cap C$. Die Nachbarn von x_A auf C liegen nun entweder wieder in A oder aber in T, da es zwischen A und B keine Kanten gibt. Da man in beiden Richtungen auf dem Kreis C schließlich x_B erreicht, muß v mindestens zwei Nachbarn in T haben. Hätte v sogar drei Nachbarn $x_1, x_2, x_3 \in T \cap C$, so enthielte G schon eine Unterteilung des $K_{3,3}$ mit Bipartition $\{x_1, x_2, x_3\}$ und $\{v, a, b\}$, wo $a \in A$, $b \in B$. Also hat jeder Knoten in $G[T]$ den genauen Grad zwei, d.h. T ist ein Kreis. Daraus folgt zunächst $|T| \geq 3$, und wir erhalten einen neuen Beweis dafür, daß eine Triangulation 3-zusammenhängend ist (Satz 6.1.32). Ist zudem die Zusammenhangszahl $\kappa(G) = 3$, so muß T offenbar ein Dreieck K_3 sein. □

Korollar 6.3.17 *Die 4-Farben-Vermutung ist genau dann richtig, wenn alle Hamiltonschen, planaren Graphen 4-färbbar sind.*

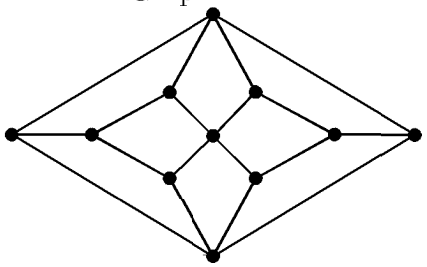
Beweis. Angenommen, alle Hamiltonschen, planaren Graphen seien 4-färbbar. Nach Satz 6.3.3 genügt es, zu zeigen, daß dann jede Triangulation $G = (V, E)$ der Ebene 4-färbbar ist. Wir induzieren über $n = |V|$. Eine Triangulation auf drei oder vier Knoten ist sicher 4-färbbar. Ist nun eine Triangulation G mit $n \geq 5$ sogar 4-zusammenhängend, so ist G nach Satz 6.3.15 Hamiltonsch und daher nach Voraussetzung 4-färbbar. Andernfalls ist $\kappa(G) = 3$, und G besitzt nach obigem Lemma ein trennendes Dreieck T. $G \setminus T$ zerfällt in genau zwei Komponenten I und A, denn wären es mehr, gäbe es wieder eine Unterteilung

des $K_{3,3}$. Wie man sich überlegt, muß eine Komponente in Innern des Dreiecks liegen, die andere im Äußeren. Die Graphen $G[I \cup T]$ und $G[A \cup T]$ sind nun beide wieder Triangulationen und somit nach Induktionsvoraussetzung 4-färbbar. Da in beiden Färbungen alle Knoten von T verschieden gefärbt sind, setzt man sie (eventuell nach Permutation der Farben) zu einer 4-Färbung von G zusammen. □

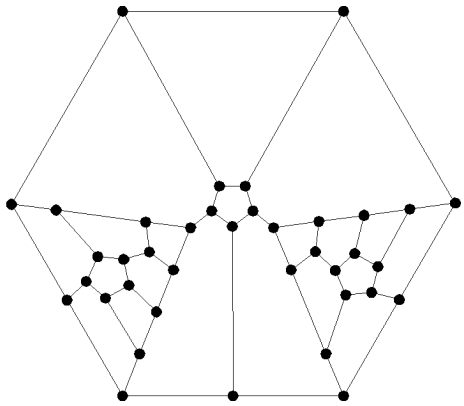
Vergleiche Proposition 6.3.14! Satz 6.3.15 läßt sich wie folgt verschärfen.

Satz 6.3.18 (TUTTE 1956; THOMASSEN 1983)
Jeder 4-zh., planare Graph ist Hamiltonsch-zusammenhängend. □

Dieser Satz sowie der obige von WHITNEY sind recht tiefliegend. Letzterer wird falsch für allgemeine 4-zusammenhängende Graphen, wie z.B. den $K_{4,5}$, und für nur 3-zusammenhängende, planare Graphen, wie die Triangulation aus obiger Figur, die ein trennendes Dreieck besitzt. Ein kleinstes [BJ70] Beispiel eines nicht-Hamiltonschen, polyedrischen Graphen ist der HERSCHEL-Graph:



Einen zudem kubischen, nicht-Hamiltonschen, polyedrischen Graphen kennen wir bereits: der TUTTE-Graph auf 46 Knoten. Ein minimaler Graph dieser Art ist der folgende Graph auf 38 Knoten (siehe [HM88]):



Das folgende Ergebnis zeigt uns allerdings, daß es (unter der Annahme P $\neq$ NP) kein im Sinne der Komplexitätstheorie einfaches Kriterium zur Charakterisierung Hamiltonscher planarer Graphen geben kann.

Satz 6.3.19 *Das Problem* HAMILTONIAN CIRCUIT *bleibt selbst bei Einschränkung auf die folgenden Graphenklassen* NP*-vollständig:*

- *kubische, polyedrische Graphen* [GJT76]*;*
- *Triangulationen der Ebene* [Wig82]. □

Im Kontrast dazu konnten CHIBA und NISHIZEKI, aufbauend auf THOMASSENS kurzem Beweis von Satz 6.3.18, einen sogar *linearen* Algorithmus angeben, der einen Hamiltonkreis in einem 4-zusammenhängenden, planaren Graphen - wie ihn Satz 6.3.18 garantiert - findet, siehe [NC88].

Übung 6.3.20 (GRINBERG 1968) **a)** *Sei G ein ebener Graph mit Hamiltonkreis C. Bezeichne r_k^i bzw. r_k^a, $1 \leq k \leq n$, die Anzahl Gebiete, deren Rand genau k Kanten enthält, im Inneren bzw. im Äußeren von C. Dann gilt*

$$\sum_{k=3}^{n} (k-2)(r_k^i - r_k^a) \;=\; 0.$$

b) *Zeige mit Hilfe des* GRINBERG*-Kriteriums, daß der* TUTTE*-Graph auf Seite 194 nicht Hamiltonsch ist.*

7.3.4 Die Hadwiger-Vermutung

Planare Graphen konnten wir dadurch charakterisieren, daß sie weder den K_5 noch den $K_{3,3}$ als Minor enthalten, vgl. Korollar 6.1.31. Auch für orientierbare Flächen höheren Geschlechts existieren solche Familien verbotener Subgraphen, vgl. Satz 6.4.19. Es erhebt sich die Frage, was man über das Vorhandensein bestimmter Konfigurationen in Graphen mit hoher chromatischer Zahl aussagen kann. Die folgende Vermutung zählt heute zu den zentralen offenen Problemen der Graphentheorie.

Vermutung (HADWIGER 1943)
Sei G ein Graph. Für alle $k \in \mathbb{N}$ ist die folgende Aussage richtig

$$(H_k) \;:\quad \chi(G) \geq k \;\Rightarrow\; G \succ K_k.$$

(H_1) und (H_2) sind offensichtlich trivial. (H_3) besagt gerade, daß ein 3-chromatischer Graph einen Kreis besitzen muß.

Satz 6.3.21 (HADWIGER 1943; DIRAC 1952b)
(H_4) ist wahr, d.h. jeder Graph G mit chromatischer Zahl $\chi(G) \geq 4$ hat den K_4 als Minor.

Beweis nach [Woo92]. Wir beweisen über Induktion nach $|V|$, daß jeder Graph $G = (V, E)$, der den K_4 nicht als Minor hat, 3-färbbar ist. Für Graphen der Ordnung höchstens 4 ist dies offenbar richtig. Sei nun G von der Ordnung $n \geq 5$ ohne K_4-Minor. Nach Proposition 5.1.1 kann G o.B.d.A. als zweifach zusammenhängend angenommen werden. Sei $e = \{r, b\}$ eine Kante in einem Kreis von G und $S \subset V$ eine minimale r-b-trennende Menge in $G \setminus e$. Dann ist $S \neq \emptyset$ stabil, denn sonst enthielte G einen zum K_4 homöomorphen Subgraphen. Setze $G \setminus e = G_r \cup G_b$ mit G_r, G_b zusammenhängend, $G_r \cap G_b = S$, $r \in G_r$ und $b \in G_b$ (dies ist aufgrund der Minimalität von S möglich). Kontrahiere G_r auf einen Knoten $s \in S$ und erhalte den (echten) Minor G_b' von G. Ebenso erhalte durch Kontraktion von G_b auf einen Knoten $s \in S$ den (echten) Minor G_r' von G. Da auch G_b' und G_r' den K_4 nicht als Minor enthalten, besitzen beide Graphen nach Induktionsvoraussetzung jeweils eine 3-Färbung. Da ferner die Knoten b bzw. r in G_b' bzw. G_r' jeweils zu s verbunden sind, ist o.B.d.A. in der 3-Färbung von G_b' b blau und s schwarz gefärbt und in der 3-Färbung von G_r' r rot und s schwarz gefärbt. Also lassen sich beide Färbungen zu einer 3-Färbung von G zusammensetzen. □

Ein weiterer Beweis wird sich aus Satz 9.1.14 ableiten.
Die Schwierigkeit von (H_k) nimmt höchstens zu mit wachsendem k:

Proposition 6.3.22 $(H_{k+1}) \Rightarrow (H_k)$.

Beweis. Sei G ein Graph mit $\chi(G) \geq k$. Gilt sogar $\chi(G) \geq k+1$, so folgt offenbar mit (H_{k+1}) auch $G \succ K_k$. Ist aber $\chi(G) = k$, so ist der Graph $G * 1$ (das ist G mit einem zusätzlichen Knoten, der mit allen Knoten von G verbunden wird) $(k+1)$-chromatisch, und aus (H_{k+1}) folgt $G * 1 \succ K_{k+1} = K_k * 1$. Somit gilt wiederum $G \succ K_k$. □

Die Tragweite schon von (H_5) war bereits HADWIGER klar: aus (H_5) folgt der 4-Farben-Satz: denn angenommen, ein planarer Graph wäre 5-chromatisch, dann enthielte er nach (H_5) schon den K_5 als Minor, im Widerspruch zu Korollar 6.1.31. Überraschend war jedoch:

Satz 6.3.23 (WAGNERscher Äquivalenzsatz, 1937/1960)

$$(H_5) \Leftrightarrow \textit{4-Farben-Vermutung}$$

Dies war einer der frühesten Versuche, das 4-Farben-Problem zu enttopologisieren (kein Bezug mehr zur Planarität!). Wie YOUNG [You71] bemerkte, ergibt sich ein einfacher Beweis dieses Satzes aus dem folgenden Lemma, das man (wegen des Satzes von KURATOWSKI) durch Fallunterscheidung an einer Unterteilung des $K_{3,3}$ beweist (siehe auch [Bol78a, Lemma V.3.10]).

Lemma 6.3.24 (HALIN 1964)
Ein 4-zusammenhängender, nicht-planarer Graph enthält den K_5 als Minor. □

Beweis von Satz 6.3.23:
Gelte also die 4-Farben-Vermutung, und sei G ein Graph mit $\chi(G) \geq 5$. Dann enthält G einen 5-chromatischen Minor H, so daß alle echten Minoren von H 4-färbbar sind. Nach Annahme ist H nicht planar. Wegen des vorausgegangenen Lemmas genügt es also, zu zeigen, daß H 4-zusammenhängend ist. Sei $T \subset V(H)$ eine minimale H-trennende Knotenmenge. Angenommen, es gälte $|T| \leq 3$. Seien $C_1, \ldots, C_k$, $k \geq 2$, die Komponenten von $H - T$, sei $H_i := H[C_i \cup T]$, $i = 1, \ldots, k$, sowie $S \subseteq T$ eine kardinalitätsmaximale stabile Menge in T. Wir zeigen, daß dann jedes H_i, $i = 1, \ldots, k$, eine 4-Färbung besitzt, in der alle Knoten aus S in einer Farbe und die Knoten aus $T \setminus S$ in $|T \setminus S|$ weiteren Farben gefärbt sind. Diese 4-Färbungen der H_i, $i = 1, \ldots, k$, lassen sich somit durch Permutation der Farben auf T zu einer von ganz H zusammensetzen – im Widerspruch zu $\chi(H) = 5$.
Zur 4-Färbung von H_i, $1 \leq i \leq k$. Zunächst bilden nach Wahl von S die restlichen Knoten $T \setminus S$ in T eine Clique (der Kardinalität 0, 1 oder 2). Wähle einen Index $j \neq i$, $1 \leq j \leq k$, und betrachte den (echten) Minor von H, der aus $H[C_i \cup C_j \cup T]$ durch Kontraktion des Subgraphen $H[C_j \cup S]$ auf einen Knoten s entsteht (beachte: da T minimal trennend gewählt wurde, ist $H[C_j \cup S]$ zusammenhängend). Nach Wahl von H besitzt dieser Minor eine 4-Färbung, und in dieser 4-Färbung sind die Knoten $(T \setminus S) + s$, weil sie eine Clique bilden, paarweise verschieden gefärbt. Diese Färbung induziert auf H_i die gewünschte Färbung. □

Einen weiteren Beweis kann man in [WB89, II,Satz 8.13] nachlesen. Wie sich herausstellt, ist (H_6) ebenfalls äquivalent zur 4-Farben-Vermutung [RST93]. WAGNER konnte folgende Abschwächung der HADWIGER-Vermutung zeigen.

Satz 6.3.25 (WAGNER 1964) *Es gibt eine Funktion $\phi : \mathbb{N} \to \mathbb{N}$, so daß gilt:*

$$\chi(G) \geq \phi(k) \Rightarrow G \succ K_k$$

und $\phi(k) \leq 2^{k-1}$.

Da jeder k-chromatische Graph G einen Subgraphen H mit $\delta(H) \geq k-1$ enthält, folgt dieser Satz unmittelbar aus dem folgenden, der lediglich die Kantendichte ausnutzt.

Satz 6.3.26 (MADER 1967) *Es gibt eine Funktion* $\psi : \mathbb{N} \to \mathbb{N}$, *so daß gilt:*

$$d(G) \geq \psi(k) \;\Rightarrow\; G \succ K_k$$

und $\psi(k) \leq 2^{k-1} - 1$, *wobei* $d(G) := \frac{1}{n}\sum_{v \in V} d(v)$ *den Durchschnittsgrad in* G *bezeichnet.*

Beweis. Wir induzieren nach k. Wegen $\psi(1) = 0$, $\psi(2) = 1$ und $\psi(3) = 2$ ist die Behauptung für $k = 1, 2, 3$ offenbar richtig. Sei also $k \in \mathbb{N}$ derart, daß für alle Graphen H gilt: $d(H) \geq 2^{k-1} - 1 \Rightarrow H \succ K_k$. Wir zeigen nun, daß dann jeder Graph $G = (V, E)$ mit $d(G) \geq 2^k - 1$ einen K_{k+1}-Minor besitzt. O.B.d.A. sei G zusammenhängend. Aufgrund der Bedingung an den Durchschnittsgrad hat G offenbar

$$m \;=\; \frac{n\, d(G)}{2} \;\geq\; (2^{k-1} - \frac{1}{2})n \;>\; (2^{k-1} - 1)n$$

viele Kanten. Für eine Knotenmenge $Z \subseteq V$, so daß $G[Z]$ zusammenhängt, bezeichne G_Z den Graphen, der aus G durch Kontraktion der Knoten aus Z zu einem einzigen Knoten entsteht. Sei $Z \subseteq V$ eine inklusionsweise maximale Knotenmenge in G mit der Eigenschaft, daß $G[Z]$ zusammenhängt und $m(G_Z) > (2^{k-1}-1)n(G_Z)$ gilt (warum existiert Z?). Wegen $Z \neq V$ wird das „Zentrum“ Z in G von einem Subgraphen $H := G[\Gamma(Z) \setminus Z]$ „umhüllt“. Betrachte den Grad $d_H(v)$ eines beliebigen Knotens v im Graphen H: Nach Wahl von Z gilt einerseits

$$m(G_{Z+v}) \;\leq\; (2^{k-1} - 1)n(G_{Z+v}) \;=\; (2^{k-1} - 1)(n(G_Z) - 1),$$

andererseits ist offenbar

$$m(G_{Z+v}) \;=\; m(G_Z) - (d_H(v) + 1) \;\geq\; (2^{k-1} - 1)n(G_Z) - d_H(v),$$

woraus zusammen $d_H(v) \geq 2^{k-1} - 1$ folgt. Mithin enthält H nach Induktionsannahme einen K_k-Minor. Da jeder Knoten von H einen Nachbarn in Z besitzt und Z zusammenhängt, enthält also G einen K_{k+1}-Minor. □

Tatsächlich konnten KOSTOCHKA [Kos84] und (unabhängig) THOMASON [Tho84c] zeigen, daß

$$\psi(k) \;=\; \mathcal{O}(k\sqrt{\log k}),$$

d.h. für jedes $\epsilon > 0$ und k hinreichend groß gilt:

$$\chi(G) \geq k^{1+\epsilon} \;\Rightarrow\; G \succ K_k.$$

Aus der Betrachtung zufälliger Graphen folgt zudem, daß diese Schranke bis auf eine Konstante bestmöglich ist [Kos84, Veg83]. Ein etwaiger Beweis der HADWIGER-Vermutung muß demnach mehr Struktur chromatischer Graphen ausnutzen als lediglich die Existenz dichter Subgraphen. BOLLOBÁS, CATLIN und ERDŐS [BCE80] bewiesen immerhin, daß die HADWIGER-Vermutung immerhin für *fast alle* Graphen richtig ist.

Anmerkungen und Übungen

Im Lichte des Satzes von KURATOWSKI klingt die folgende Verschärfung der Vermutung von HADWIGER ebenfalls plausibel:

Vermutung (HAJÓS 1961)
Sei G ein Graph. Für alle $k \in \mathbb{N}$ ist die folgende Aussage richtig
(U_k) : *Ist $\chi(G) \geq k$, so enthält G eine Unterteilung des K_k.*

(U_1), (U_2) und (U_3) sind wieder trivial, (U_4) ist wegen Übung 6.1.30 äquivalent zu Satz 6.3.21. Es gilt ebenfalls $(U_{k+1}) \Rightarrow (U_k)$, und auch (U_5) impliziert den 4-Farben-Satz. ERDŐS und FAJTLOWICZ [EF81] zeigten aber, daß die HAJÓS-Vermutung im Gegensatz zur HADWIGER-Vermutung *fast immer falsch* ist!

Übung 6.3.27 [Cat79] *Man zeige, daß der Graph $C_5[K_3]$ (das ist ein C_5, bei dem jeder Knoten durch ein Dreieck ersetzt wurde und benachbarte Dreiecke vollständig bipartit verbunden werden) ein Gegenbeispiel zur Vermutung von* HAJÓS *darstellt.*

Übung 6.3.28 [Mad67] *Sei F ein Graph mit k Kanten und ohne isolierte Knoten. Dann enthält jeder Graph G vom Durchschnittsgrad $d(G) \geq 2^k$ eine Unterteilung von F.*

Die Bedingung aus dem letzten Satz konnte kürzlich auf $d(G) \geq Ck^2$ für eine Konstante $C \in \mathbb{R}$ verbessert werden, was bis auf eine Konstante bestmöglich ist, siehe [KS96a].

7.3.5 Nirgends verschwindende Flüsse

Eine 3-Färbung der Kanten eines kubischen, planaren Graphen mit den Elementen der KLEINschen Vierergruppe $\mathbb{Z}_2 \times \mathbb{Z}_2$ wie im Beweis zu Satz 6.3.7 kann man auch als einen verallgemeinerten Fluß auffassen, denn in jedem Knoten addieren sich die Werte auf den drei inzidierenden Kanten zu Null. Hierbei kann man sogar jede beliebige Orientierung der Kanten zugrundelegen, denn in $\mathbb{Z}_2 \times \mathbb{Z}_2$ ist jedes Element sein eigenes Inverses.
Wie TUTTE 1954 bemerkte, kann man bei ebenen Graphen generell alle mit Knotenfärbungen verknüpften Probleme via geometrischer Dualisierung als Flußprobleme formulieren - ganz analog der Dualität zwischen Potentialdifferenzen (bzw. Spannung) und Strom in elektrischen Netzwerken.

Definition 6.3.29 *Sei H eine endliche, Abelsche Gruppe mit additiver Notation und $G = (V, E)$ ein Graph mit einer Orientierung $\omega : E \to V$ der Kanten. Ein* H-Fluß *(auch Zirkulation) in G bezüglich ω ist eine Abbildung $f : E \to H$, die das erste* KIRCHHOFF*sche Gesetz erfüllt:*

$$\sum_{e \in \partial^+(v)} f(e) \quad = \quad \sum_{e \in \partial^-(v)} f(e) \qquad \forall v \in V.$$

Ein nirgends verschwindender H-Fluß *ist ein H-Fluß f mit $\forall e \in E : f(e) \neq 0$.*

Sind ω und ω' zwei verschiedene Orientierungen von G, so entsprechen sich die H-Flüsse in G bez. ω bzw. ω' offenbar bijektiv (man ersetze ggf. $f(e)$ durch $-f(e)$), so daß die Existenz eines H-Flusses in G unabhängig von der Orientierung von G ist - also eine Eigenschaft des ungerichteten Graphen. Grundlegend für die Untersuchung nirgends-verschwindender Flüsse ist das folgende Ergebnis von TUTTE:[9]

[9] Ein eleganter Beweis ergibt sich aus der TUTTE-GROTHENDIECK-Theorie, siehe [BO91]

Lemma 6.3.30 (TUTTE 1954a) *Sei G ein Graph und H eine Abelsche Gruppe. Die Anzahl der verschiedenen nicht-verschwindenden H-Flüsse in G (und damit die Existenz eines solchen) hängt nur von der Ordnung $|H|$ der Gruppe ab!* □

Wir sagen daher, G ist ein *F_k-Graph* oder G besitzt einen *k-Fluß* für ein $k \geq 2$ genau dann, wenn G einen *nirgends verschwindenden* H-Fluß für *irgendeine* Abelsche Gruppe der Ordung $|H| = k$ besitzt.

Ein F_k-Graph ist auch ein F_l-Graph für alle $l \geq k$ (TUTTE). Hat ein Graph eine Brücke, so kann er offenbar für kein $k \in \mathbb{N}$ einen k-Fluß besitzen, und ein Graph ist ein F_2-Graph genau dann, wenn er Eulersch ist. F_3-Graphen wurden (wie bei 3-färbbaren Graphen) bisher noch nicht charakterisiert.

Nach obigem Lemma besitzt ein Graph einen 4-Fluß genau dann, wenn er einen nirgends-verschwindenden $\mathbb{Z}_2 \times \mathbb{Z}_2$-Fluß besitzt. Damit besagt Satz 6.3.7 von TAIT gerade, daß ein kubischer, planarer Graph genau dann 4-Flächen-färbbar ist, wenn er einen 4-Fluß besitzt.

Korollar 6.3.31 *Die 4-Farben-Vermutung ist äquivalent dazu, daß jeder brückenlose, kubische, planare Graph einen 4-Fluß besitzt.* □

TUTTE fand heraus, daß eine solche Dualität zwischen Färbungs- und Flußproblemen für planare Graphen allgemein richtig ist:

Satz 6.3.32 (TUTTE 1954a) *Ein (brückenloser) ebener Graph besitzt einen k-Fluß ($k \geq 2$) genau dann, wenn seine Gebiete k-färbbar sind.*

Beweis (siehe auch [BO91, Corollary 6.3.5]).
„$\Leftarrow$“: Sei Δ eine k-Färbung der Gebiete eines ebenen Graphen G. Orientiere die Kanten von G (und die Ebene). Für jede Kante e definiere $f(e) := \Delta(F_r) - \Delta(F_l)$, wobei F_r und F_l das links bzw. rechts an e angrenzende Gebiet sei. Dann überprüft man leicht, daß f ein nirgends verschwindender $\mathbb{Z}_k$-Fluß ist.
„$\Rightarrow$“: Sei f umgekehrt ein nirgends verschwindender $\mathbb{Z}_k$-Fluß auf dem ebenen Graphen $G = (V, E, R)$ bezüglich einer Orientierung ω der Kanten von G. O.B.d.A. sei G zusammenhängend und daher nach Voraussetzung zweifach kantenzusammenhängend. Dann ist für jedes Gebiet $r \in R$ der Einbettung von G der Gebietsrand ∂r ein Zykel, der „im Uhrzeigersinn“ orientiert einen $\mathbb{Z}_k$-Fluß z_r in G bezüglich ω definiert durch

$$z_r(a) := \begin{cases} 1 & \text{falls } a \in \partial r \text{ und im Uhrzeigersinn orientiert,} \\ -1 & \text{falls } a \in \partial r \text{ und gegen den Uhrzeigersinn orientiert,} \\ 0 & \text{falls } a \in E \setminus \partial r. \end{cases}$$

Nach dem Satz von EULER gilt $|R| - 1 = m - (n-1)$. Wenn man also Schritt für Schritt jeweils ein inneres Gebiet r von G mit dem äußeren durch Herausnahme einer Kante $e_r \in \partial r$ vereinigt, verbleibt nach $|R| - 1$ Schritten ein spannender Baum T von G mit $n - 1$ Kanten. Seien $r_1, \ldots, r_{|R|-1}$ die inneren Gebiete von G in der Reihenfolge, wie sie mit dem äußeren vereinigt werden. Definiere $\mathbb{Z}_k$-Flüsse f^i, $0 \leq i \leq |R| - 1$, durch $f^0 := f$ und (beachte: die Linearkombination von Flüssen ist wieder ein Fluß)

$$f^i := f^{i-1} - f^{i-1}(e_{r_i}) \cdot z_{r_i}$$

(dies ist jeweils punktweise für alle $a \in E$ zu verstehen). Für alle $1 \leq i \leq j \leq |R| - 1$ gilt dann: $f^j(r_i) = 0$, d.h. $f^{|R|-1}$ verschwindet auf allen entfernten Kanten e_{r_i}. Da $f^{|R|-1}$

jedoch ein Fluß ist und die übrigen Kanten von G einen Baum bilden, verschwindet $f^{|R|-1}$ sogar schon ganz (beachte: ein maximaler Subgraph von T mit Flußwerten $\neq 0$ enthielte ein Blatt). Durch Auflösen der Rekursion erhält man daher

$$f \;=\; \sum_{i=1}^{|R|-1} \phi(r_i) \cdot z_{r_i},$$

wobei $\phi(r_i) := f^{i-1}(e_{r_i})$ gesetzt wurde. Diese Funktion $\phi : R \to \mathbb{Z}_k$, durch 0 auf das äußere Gebiet fortgesetzt, stellt nun (modulo k genommen) eine zulässige k-Färbung der Gebiete von G dar, denn eine Kante e ist in den Gebietsrändern von genau zwei Gebieten r_ℓ und r_r enthalten; da e im linken Gebietsrand im Gegenuhrzeigersinn und im rechten im Uhrzeigersinn orientiert ist, gilt mithin $0 \neq f(e) = \phi(r_r) - \phi(r_\ell)$. □

Ein $\mathbb{Z}_k$-Fluß läßt sich stets als Summe von $\mathbb{Z}_k$-Flüssen auf den Zykeln einer Zykelbasis des Zykelraums darstellen. Die Planarität geht hier also via das MacLanesche Planaritätskriterium (Übung 6.1.56) ein.

Für $k = 2$ erhalten wir übrigens die Aussage von Übung 6.3.11 zurück.

In der Form von Korollar 6.3.31 läßt sich die 4-Farben-Vermutung nun leicht auf allgemeine Graphen verallgemeinern - in der Hoffnung, den 4-Farben-Satz vielleicht von dieser höheren Warte aus kombinatorisch bzw. gruppentheoretisch beweisen zu können. Zunächst zeigt die Kleinsche Vierergruppe $\mathbb{Z}_2 \times \mathbb{Z}_2$, wie sich der Satz 6.3.7 von Tait auf allgemeine Graphen verallgemeinert:

Lemma 6.3.33 (Tutte 1950)
Ein kubischer Graph besitzt genau dann einen 4-Fluß, wenn er 3-Kanten-färbbar ist. □

Da der Petersen-Graph (s. Seite 88) nicht 3-Kanten-färbbar ist, sind brückenlose (kubische) Graphen also i.a. zwar keine F_4-Graphen. Tutte vermutete aber, daß der Petersen-Graph im wesentlichen schon der einzige solche Graph ist:

4-Fluß-Vermutung (Tutte 1966)
Ein brückenloser Graph ohne den Petersen*-Graph als Minor ist ein F_4-Graph.*

Da der Petersen-Graph nicht planar ist, implizierte die Richtigkeit dieser Vermutung (selbst schon für kubische Graphen) den 4-Farben-Satz - eine Herausforderung für alle arbeitslos gewordenen (oder mit dem Beweis des 4-Farben-Satzes unzufriedenen) Tetrachromatologen.

Die Hoffnung, die 4-Fluß-Vermutung beweisen zu können, stützt sich dabei u.a. auf die Erfahrung, daß sich eine Reihe von Resultaten über Flüsse in planaren Graphen auf allgemeine Graphen verallgemeinern läßt, wie wir es bereits in Lemma 6.3.33 beobachten konnten. Proposition 6.3.13 verallgemeinert sich beispielsweise wie folgt:

Proposition 6.3.34 *Ein Graph $G = (V, E)$ besitzt einen 4-Fluß genau dann, wenn E die Vereinigung von zwei Zykeln ist.*

Beweis. Seien $Z_1, Z_2 \subseteq E$. Für $e \in E$ definiere $\phi(e) := (\phi_1(e), \phi_2(e))$ mit

$$\phi_i(e) := \begin{cases} 1 & \text{falls } e \in Z_i, \\ 0 & \text{sonst.} \end{cases}$$

Dann ist ϕ ein 4-Fluß genau dann, wenn die Z_i Zykeln sind mit $E \subseteq Z_1 \cup Z_2$. □

Ein anderes Beispiel ist die folgende Charakterisierung:

Proposition 6.3.35 (MINTY 1967)
Ein kubischer Graph besitzt genau dann einen 3-Fluß, wenn er bipartit ist.

Beweis. Ist G bipartit mit Teilen U und V, so genügt es, alle Kanten von U nach V zu richten und mit 1 zu gewichten, um einen $\mathbb{Z}_3$-Fluß zu erhalten. Ist umgekehrt ein nirgends verschwindender $\mathbb{Z}_3$-Fluß f gegeben, so führt man, beginnend in einem beliebigen Knoten v, eine Breadth-First-Search durch und zeigt induktiv, daß man eine Orientierung auf G so wählen kann, daß die Sphären um v abwechselnd jeweils nur aus Quellen bzw. Senken bestehen. □

Eingeschränkt auf planare Graphen ergibt sich daraus nämlich via Satz 6.3.32 das

Korollar 6.3.36 (KEMPE 1879; HEAWOOD 1898)
Eine Triangulation ist 3-färbbar genau dann, wenn sie Eulersch ist. □

TUTTE suchte auch nach einer Verallgemeinerung des (vergleichsweise einfachen!) HEAWOODschen 5-Farben-Satzes für allgemeine Graphen.

5-Fluß-Vermutung (TUTTE 1954)
Ein brückenloser Graph besitzt einen 5-Fluß.

(Man überlege sich dies für den PETERSEN-Graph.) In diesem Zusammenhang bewies JAEGER, daß jeder brückenlose Graph ein F_8-Graph ist, und SEYMOUR noch stärker, daß jeder brückenlose Graph sogar schon ein F_6-Graph ist - im Unterschied zu Färbungsproblemen, wo in natürlicher Weise Graphen beliebig hoher chromatischer Zahl auftreten.

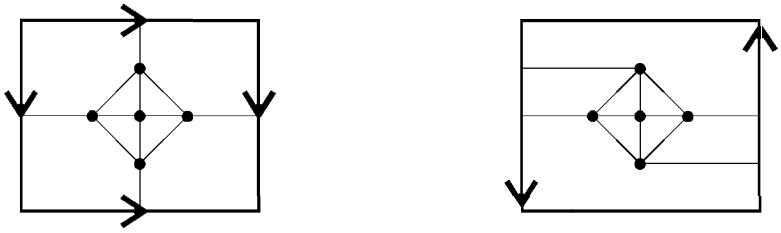
Der Dreifarbensatz 6.3.4 von GRÖTZSCH schließlich besagt via Dualität und Satz 6.3.32, daß jeder 4-kantenzusammenhängende, planare Graph einen 3-Fluß besitzt. Entsprechend lautet hier die

3-Fluß-Vermutung (TUTTE 1966)
Jeder 4-kantenzusammenhängende Graph besitzt einen 3-Fluß.

Hier konnte JAEGER immerhin die Existenz eines 4-Flusses zeigen, vgl. [Jae88].

6.4 ⋆ Graphen höheren Geschlechts

Trivialerweise ist jeder Graph überschneidungsfrei in den $\mathbb{R}^3$ einbettbar. Ist ein Graph nicht planar, also nicht auf die Kugeloberfläche einbettbar, so kann man sich aber fragen, auf welche andere Flächen man ihn (überschneidungsfrei) einbetten kann. So ist beispielsweise der K_5 leicht in die Oberfläche eines Torus oder eines Möbiusbandes einzubetten (gegenüber liegende Seiten mit Pfeil sind entlang der Pfeilrichtung zu indentifizieren):



Wir wollen uns in diesem Abschnitt mit der Einbettbarkeit eines Graphen in beliebige geschlossene Flächen beschäftigen.

Definition 6.4.1 *Eine* geschlossene Fläche *ist eine kompakte 2-dimensionale Mannigfaltigkeit, d.h. ein kompakter, zusammenhängender* HAUSDORFF*-Raum, in dem jeder Punkt eine Umgebung besitzt, die homöomorph zur offenen Kreisscheibe ist.*

Die geschlossenen Flächen zerfallen in zwei Klassen. Eine geschlossene Fläche S heißt *orientierbar*, falls für jede geschlossene Jordankurve C auf S eine Rotationsrichtung erhalten bleibt, wenn man einmal um C herumgeht, andernfalls heißt S *nicht-orientierbar*.
Beispiele für orientierbare, geschlossene Flächen sind die Kugeloberfläche, die Oberfläche eines Torus und allgemein eine Kugel mit einer Anzahl aufgesetzter *Henkel*. Einen Henkel aufsetzen heißt dabei, zwei disjunkte Kreisscheiben aus der Fläche herauszuschneiden und die entstandenen Ränder so zu identifizieren, daß eine Orientierung in Uhrzeigersinn auf dem einen Kreisrand zu einer im Gegenuhrzeigersinn auf dem anderen korrespondiert. Dies erreicht man beispielsweise, indem man die beiden Kreisränder durch einen „Schlauch“ miteinander verbindet. Eine Kugel mit h aufgesetzten Henkeln wird als S_h notiert.
Die nicht-orientierbaren, geschlossenen Flächen sind anschaulich weniger leicht vorstellbar. Ein Beispiel einer nicht-orientierbaren (aber nicht geschlossenen) Fläche ist das obige Möbiusband. Wenn man einen Pfeil (oder einen orientierten Kreis) einmal um das Möbiusband herumführt, wird die Pfeilrichtung (bzw. die Orientierung) umgekehrt. Eine zudem geschlossene, nicht orientierbare Fläche erhält man beispielsweise, indem man in die Kugeloberfläche ein kreisrundes Loch schneidet und je zwei gegenüberliegende Punkte auf dem Rand des Loches identifiziert. Damit erhält man eine Kugel mit einer sogenannten *Kreuzhaube*. Allgemein notiert man eine Kugel mit k aufgesetzten Kreuzhauben als N_k. N_1 werden wir als Kreisscheibe zeichnen, bei der diametral gegenüberliegende Punkte auf dem Kreisrand zu identifizieren sind.
Der Klassifikationssatz für geschlossene Flächen besagt, daß wir uns bei unseren Untersuchungen auf die Flächen S_h und N_k beschränken können.

Satz 6.4.2 (DEHN, HEEGAARD 1907)
Jede geschlossene Fläche ist homöomorph zur S_h oder N_k für ein $h \geq 0$ oder $k > 0$. □

Für einen topologischen Beweis siehe [ST80]. Wie beim Jordanschen Kurvensatz gibt es aber auch hier einen eleganten graphentheoretischen Zugang, siehe [Tho92, Vin93].

Es erhebt sich nun die Frage, wieviele Henkel bzw. Kreuzhauben notwendig und hinreichend sind, um einen Graphen in eine S_h oder N_k einbetten zu können.

Definition 6.4.3 *Das (orientierbare)* Geschlecht $\gamma(G)$ *eines Graphen G ist die kleinste Zahl h, so daß G in die S_h einbettbar ist. Sein* nicht-orientierbare Geschlecht $\bar{\gamma}(G)$ *ist die kleinste Zahl k, so daß er in die N_k einbettbar ist.*

Planare Graphen haben also Geschlecht 0. Führt man jede Kante eines Graphen $G = (V, E)$ über einen eigenen Henkel, ersieht man sofort $\gamma(G) \leq |E|$. Proposition 6.4.9 wird eine ähnliche Schranke auch für das nicht-orientierbare Geschlecht liefern. Diese Graphenparameter sind also wohldefiniert.

Übung 6.4.4 *Der K_5, der $K_{3,3}$ und der* PETERSEN*-Graph haben orientierbares und nicht-orientierbares Geschlecht 1.*

Die Eulersche Formel hat uns gezeigt, daß der Wert $|V| - |E| + |R|$ für jede Einbettung eines planaren Graphen in die Ebene konstant ist. Dies gilt allgemein für geschlossene Flächen.

Satz 6.4.5 (EULER-POINCARÉsche Formel) *Es sei $G = (V, E, R)$ ein zusammenhängender Graph, der in eine geschlossene Fläche F eingebettet ist. Dann gilt:*

$$|V| - |E| + |R| \;\geq\; \begin{cases} 2 - 2h & \text{falls } F \doteq S_h \ (h \geq 0), \\ 2 - k & \text{falls } F \doteq N_k \ (k \geq 1) \end{cases} \tag{6.2}$$

mit Gleichheit, falls jedes Gebiet der Einbettung homöomorph zu einer offenen Kreisscheibe ist (sogenannte 2-Zellen-Einbettung*). Der Wert der rechten Seite von (6.2) heißt die* Euler-Charakteristik $\epsilon(F)$ *der Fläche F.*

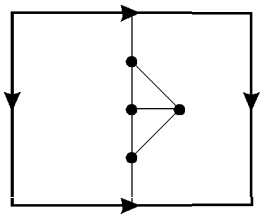
Beweis. im orientierbaren Fall (der nichtorientierbare läuft analog). Wir beweisen den Satz allgemeiner für alle Multigraphen und induzieren nach h.
Für $h = 0$ und einfache Graphen ist (6.2) gerade die Eulersche Formel, denn eine Einbettung eines planaren Graphen in die Ebene ist eine 2-Zellen-Einbettung genau dann, wenn er zusammenhängend ist. Für einen ebenen Multigraphen G gilt (6.2) zunächst für den G zugrundeliegenden einfachen Graphen G', den man erhält, wenn man alle Schleifen und bei parallelen Kanten alle bis auf eine einzige Kante eliminiert. Fügt man die eliminierten Kanten einzeln wieder zu G' hinzu, erhöht sich auch die Anzahl der Gebiete jeweils um eins.
$h \to h + 1$: Es sei $G = (V, E, R)$ in die S_{h+1} 2-Zellen-eingebettet. Ziehe um einen beliebigen der $h + 1 > 0$ Henkel eine Kurve C so, daß C keinen Knoten von G enthält. Dann schneidet C Kanten von G, denn sonst läge C in einem Gebiet von G, ließe sich aber nicht auf einen Punkt zusammenziehen, d.h. dieses Gebiet wäre nicht homöomorph zu offenen Kreisscheibe. Ohne Einschränkung sei die Anzahl k aller Schnittpunkte endlich. Wann immer C eine Kante von G trifft, sei der Kreuzungspunkt ein neuer Knoten. Die durch diese neuen Knoten definierten Abschnitte auf C und den gekreuzten Kanten von G seien neue Kanten (hier entstehen möglicherweise Schleifen oder Mehrfachkanten). Schneide nun den Henkel entlang C auf. C ist dann Rand beider zurückgebliebener Teile des Henkels. Diese

verschließt man jeweils mit einer Kappe und erhält wieder eine geschlossene Fläche, aber vom Geschlecht h. Der auf dieser Fläche eingebettete, auf diese Weise aus G hervorgegangene Graph heiße $G' = (V', E', R')$.
Offenbar enthält G' gerade die $2k$ Knoten auf den beiden Kreisen mehr als G. Vor dem Aufschneiden unterteilte jeder der k Knoten auf C die geschnittene Kante von G einmal. Dadurch kamen also k Kanten hinzu. Weiter enthält G' zusätzlich die $2k$ Kanten auf den beiden aus C hervorgegangenen Kreisen. Somit gilt $|E'| = |E| + 3k$. Vor dem Aufschneiden teilte jede Kante des Kreises C das Gebiet von G, in dem sie lag, in zwei Gebiete. Da das Gebiet vorher homöomorph zur offenen Kreisscheibe war, sind es auch die beiden neuen. Schließlich kamen nach dem Aufschneiden noch die beiden Verschlußkappen mit Rand C hinzu. Also folgt $|R'| = |R| + k + 2$, und auch G' ist 2-Zellen-eingebettet. Mithin ist G' zusammenhängend, und laut Induktionsvoraussetzung gilt

$$\begin{array}{rcrcl} |V'| - |E'| + |R'| & = & (|V| + 2k) - (|E| + 3k) + (|R| + k + 2) & = & 2 - 2h, \\ \Rightarrow & & |V| - |E| + |R| & = & 2 - 2(h+1). \end{array}$$

□

Für jede 2-Zellen-Einbettung eines Graphen in eine geschlossene Fläche hängt also der Wert $|V| - |E| + |R|$ und damit die Anzahl der Gebiete der Einbettung nur von (der Euler-Charakteristik) der Fläche ab.
Beispiel. Die folgende Einbettung des K_4 in den Torus zeigt, daß in der EULER-POINCARÉschen Formel (6.2) strikte Ungleichheit gelten kann, wenn nicht jedes Gebiet homöomorph zur offenen Kreisscheibe ist.



Das „äußere“ Gebiet enthält hier geschlossene Kurven, die sich nicht auf einen Punkt zusammenziehen lassen. Es gilt:

$$|V| - |E| + |R| = 4 - 6 + 3 = 1 > 0 = 2 - 2 \cdot 1 = \epsilon(S_1).$$ □

Übung 6.4.6 *Keine Einbettung eines unzusammenhängenden Graphen ist eine 2-Zellen-Einbettung.*

Mit Hilfe des folgenden topologischen Ergebnisses

Satz 6.4.7 (YOUNGS 1963) *Eine Einbettung eines zusammenhängenden Graphen G auf einer Fläche vom Geschlecht $\gamma(G)$ ist stets eine 2-Zellen-Einbettung.* □

erhalten wir aus der EULER-POINCARÉschen Formel das folgende nützliche

Korollar 6.4.8 *Sei $G = (V, E, R)$ ein auf einer Fläche vom Geschlecht $\gamma(G)$ eingebetteter, zusammenhängender Graph. Dann gilt*

$$|V| - |E| + |R| = 2 - 2\gamma(G).$$ □

In welcher Beziehung stehen orientierbares und nicht-orientierbares Geschlecht eines Graphen?

Proposition 6.4.9 *Für jeden Graphen G gilt*

$$\bar{\gamma}(G) \;\leq\; 2\gamma(G)+1.$$

Beweis. Schneidet man aus zwei Flächen F_1 und F_2 jeweils eine Kreisscheibe heraus und identifiziert die entstandenen Ränder beider Flächen, erhält man eine Fläche $F_1 \circ F_2$, für deren Euler-Charakteristik gilt:

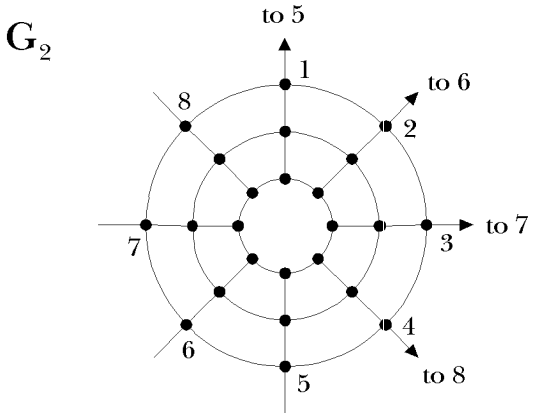
$$\epsilon(F_1 \circ F_2) \;=\; \epsilon(F_1)+\epsilon(F_2)-2. \tag{6.3}$$

Um dies einzusehen, wähle man auf beiden Flächen 2-Zellen-eingebettete, triangulierte Graphen G_1 und G_2, identifiziere ein Dreieck auf F_1 mit einem auf F_2 und berechne $|V(G)|$, $|E(G)|$ und $|R(G)|$ des neu entstandenen Graphen G. (6.3) folgt dann sofort aus (6.2).

Um nun die Proposition zu beweisen, bette man G auf die $S_{\gamma(G)}$ ein und füge in eines der Gebiete von G eine Kreuzhaube N_1 ein. Die entstandene Fläche F ist nicht orientierbar und hat nach (6.3) die Euler-Charakteristik

$$\epsilon(F) \quad = \quad 2-2\gamma(G)\;+\;1\;-\;2 \quad = \quad 2\;-\;(2\gamma(G)+1). \qquad \square$$

Wie wir in Satz 6.4.13 unten sehen werden, ist diese Ungleichung beispielsweise für den K_7 scharf. Daß sich umgekehrt das orientierbare Geschlecht nicht durch das nicht-orientierbare beschränken läßt, zeigten AUSLANDER et al. [ABY63] anhand der Graphen $\{G_n\}_{n\in I\!N}$:



G_n besteht aus $n+1$ konzentrischen Kreisen der Länge $4n$, wobei gegenüberliegende Knoten auf dem äußersten Kreis durch eine Kante verbunden sind. Offensichtlich gilt $\bar{\gamma}(G_n) \equiv 1$ und $\gamma(G_n) \leq n$. Den Beweis von $\gamma(G_n) \geq n$ findet man z.B. in [GT87, S.141].

In Verallgemeinerung der Situation in der Ebene genügt es, das Geschlecht von zweifach zusammenhängenden Graphen zu studieren.

Satz 6.4.10 (BATTLE, HARARY, KODAMA, YOUNGS 1962) *Für einen Graphen G mit Blöcken $B_1, \ldots, B_k$ gilt:*

$$\gamma(G) = \sum_{i=1}^{k} \gamma(B_i). \qquad \square$$

(Wobei die Ungleichung „$\leq$“ trivial ist.)

Ein Wald ist planar. Überlegen wir uns also eine untere Schranke für das Geschlecht eines Graphen in Abhängigkeit von seiner Taillenweite.

Lemma 6.4.11 *Für einen zusammenhängenden Graphen G der Ordnung ≥ 3 mit Taillenweite g gilt*

$$\gamma(G) \;\geq\; \frac{g-2}{2g}m - \frac{1}{2}n + 1.$$

Beweis. Sei G auf der Fläche $S_{\gamma(G)}$ (2-Zellen-) eingebettet mit Gebieten R. Da jedes Gebiet von mindestens g Kanten berandet wird, gilt $g|R| \leq 2|E|$. Aus Korollar 6.4.8 folgt mithin

$$|V| - |E| + \frac{2}{g}|E| \;\geq\; 2 - 2\gamma(G)$$

und daraus die Behauptung. □

Untersuchen wir damit das Geschlecht spezieller Graphenklassen. Setzt man $|V| = n$, $|E| = \frac{1}{2}(n^2 - n)$ und $g = 3$, findet man:

Lemma 6.4.12 *Für $n \geq 3$ gilt:*

(i) $\gamma(K_n) \geq \left\lceil \frac{(n-3)(n-4)}{12} \right\rceil$,

(ii) $\bar{\gamma}(K_n) \geq \left\lceil \frac{(n-3)(n-4)}{6} \right\rceil$. □

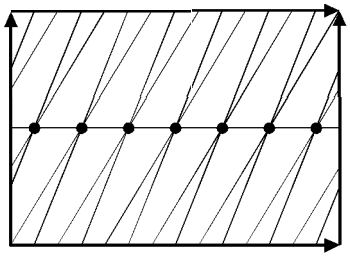
Wie man schon 1890 vermutete, gilt hier tatsächlich sogar (fast immer) Gleichheit. Der nichtorientierbare Fall stellte sich als der leichtere heraus - er wurde 1954 schließlich (im wesentlichen) von RINGEL gelöst. Den bemerkenswerten Beweis für das orientierbare Geschlecht konnten RINGEL und YOUNGS erst 1968 (!) nach langer Vorarbeit in Einzelfällen durch etliche andere Wissenschaftler abschließen. Er findet sich in bzw. ist Gegenstand des Buches [Rin74].

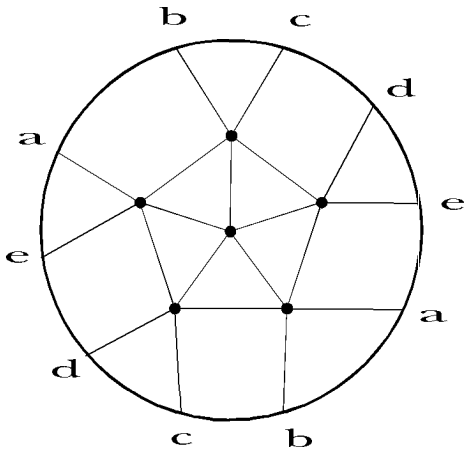
Satz 6.4.13 (RINGEL 1954; RINGEL, YOUNGS 1968) *Für $n \geq 3$ gilt:*

(i) $\bar{\gamma}(K_n) = \left\lceil \frac{(n-3)(n-4)}{6} \right\rceil$ *außer* $\bar{\gamma}(K_7) = 3$,

(ii) $\gamma(K_n) = \left\lceil \frac{(n-3)(n-4)}{12} \right\rceil$. □

Der größte auf dem Torus einbettbare, vollständige Graph ist also der K_7, der größte in der projektiven Ebene der K_6:





Man beachte, daß die Einbettung des K_7 den Torus trianguliert und das „Dual“ der Einbettung des K_6 in die Fläche N_1 gerade der PETERSEN-Graph ist. Dies stellt eine seltsame Bijektion zwischen den Kanten dieser Graphen her.

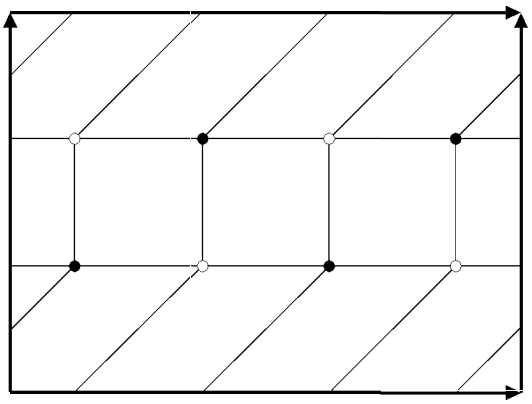
Auch für die Klasse der vollständig bipartiten Graphen ist die untere Schranke für das Geschlecht aus Lemma 6.4.11 scharf:

Satz 6.4.14 (RINGEL 1965) *Für $m, n \geq 2$ gilt:*

i) $\bar{\gamma}(K_{m,n}) = \left\lceil \frac{(m-2)(n-2)}{2} \right\rceil$,

ii) $\gamma(K_{m,n}) = \left\lceil \frac{(m-2)(n-2)}{4} \right\rceil$. □

Es gibt also bereits zweifärbbare Graphen beliebig hohen Geschlechts! Der größte auf dem Torus einbettbare, vollständig bipartite Graph ist demnach der $K_{4,4}$:



Übung 6.4.15 (RINGEL 1955; BEINEKE, HARARY 1965) *Das Geschlecht des d-dimensionalen Würfel-Graphen Q_d, $d \geq 2$, (vgl. Proposition 1.1.16) ist $\gamma(Q_d) = (d-4)2^{d-3}+1$.* [*Hinweis:* Man finde induktiv eine Einbettung, in der die 2^{d-2} disjunkten Kreise

$$(0, x_2, \ldots, x_{d-1}, 0),\ (0, x_2, \ldots, x_{d-1}, 1),$$
$$(1, x_2, \ldots, x_{d-1}, 1),\ (1, x_2, \ldots, x_{d-1}, 0),$$

$x_i \in \{0, 1\}$, Gebietsränder sind.]

Übung 6.4.16 Für einen Graphen der Ordnung ≥ 3 gilt: $m \leq 3n + 6(\gamma(G) - 1)$.

Die Komplexität des Einbettungsproblems. Während man also das Geschlecht spezieller Graphenfamilien bestimmen konnte, klassifiziert der nächste Satz das allgemeine Problem als NP-schwer.

Satz 6.4.17 (THOMASSEN 1989) *Das Entscheidungsproblem* GRAPH GENUS*:*

INSTANZ: ein Graph $G = (V, E)$ und ein $h \in \mathbb{N}_0$,
FRAGE: $\gamma(G) \leq h$*?*

ist NP-vollständig. □

Für festes h gibt es aber polynomielle Algorithmen. Da das Geschlecht eines jeden Graphen nach Satz 6.4.13 $\mathcal{O}(n^2)$ ist, müssen diese (falls P $\neq$ NP) aber eine in h exponentielle Laufzeit haben.

Satz 6.4.18 (FILOTTI, MILLER, REIF 1979) *Sei $h \in \mathbb{N}_0$ fest. Dann gibt es einen Algorithmus, der zu einem gegebenen Graphen $G = (V, E)$ in Zeit $\mathcal{O}(|V|^{\mathcal{O}(h)})$ entscheidet, ob $\gamma(G) \leq h$, und der, falls dies der Fall ist, G in die Fläche S_h einbettet.* □

Wie für planaren Graphen gibt es auch für Flächen höheren Geschlechts eine Charakterisierung der auf ihnen einbettbaren Graphen durch verbotene Subgraphen. So zeigte VOLLMERHAUS 1968, daß es für jedes $\gamma \geq 0$ eine endliche Familie $\mathcal{F}_\gamma$ von Graphen gibt, so daß $\gamma(G) \leq \gamma$ genau dann, wenn G keinen Subgraph homöomorph zu einem Element aus $\mathcal{F}_\gamma$ enthält. Eine Charakterisierung über Minoren folgt aus der Richtigkeit einer Vermutung von WAGNER aus dem Jahre 1937, die ROBERTSON und SEYMOUR im Rahmen ihres „graph minors project"s bewiesen, siehe [RS85a, RS85b, RS90, RS95].

Satz 6.4.19 (ROBERTSON, SEYMOUR) *Sei $\mathcal{K}$ eine bezüglich Minorbildung abgeschlossene Graphenklasse, d.h. es gilt:*

$$G \in \mathcal{K} \ \wedge \ G \succ H \ \Rightarrow \ H \in \mathcal{K}.$$

Dann gibt es eine endliche Menge $Obstr(\mathcal{K})$ von Graphen, so daß für alle Graphen G gilt:

$$G \in \mathcal{K} \quad \Leftrightarrow \quad G \not\succ H \quad \forall H \in Obstr(\mathcal{K}).$$ □

Dieses Ergebnis zählt zu den tiefliegendsten und bedeutendsten Errungenschaften der Graphentheorie. Weiter konnten sie zeigen:

Satz 6.4.20 (ROBERTSON, SEYMOUR 1995) *Sei H ein fester Graph. Dann besitzt das Entscheidungsproblem* MINOR CONTAINMENT *(H):*

INSTANZ: ein Graph $G = (V, E)$,
FRAGE: $G \succ H$?

einen $\mathcal{O}(n^3)$-Algorithmus. □

Da die Klasse aller auf eine Fläche S_h einbettbaren Graphen bezüglich Minorenbildung abgeschlossen ist, ergibt sich aus den obigen beiden Sätzen das

Korollar 6.4.21 *Für eine geschlossene Fläche S_h, $h \geq 0$, besitzt das Problem* GRAPH EMBEDDING*:*

INSTANZ: ein Graph $G = (V, E)$,
FRAGE: ist G auf S_h einbettbar?

einen $\mathcal{O}(n^3)$-Algorithmus. □

Man beachte jedoch, daß zum einen die Konstante des $\mathcal{O}(|V|^3)$-Algorithmus aus Satz 6.4.20 astronomisch (in $|H|$) ist und man zum anderen die Hindernismenge $Obstr(S_h)$ der minorminimalen, nicht auf die Fläche S_h einbettbaren Graphen nicht kennt. Der Beweis des Satzes 6.4.19 ist hochgradig nicht-konstruktiv und die Aussage des Korollars daher eine reine Existenzaussage.

Ein konstruktiver $\mathcal{O}(n^{10})$-Algorithmus für dieses Problem, der allerdings den ROBERTSON-SEYMOUR-Algorithmus für das Entscheidungsproblem benutzt, findet sich in [Arch90]. Zur Approximation von GRAPH GENUS siehe [CKK93].

Die chromatische Zahl von Flächen höheren Geschlechts. HEAWOOD ging auch der zur 4-Farben-Vermutung analogen Fragestellung für Flächen höheren Geschlechts nach.

Definition 6.4.22 *Die chromatische Zahl $\chi(S_\gamma)$ ($\chi(N_\gamma)$) einer orientierbaren (nicht orientierbaren) geschlossenen Fläche S_γ (N_γ) vom Geschlecht γ ist*

$$\begin{aligned}\chi(S_\gamma) &:= \max\{\chi(G) : G \text{ auf } S_\gamma \text{ einbettbar}\},\\ \chi(N_\gamma) &:= \max\{\chi(G) : G \text{ auf } N_\gamma \text{ einbettbar}\}.\end{aligned}$$

Die 4-Farben-Vermutung besagt also gerade $\chi(S_0) = 4$ (und nicht 5).
HEAWOOD zeigte die Wohldefiniertheit von $\chi(S_\gamma)$ bzw. $\chi(N_\gamma)$ auch für $\gamma > 0$:

Satz 6.4.23 (Map Color Theorem) (HEAWOOD 1890) *Sei $\gamma > 0$.*

$$\chi(S_\gamma) \leq \left\lfloor \frac{7+\sqrt{1+48\gamma}}{2} \right\rfloor =: H(\gamma) \tag{6.4}$$

$$\chi(N_\gamma) \leq \left\lfloor \frac{7+\sqrt{1+24\gamma}}{2} \right\rfloor =: \overline{H}(\gamma) \tag{6.5}$$

Beweis. Wir zeigen nur den orientierbaren Fall, der nicht-orientierbare läuft analog. Sei $G = (V, E)$ ein auf S_γ eingebetteter Graph mit $\chi(G) = k$. Ohne Einschränkung sei G kritisch k-chromatisch, d.h.

$$\chi(G - v) = k - 1 \qquad \forall v \in V$$

(sonst entferne Knoten). Dann gilt für die Knotengrade

$$d(v) \geq k - 1 \qquad \forall v \in V,$$

denn sonst wäre eine $(k-1)$-Färbung von $G - v$ auf v fortsetzbar. Es folgt

$$2|E| \geq |V|(k-1).$$

Außerdem ist G zusammenhängend, und wir dürfen $|V| =: n \geq 3$ annehmen. Aus der EULER-POINCARÉschen Formel (6.2):

$$|V| - |E| + |R| \geq 2 - 2\gamma$$

ergibt sich daher wegen $3|R| \leq 2|E|$

$$|E| \leq 3|V| + 6(\gamma - 1).$$

Setzt man beide Ungleichungen zusammen, erhält man also

$$\begin{aligned}&\tfrac{n(k-1)}{2} \leq |E| \leq 3n + 6(\gamma-1)\\ \Rightarrow\quad &(k-7)\,n \leq 12(\gamma-1).\end{aligned}$$

Für $\gamma = 0$ ergibt sich daraus die wertlose Ungleichung $k \leq 7 - \frac{12}{n}$ oder $k \leq 6$ (wir wissen aus Satz 6.3.1 bereits $k \leq 5$!). Ist aber $k \geq 7 \Rightarrow \gamma > 0$, so sind beide Seiten der Ungleichung nichtnegativ, und wir können wegen $k \leq n$ folgern

$$\begin{aligned}&(k-7)\,k \leq 12(\gamma-1)\\ \Rightarrow\quad &k \leq \tfrac{7}{2} + \sqrt{\tfrac{49}{4} + 12(\gamma-1)}\\ \Rightarrow\quad &k \leq \tfrac{7+\sqrt{1+48\gamma}}{2}\end{aligned}$$

□

Bemerkung.

- Für $\gamma = 0$ ergäbe sich in (6.4) der 4-Farben-Satz, nur leider funktioniert der Beweis in diesem Fall nicht.
- Wie anhand des $K_{m,n}$ gesehen (s. Satz 6.4.14), gibt es umgekehrt schon 2-färbbare Graphen beliebig hohen Geschlechts.

HEAWOOD glaubte sogar, Gleichheit in (6.4) gezeigt zu haben. Für $\gamma = 1$ folgt dies wegen der Einbettbarkeit des K_7 auf dem Torus, s. Abbildung Seite 208. HEFFTER wies 1891 auf die Unvollständigkeit in HEAWOODs Beweisführung hin und steuerte die Gleichheit in einigen weiteren Fällen bei. DIRAC zeigte, daß man sich allgemein auf die Untersuchung der vollständigen Graphen beschränken konnte, da jeder Graph G auf S_γ mit maximaler chromatischer Zahl $\chi(G) = \chi(S_\gamma)$ schon einen $K_{\chi(S_\gamma)}$ enthält, so daß Gleichheit in Satz 6.4.23 genau dann gilt, wenn $K_{H(\gamma)}$ auf der Fläche S_γ einbettbar ist. Daher löste erst die vollständige Bestimmung des Geschlechts des vollständigen Graphen das Problem. Hierin liegt die eigentliche Bedeutung des Satzes 6.4.13 von RINGEL und YOUNGS!

Korollar 6.4.24 *Für $\gamma > 0$ gilt Gleichheit in (6.4):*

$$\begin{aligned} \chi(S_\gamma) &= H(\gamma) \\ \chi(N_\gamma) &= \overline{H}(\gamma) \end{aligned}$$

außer im Fall der KLEIN*schen Flasche N_2, wo $\chi(N_2) = 6$.*

Beweis (wieder nur für den orientierbaren Fall, der nicht-orientierbare läuft außer im Fall der KLEINschen Flasche N_2 analog).
Aus Satz 6.4.13 folgt für den größten auf S_γ einbettbaren K_n:

$$\begin{array}{rccccccccc} & \left\lceil \frac{(n-3)(n-4)}{12} \right\rceil & = \gamma(K_n) & \leq & \gamma & < & \gamma+1 & \leq & \gamma(K_{n+1}) & = & \left\lceil \frac{(n-2)(n-3)}{12} \right\rceil \\ \Rightarrow & & \frac{n^2-7n+12}{12} & \leq & \gamma & < & \frac{n^2-5n+6}{12}. & & & & \end{array}$$

Durch Auflösen nach n ergibt sich hieraus

$$\begin{array}{rccccc} & \frac{5+\sqrt{1+48\gamma}}{2} & < & n & \leq & \frac{7+\sqrt{1+48\gamma}}{2} \\ \Rightarrow & & & n & = & \left\lfloor \frac{7+\sqrt{1+48\gamma}}{2} \right\rfloor \end{array}$$

und mithin $\chi(S_\gamma) \geq \chi(K_n) = n = H(\gamma)$. □

Interessanterweise konnte also das Problem, $\chi(S_\gamma)$ zu bestimmen, für $\gamma > 0$ gelöst werden, während das Problem in der Ebene nach wie vor offen blieb.

6.5 Der 4-Farben-Satz und Komplexitätsfragen

Die chromatische Zahl von Flächen höheren Geschlechts war bestimmt, MAYER [May75] konnte die BIRKHOFF-Zahl, die angibt, wieviele Knoten ein Gegenbeispiel zur 4-Farben-Vermutung mindestens haben muß, auf 96 hochschrauben, aber ein Beweis der 4-Farben-Vermutung war nicht in Sicht.

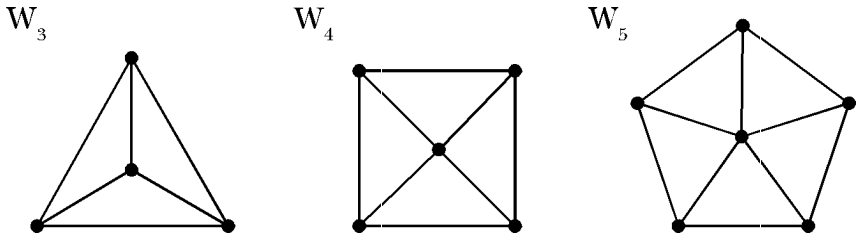
Ein einfaches Abzählargument schließt zumindest eindeutig färbbare, 5-chromatische planare Graphen aus.

Übung 6.5.1 (CHARTRAND, GELLER 1969)
a) *Jeder eindeutig 4-färbbare, planare Graph ist maximal planar.*
b) *Kein planarer Graph ist eindeutig 5-färbbar.*

Bewiesen wurde die 4-Farben-Vermutung kurioserweise schließlich mit Methoden, die KEMPES ursprüngliche Beweisidee aufgreifen. Zum Beweis wird die Annahme, es gäbe einen planaren Graphen, der nicht 4-färbbar und somit notwendig 5-chromatisch ist, zum Widerspruch geführt. Unter dieser Widerspruchsannahme gibt es offenbar auch einen knotenminimalen und maximal planaren, 5-chromatischen Graphen. Eine solches Gegenbeispiel zur 4-Farben-Vermutung, eine 5-chromatische Triangulation der Ebene mit minimal vielen Knoten, wollen wir einen *irreduziblen* Graphen nennen.

Übung 6.5.2 *Ein knotenminimaler, 5-chromatischer planarer Graph ist eine Triangulation.*

Zum Beweis der 4-Farben-Vermutung genügt es dann, zu zeigen, daß irreduzible Graphen nicht existieren. Um die gebräuchliche Notation einzuführen, vergegenwärtigen wir uns noch einmal die Beweisstruktur des 5-Farben-Satzes 6.3.1. Aus Korollar 6.1.14 wissen wir, daß jede ebene Triangulation G einen Knoten vom Grad höchstens 5 besitzt, d.h. G enthält mindestens eine der folgenden Konfigurationen:



Dabei verstehen wir unter einer *Konfiguration* in einer Triangulation einen trennenden Kreis C mit nichtleerem „Inneren“ (das hier nur aus einem Knoten besteht). Die Länge des Kreises heißt die *Ringgröße* der Konfiguration. Eine Menge von Konfigurationen (wie die aus obiger Abbildung), von denen mindestens eine in jeder Triangulation der Ebene auftreten muß, nennen wir *unvermeidbar*.
Die Beweisidee im Fall des 5-Farben-Satzes ist es nun, nachzuweisen, daß alle Konfigurationen der obigen unvermeidbaren Menge *reduzierbar* sind, d.h. in einem knotenminimalen Gegenbeispiel zum 5-Farben-Satz nicht auftreten können - folglich gibt es keine Gegenbeispiele. Warum sind die obigen 3 Konfigurationen reduzierbar? Nun, nehmen wir an, eine knotenminimale, 6-chromatische Triangulation besäße eine dieser Konfigurationen. Entfernen wir jeweils den Knoten v im Inneren des Kreises, wissen wir aufgrund der Knotenminimalität, daß der Restgraph 5-färbbar ist. In den ersten beiden Fällen läßt sich diese

5-Färbung unmittelbar auf den Knoten v fortsetzen. Im letzten Fall zeigte ein KEMPE-Ketten-Argument, daß es, falls die 5-Färbung des Restgraphen tatsächlich alle 5 Farben auf dem Ring benutzt, auch eine 5-Färbung des Restgraphen gibt, die nur 4 Farben auf dem Ring benutzt, so daß auch hier wieder eine Farbe für den Knoten v übrigbleibt.

KEMPEs Ansatz war, zu zeigen, daß die Graphen W_3, W_4, W_5 aus der obigen unvermeidbaren Menge reduzierbar sind, d.h. in einem irreduziblen Graphen nicht auftreten können. W_3 ist natürlich reduzierbar, für W_4 folgt dies aus einem KEMPE-Ketten-Argument wie im Beweis des 5-Farben-Satzes. Das bedeutet:

Korollar 6.5.3 *Ein irreduzibler Graph ist vom minimalen Grad $\delta \geq 5$.* □

Der Beweis der 4-Farben-Satzes wäre nun also komplett, könnte man auch die Reduzierbarkeit von W_5 nachweisen, denn dies stände ja im Widerspruch zur Unvermeidbarkeit der Menge $\{W_3, W_4, W_5\}$. Beim Nachweis der Reduzierbarkeit von W_5 irrte KEMPE.

Um 1935 griff HEESCH jedoch diese Idee wieder auf, um auch die 4-Farben-Vermutung zu beweisen, und sie führte letztendlich auch zum Ziel [AH77, AHK77]:

Satz 6.5.4 (APPEL, HAKEN, KOCH 1977) *Jeder planare Graph ist 4-färbbar.* □

Die drei Autoren fanden eine unvermeidbare Menge von 1818 reduzierbaren Konfigurationen, alle Ringgröße höchstens 14. Die Anzahl konnte man seither auf 1258 drücken. Es ist unmittelbar einsichtig, warum für diesen Beweisansatz der Einsatz eines Computers notwendig ist. Da bis heute auch noch kein „eleganterer" Beweis gefunden wurde, der ohne diese gigantische Fallunterscheidung auskommt, mutmaßen die Autoren, daß dieser Satz womöglich von einer Art ist, die ein solches Vorgehen und damit den Einsatz des Computers unumgänglich macht. Nur leider kann ein solcher Beweis kein Verständnis dafür vermitteln, was der „tiefere Grund" für die Richtigkeit einer Aussage ist.

ROBERTSON, SANDERS, SEYMOUR und THOMAS [RSST94] bestätigten kürzlich diesen Beweisansatz durch einen neuen Computerbeweis des 4-Farben-Satzes 6.5.4, der nur 633 Konfigurationen benutzt und zudem auch konzeptionell einfacher ist.

Zum Nachweis der Reduzierbarkeit einer Konfiguration kann man offenbar sämtliche 4-Färbungen des zugrundeliegenden irreduziblen Graphen G zu Hilfe nehmen, bei dem man die Konfiguration durch eine beliebige andere Konfiguration derselben Ringgröße aber auf weniger Knoten ersetzt hat. Eine einfache Anwendung dieser Beobachtung ist

Proposition 6.5.5 (BIRKHOFF 1913)
Konfigurationen mit Ringgröße höchstens 4 (gleich was ihr Inneres!) sind reduzierbar.

Beweis. Sei $C = (v_1, \ldots, v_l)$, $l \geq 3$, der trennende Kreis einer solchen Konfiguration in einem irreduziblen Graphen G. Bezeichne I bzw. A das Innere von C bzw. das Äußere von C jeweils vereinigt mit C. I und A enthalten beide mindestens einen Knoten weniger als G, sind also 4-färbbar.
Im Fall $l = 3$ sind natürlich alle Knoten v_i, $i = 1, 2, 3$, verschieden gefärbt, und durch Permutation der Farben setzt man die 4-Färbungen von I und A zu einer 4-Färbung von G zusammen, die Konfiguration tritt daher in einem irreduziblen Graphen nicht auf und ist also reduzierbar.
Im Fall $l = 4$ ist man ebenso fertig, falls I und A 4-Färbungen besitzen, die auf C alle vier Farben benutzen. Ohne Einschränkung besitze also I nur 4-Färbungen, die auf C höchstens 3 Farben benutzen (durch Einfügen von Diagonalen auf C erkennt man, daß

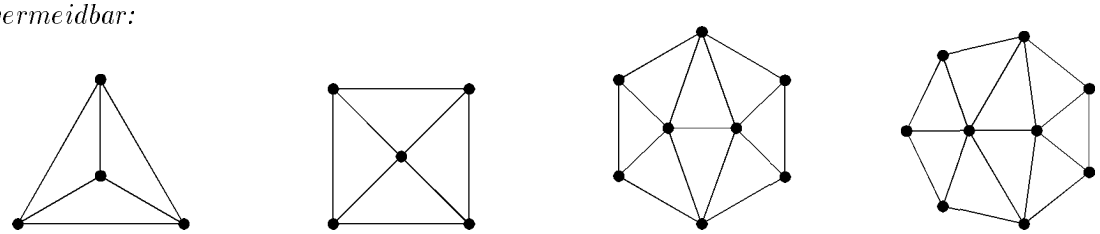
man die Benutzung von 3 Farben auf C erzwingen kann). Sei Δ nun eine 4-Färbung von $A + \{v_1, v_3\}$. Benutzt Δ nur drei Farben auf C, kann man sie offenbar mit einer 4-Färbung von $I + \{v_1, v_3\}$ zusammensetzen. Benutzt Δ aber 4 Farben auf C, so läßt sich daraus mit Hilfe eines KEMPE-Ketten-Arguments wie beim 5-Farben-Satz immer auch eine 4-Färbung Δ' von A gewinnen, die nur 3 Farben auf C verwendet. Diese kann man nun je nachdem mit einer 4-Färbung von $I + \{v_1, v_3\}$ bzw. $I + \{v_2, v_4\}$ zusammensetzen. □

Aus dem Beweis zu Lemma 6.3.16 wissen wir, daß jede minimal trennende Menge in einer Triangulation ein Kreis ist. Folglich erhalten wir aus obiger Proposition das

Korollar 6.5.6 *Ein irreduzibler Graph ist 5-zusammenhängend.* □

Ein wichtiges Hilfsmittel zum Nachweis der Unvermeidbarkeit einer Menge von Konfigurationen ist die auf HEESCH [Hee69] zurückgehende Methode des Entladens, die wir am folgenden Beispiel illustrieren wollen. Es ersetzt den W_5 durch zwei andere Konfigurationen.

Proposition 6.5.7 (WERNICKE 1904) *Die folgende Menge von Konfigurationen ist unvermeidbar:*



Beweis. Sei also im Widerspruch zur Behauptung $G = (V, E)$ eine irreduzible Triangulation ohne Knoten vom Grad ≤ 4 und ein Knoten vom Grad 5 sei nicht zu einem anderen Knoten vom Grad 5 oder 6 benachbart. Lade jeden Knoten $v \in V$ entsprechend seinem Grad mit der Ladung $l(v) := 6 - d(v)$. Die gesamte verteilte Ladung beläuft sich daher auf

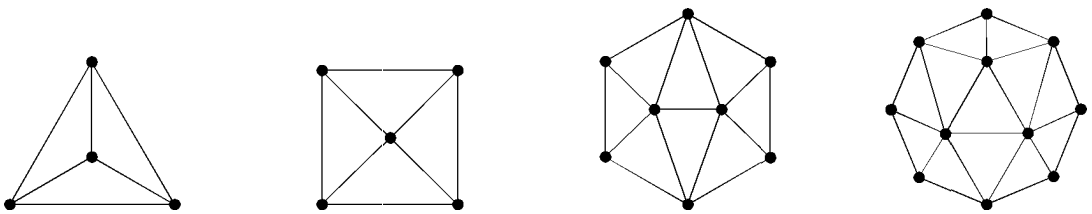
$$\sum_{v \in V} \{6 - d(v)\} \;=\; 6|V| - 2|E| \;=\; 12,$$

da für eine Triangulation $|E| = 3|V| - 6$. Entlade nun jeden positiv geladenen Knoten, d.h. jeden Knoten vom Grad 5, durch gleichmäßige Verteilung seiner Ladung auf seine Nachbarn. Nach Annahme tragen alle Knoten vom Grad 5 oder 6 jetzt die Ladung 0. Da G eine Triangulation ist, darf ein Knoten v vom Grad $d(v) \geq 7$ in G nach Annahme nur höchstens $\frac{d(v)}{2}$ viele Nachbarn vom Grad 5 haben und ist daher noch immer negativ geladen:

$$d(v) \;>\; \frac{20}{3} \quad \Rightarrow \quad 6 - d(v) + \frac{d(v)}{2} \cdot \frac{1}{5} \;<\; 0.$$

Die Gesamtladung wäre nach dem Umverteilen also höchstens Null, Widerspruch. □

Übung 6.5.8 (APPEL, HAKEN) *Die folgende Menge ist unvermeidbar:*



[*Hinweis:* ein Knoten vom Grad 5 gebe nun die Ladung 1/3 an alle seine Nachbarn vom Grad mindestens 7 ab.]

Während der Nachweis der Unvermeidbarkeit einer Menge von Konfigurationen oder der Reduzierbarkeit einer speziellen Konfiguration (unter bestimmten Methoden) kein prinzipielles Problem darstellt, aber gerade bei größerer Ringgröße übermäßig zeitaufwendig sein kann, besteht die Leistung von APPEL, HAKEN und KOCH im wesentlichen in der feinen Abstimmung des Zusammenspiels der beiden widerstrebenden Probleme. Hier kamen insbesondere wahrscheinlichkeitstheoretische Argumente zum Zuge.

Die Komplexität des planaren Färbungsproblems

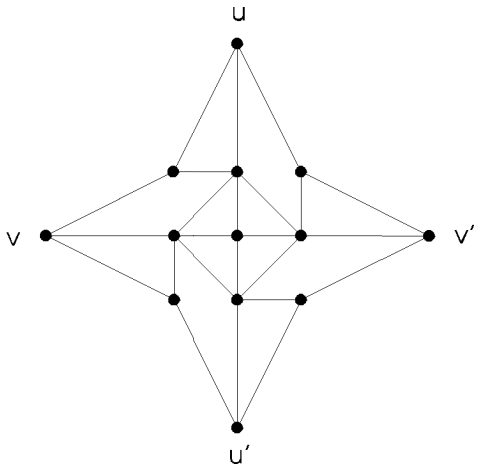
Wir wissen (oder glauben) nun, daß jeder planare Graph 4-färbbar ist. Aus den oben erwähnten Beweisen des 4-Farben-Satzes ergeben sich sogar ein $\mathcal{O}(n^4)$-Algorithmus [AH89] bzw. ein $\mathcal{O}(n^2)$-Algorithmus [RSST96] (allerdings mit großen Konstanten), um eine entsprechende Färbung zu konstruieren. Ob ein Graph zweifärbbar (bipartit) ist, läßt sich (leicht) in linearer Zeit feststellen. Ist ein planarer Graph jedoch nicht zweifärbbar, so ist das Problem, zu entscheiden, ob seine chromatische Zahl 3 oder 4 ist, schon NP-vollständig, denn es gilt:

Satz 6.5.9 (STOCKMEYER 1973) *Das Problem* PLANAR 3-COLORABILITY*:*

INSTANZ: ein planarer Graph G,
FRAGE: $\chi(G) \leq 3$*?*

ist NP*-vollständig.*

Beweis. Eine Einbettung und eine 3-Färbung sind ein Zertifikat für ein Element der Sprache der planaren, 3-färbbaren Graphen, das sicher in polynomieller Zeit überprüfbar ist. Daher ist PLANAR 3-COLORABILITY $\in$ NP.
Zum Nachweis der NP-Vollständigkeit reduzieren wir vom NP-vollständigen Entscheidungsproblem 3-COLORABILITY. Sei also eine Instanz von 3-COLORABILITY, d.h. ein Graph $G = (V, E)$, gegeben. Betrachte den „Baustein“ H:



Wie man sich leicht überzeugt, hat der Baustein H folgende Eigenschaften:

1. In jeder 3-Färbung von H tragen jeweils u und u' sowie v und v' dieselbe Farbe.

2. Für alle $(i,j) \in \{1,2,3\} \times \{1,2,3\}$ gibt es eine 3-Färbung Δ von H mit $\Delta(u) = \Delta(u') = i$ und $\Delta(v) = \Delta(v') = j$.

Wir konstruieren nun in polynomieller Zeit aus G einen planaren Graphen G' wie folgt.

- Zeichne G so in die Ebene, daß Kanten (d.h. die zugehörigen Jordankurven) nur ihre Endpunkte mit V gemeinsam haben, sich nicht selbst schneiden, andere Kanten höchstens einmal schneiden und sich in einem Punkt, der kein Knoten ist, höchstens zwei Kanten schneiden (Übung).

- Füge auf jeder Kante $e = \{x,y\} \in E$, die von $k(e)$ anderen Kanten gekreuzt wird, $k(e)+1$ neue Knoten (vom Grad 2) zwischen den Überkreuzungen und zwischen x bzw. y und der benachbarten Überkreuzung ein. Die letzteren beiden Knoten nenne x' bzw. y'. Wegen (6.6) werden dabei insgesamt höchstens polynomiell viele neue Knoten eingefügt.

- Man ersetze nun jede Überkreuzung zweier Kanten $\{u,u'\}$ und $\{v,v'\}$ durch den Baustein H, indem man die vier Anschlüsse von H mit den der Überkreuzung benachbarten, neu eingefügten vier Knoten identifiziert. Wieder werden nur polynomiell viele Knoten eingefügt.

- Für jede Kante $e = \{x,y\} \in E$ kontrahiere schließlich beliebig eine der Kanten $\{x,x'\}$ oder $\{y,y'\}$.

Dann gilt: $$G \textit{ 3-färbbar} \Leftrightarrow G' \textit{ 3-färbbar},$$
denn einerseits kann jede 3-Färbung von G wegen (2.) auf G' fortgesetzt werden, färbt man jeden auf einer Kante $\{x,y\} \in E$ eingefügten Knoten wie den, der durch Kontraktion entstand, und umgekehrt induziert jede 3-Färbung Δ von G' eine solche von G, denn aus $\Delta(x) = \Delta(y)$ für zwei in G benachbarte Knoten x, y folgte, falls x die kontrahierte Kante, wegen (1.) auch schon $\Delta(y') = \Delta(y)$, Widerspruch. Die Reduktion ist offensichtlich in polynomieller Zeit berechenbar. □

Tatsächlich bleibt auch das Problem PLANAR 3-COLORABILITY eingeschränkt auf Graphen vom Maximalgrad 4 NP-vollständig:

Korollar 6.5.10 [GJS76] PLANAR 3-COLORABILITY ($\Delta \leq 4$) *ist* NP*-vollständig.*

Beweis. Die Bausteine H_k aus dem Beweis von Satz 5.4.3 sind planar. □

Trotzdem lassen sich 3-färbbare, planare Graphen hübsch charakterisieren:

Übung 6.5.11 (HEAWOOD 1898) *Ein planarer Graph ist 3-färbbar genau dann, wenn er Subgraph einer Eulerschen Triangulation ist.*
[*Hinweis:* Korollar 6.3.36. Zu „⇒“: betrachte zunächst 2-zusammenhängende Graphen]

6.6 ⋆ Kreuzungszahl, Dicke, Arborizität und Buchdicke

Kreuzungszahl. Auch nicht planare Graphen kann man in die Ebene zeichnen - allerdings nur mit „Überkreuzungen". Wir kommen dabei überein, daß

- die eine Kante repräsentierende Jordankurve nur ihre beiden Endpunkte mit der Knotenmenge gemeinsam hat;
- sich in einem Punkt, der kein Knoten ist, höchstens zwei (verschiedene) Kanten scheiden;
- sich zwei Kanten höchstens einmal kreuzen und
- sich inzidierende Kanten nicht schneiden.

Da sich eine beliebige Zeichnung eines Graphen in die Ebene stets ohne zusätliche Kreuzungen in eine Zeichnung, die den obigen vier Bedingungen genügt, umwandeln läßt (Übung), betrachten wir im folgenden o.B.d.A. nur noch derartige Zeichnungen. Das minimale $k \in \mathbb{N}_0$, so daß sich ein Graph G mit k Überkreuzungen in die Ebene zeichnen läßt, heißt die *Kreuzungszahl* $cr(G)$ von G.

Wenn man die Knoten eines vollständigen Graphen auf einem Kreis anordnet und die Kanten als Geradenstücke einzeichnet, so entsprechen sich Kreuzungen und vierelementige Teilmengen der Knotenmenge bijektiv; also gilt

$$cr(G) \;\leq\; cr(K_n) \;\leq\; \binom{n}{4}. \tag{6.6}$$

Da man jede Überkreuzung durch einen Henkel oder eine Kreuzhaube eliminieren kann, stellt die Kreuzungszahl eine obere Schranke für das orientierbare wie das nichtorientierbare Geschlecht dar. I.a. können Kreuzungszahl und Geschlecht jedoch beliebig weit auseinanderfallen:

Übung 6.6.1 a) *Der* Petersen*-Graph hat Kreuzungszahl 2.*
b) *Man konstruiere Graphen G_k, $k \in \mathbb{N}$, für die zwar $cr(G_k) \geq k$, die aber eine Kante $e_k \in E(G_k)$ enthalten, so daß $G_k - e_k$ planar ist.*

Das Problem, für einen Graphen eine Zeichnung in die Ebene mit minimal vielen Überkreuzungen zu finden, ist von Bedeutung bei Verdrahtungs- und Layout-Problemen im VLSI-Design (siehe [Lei83]) sowie bei der automatisierten Erstellung von „ästhetischen" und übersichtlichen Diagrammen von Graphen (siehe [BETT94]); die Berechnung der Kreuzungszahl eines Graphen ist jedoch NP-schwer [GJ83]. Die Kreuzungszahl konnte bisher nur für wenige Graphen bestimmt werden.

Übung 6.6.2 *Man finde eine Einbettung des K_6 in die Ebene mit 3 Überkreuzungen und zeige, daß dies bestmöglich ist.*

Allgemein kennt man für die Kreuzungszahl des vollständigen Graphen K_n lediglich die folgende Verbesserung der trivialen Schranke (6.6):

Satz 6.6.3 (Guy 1960; Blažek, Koman 1964) *Für die Kreuzungszahl des vollständigen Graphen gilt:*

$$cr(K_n) \;\leq\; \frac{1}{4}\left\lfloor\frac{n}{2}\right\rfloor\left\lfloor\frac{n-1}{2}\right\rfloor\left\lfloor\frac{n-2}{2}\right\rfloor\left\lfloor\frac{n-3}{2}\right\rfloor. \tag{6.7}$$

Beweis. Für gerades n zeichnen wir den K_n wie folgt auf einen Zylinder. Sei $n = 2k$ für ein $k \in \mathbb{N}$. Auf den Kreisrand eines jeden der beiden „Deckel“ des Zylinders setzen wir je k Knoten äquidistant. Die Knoten auf je einem Kreisrand verbinden wir untereinander durch Diagonalen auf dem zugehörigen Deckel. Da genau jede vierelementige Teilmenge der Knoten auf einem Kreisrand eine Kreuzung induziert, entstehen also auf den Deckeln auf diese Weise genau $2 \cdot \binom{k}{4}$ Kreuzungen. Die Kanten zwischen den Knoten aus verschiedenen Kreisrändern führen wir über geodätische (d.h. lokal kürzeste, hier also Helix-artige) Kurven auf dem Zylindermantel. Seien die Knoten aus den Kreisrändern jeweils mit $0, \ldots, k-1$ durchnumeriert, so daß sich Knoten mit gleicher Nummer „gegenüberliegen“. Gegenüberliegende Knoten verbinden wir durch ein Geradenstück. Dann verbinden wir jeden Knoten i auf dem unteren Deckel mit dem Knoten $i+1$ auf dem oberen durch eine kürzeste Kurve (dabei entstehen noch keine Kreuzungen). In der dadurch definierten „Richtung“ verbinden wir nun auch jeden Knoten i auf dem unteren Deckel mit dem Knoten $i+2$ auf dem oberen durch eine Geodätische; es entstehen genau k Kreuzungen. Wenn man so fortfährt, führt dies zu

$$\sum_{i=0}^{k-1} k \cdot \sum_{j=1}^{i-1} j \;=\; k \cdot \sum_{i=2}^{k-1} k \cdot \sum_{j=1}^{i-1} j \;=\; \frac{k^2(k-1)(k-2)}{6}$$

vielen Kreuzungen auf dem Zylindermantel. Insgesamt enthält diese Zeichnung also wie gewünscht $\frac{k(k-1)^2(k-2)}{4}$ viele Kreuzungen.

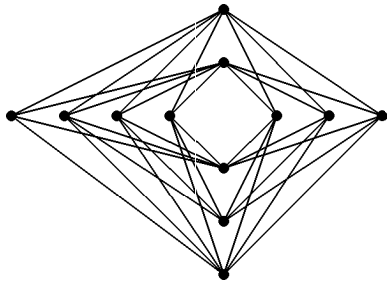
Für ungerades n, $n = 2k-1$, zeichnen wir erst den K_{n+1} wie eben ein und entfernen dann wieder einen Knoten. Auf dem zugehörigen Deckel fallen dadurch $\binom{k-1}{3}$ Kreuzungen fort. Da jede Kreuzung auf dem Zylindermantel je zwei Knoten in jedem Kreisrand betrifft, ist nach obiger Rechnung jeder Knoten an $\frac{k(k-1)(k-2)}{3}$ vielen Kreuzungen auf dem Zylindermantel beteiligt. Auch diese Einbettung hat mithin wie gewünscht

$$\frac{k(k-1)^2(k-2)}{4} \;-\; \binom{k-1}{3} \;-\; \frac{k(k-1)(k-2)}{3} \;=\; \frac{(k-1)^2(k-2)^2}{4}$$

viele Kreuzungen. □

Guy [Guy71, Guy72] zeigte Gleichheit in (6.7) für die Fälle $1 \le n \le 10$. Kleitman [Kle70] konnte als untere Schranke $cr(K_n) \ge (n-3)^4/120$ nachweisen. Man vermutet Gleichheit in (6.7).

Für die Kreuzungszahl der vollständig bipartiten Graphen (ihre Bestimmung wird auch „Turáns brick-factory problem“ genannt) liefert bereits das folgende einfache Schema die beste bekannte obere Schranke. Die Knoten der einen Bipartitionsklasse werden entlang der x-Achse, die der anderen entlang der y-Achse angeordnet, so daß die Knoten auf jeder Achse jeweils möglichst gleichmäßig um den Nullpunkt verteilt sind. Die folgende Abbildung zeigt eine entsprechende Zeichnung des $K_{5,7}$:



Übung 6.6.4 a) [Zar54] *Für die Kreuzungszahl des vollständig bipartiten Graphen gilt:*

$$cr(K_{r,s}) \leq \left\lfloor \frac{r}{2} \right\rfloor \left\lfloor \frac{r-1}{2} \right\rfloor \left\lfloor \frac{s}{2} \right\rfloor \left\lfloor \frac{s-1}{2} \right\rfloor . \tag{6.8}$$

b) $cr(K_{4,4}) = 4$.

KLEITMAN [Kle70] konnte Gleichheit in (6.8) zeigen für $1 \leq \min\{r,s\} \leq 6$.

Im Gegensatz zur Situation bei den planaren Graphen unterscheidet sich die sogenannte gradlinige Kreuzungszahl $\overline{cr}(G)$ eines Graphen, bei der die Kanten eines Graphen Gradenstücke sind, von der Kreuzungszahl $cr(G)$. Es gilt beispielsweise $\overline{cr}(K_8) = 19 > 18 = cr(K_8)$ [Guy71, Guy72]. Tatsächlich kann die Differenz zwischen den beiden Graphenparametern beliebig groß werden, siehe [BN92, BN93].

Gute untere Schranken für Kreuzungszahlen gewinnt man durch VLSI-Techniken, siehe [Lei83]. Einen Überblick über Ergebnisse und Probleme zur Kreuzungszahl geben [EG73, SSV95].

Dicke. Eine weitere Maßzahl neben der Kreuzungszahl für die Nichtplanarität eines Graphen G ist die sogenannte *Dicke* $th(G)$ (engl. thickness). Sie bezeichnet die kleinste Anzahl planarer Subgraphen von G, deren (Kanten-)Vereinigung gleich G ist. Offensichtlich gilt

$$th(G) \geq \frac{m}{3n-6}.$$

Angewandt auf die vollständigen Graphen erhalten wir ($n \geq 3$):

$$th(K_n) \geq \left\lceil \frac{n(n-1)}{6(n-2)} \right\rceil = \left\lfloor \frac{n(n-1)+6(n-2)-1}{6(n-2)} \right\rfloor = \left\lfloor \frac{n+7}{6} \right\rfloor . \tag{6.9}$$

Übung 6.6.5 *Die Dicke des K_8 und des $K_{6,6}$ ist zwei.*

Man folgert, daß auch $th(K_n) = 2$, $n = 5,6,7$, und $th(K_9) \leq 3$. Tatsächlich gilt in (6.9) stets Gleichheit bis auf die Ausnahmen $th(K_9) = th(K_{10}) = 3$. Mehr zur Dicke von speziellen Graphenfamilien in [Bei67a, Bei67b, Kle67]. Die Dicke eines Graphen zu berechnen ist NP-schwer [Man83].

Vergleichen wir Dicke und Geschlecht eines Graphen: Wie der $K_{4,s}$ zeigt, kann man das Geschlecht eines Graphen nicht durch eine Funktion der Dicke allein nach oben abschätzen. Es gilt jedoch

Proposition 6.6.6 [DH91] $\gamma(G) \leq (th(G)-1)(n-1)$.

Beweis. Sei $G = (V,E)$ die Vereinigung von planaren Graphen $G^{(1)}, \ldots, G^{(th)}$, jeweils auf der Knotenmenge V. Seien zunächst alle $G^{(i)}$, $i = 1, \ldots, th$, auf jeweils einer separaten Sphäre $S^{(i)}$ eingebettet. Wir entfernen nun die Knoten der Graphen $G^{(i)}$, $i = 2, \ldots, th$. Sei $v \in G^{(i)}$, $i \in \{2, \ldots, th\}$, ein solcher Knoten. Schneide um v eine kleine Kreisscheibe aus $S^{(i)}$ heraus, so daß diese Kreisscheibe nur den Knoten v sowie mit v inzidierende Kanten enthält und jede mit v inzidierende Kante die Kreisscheibe genau einmal verläßt. Ebenso schneiden wir in der Umgebung des Repräsentanten von v auf $S^{(1)}$ eine kleine Kreisscheibe in einem an diesen Knoten anliegenden Gebiet aus, so daß diese Kreisscheibe keine Kante von $G^{(1)}$ ausschneidet. Die beiden Kreisscheiben verbinden wir durch eine Röhre (einen „Schlauch“) $R^{(v,i)}$. Über diesen Schlauch lassen sich dann die mit v in $G^{(i)}$ inzidierenden

Kanten mit dem Repräsentanten von v in $G^{(1)}$ verbinden.
Für einen festen Knoten $v_0 \in V$ vereinigen nun die Röhren $R^{(v_0,i)}$, $i = 2, \ldots, th$, die th Sphären zu einer einzigen (zu $S^{(1)}$ homöomorphen) Sphäre. Die restlichen $(n-1)(th-1)$ Röhren bilden Henkel. □

Wir ahmen einen Beweis aus [DH91] nach, um umgekehrt die Dicke durch das Geschlecht abzuschätzen.

Proposition 6.6.7 [DH91] $$th(G) \leq \left\lceil 4 + \sqrt{2\gamma(G)} \right\rceil .$$

Beweis. O.B.d.A. gilt offenbar $\gamma(G) \geq 1$. Sei $k \in \mathbb{N}$ (wir werden k später bestimmen). Setze $H := G$. $G_1, \ldots, G_k$ seien zunächst leere Graphen auf der Knotenmenge V. Solange es nun in H einen Knoten v vom Grad $0 < d(v) \leq k$ gibt, entferne alle mit v inzidierenden Kanten aus H, verteile sie jeweils einzeln auf die Graphen $G_1, \ldots, G_{d(v)}$ und iteriere.
Wenn schließlich alle Knoten in H Grad 0 oder $\geq k+1$ haben, dann sind die erhaltenen Graphen G_i, $i = 1, \ldots, k$, kreisfrei: denn angenommen, C sei ein Kreis in einem G_j; Betrachte den Knoten c auf C, der zuerst aus H entfernt wurde; Aus der Verfahrensweise, wie die mit c inzidierenden Kanten auf die Graphen G_i verteilt wurden, folgt $d_C(v) \leq 1$ - Widerspruch.
Sei nun H' der von den Knoten vom Grad $\geq k+1$ in H induzierte Subgraph und n' bzw. m' seine Knoten- bzw. Kantenzahl. Mit Übung 6.4.16 gilt

$$n' \cdot (k+1) \leq 2m' \leq 2\left(3n' + 6(\gamma(H') - 1)\right) \leq 6n' + 12(\gamma(G) - 1),$$

woraus $n' \leq \frac{12(\gamma(G)-1)}{k-5}$ folgt. Da $th(K_n) \leq \frac{n+9}{6}$ für alle $n \in \mathbb{N}$ (s.o.), finden wir

$$\begin{aligned} th(H) &= th(H') \leq th(K_{n'}) \\ &\leq \frac{n'+9}{6} \\ &\leq \frac{1}{6}\left(\frac{12(\gamma(G)-1)}{k-5} + 9\right) \\ &= \frac{4\gamma(G) + 3k - 19}{2(k-5)}. \end{aligned}$$

Diese obere Schranke für die Dicke von H strebt für $k \to \infty$ gegen $3/2$. Wenn wir also k nur groß genug wählen, können wir erreichen, daß diese obere Schranke $\leq k$ ist. Löst man die entsprechende quadratische Ungleichung in k, findet man $k \geq 13/4 + \sqrt{\frac{17}{16} + 2\gamma(G)}$. Mithin ist für

$$k := \left\lceil 4 + \sqrt{2\gamma(G)} \right\rceil \overset{(\gamma(G)\geq 1)}{\geq} 13/4 + \sqrt{\frac{17}{16} + 2\gamma(G)}$$

die Dicke von H höchstens k.
Sei $P_1, \ldots, P_{th}$, $th := th(H) \leq k$, eine Zerlegung von $E(H)$ (bzw. von $E(H')$) in planare Graphen. Der Beweis ist vollendet, wenn wir zeigen, daß die Kantenvereinigung der Graphen G_i und P_i, $i = 1, \ldots, th$, jeweils planar ist. Dazu wiederum genügt es, zu zeigen, daß jeder Baum T im Wald G_i nur höchstens einen Knoten aus $V(H')$ enthält: angenommen, T enthielte zwei Knoten x und y aus $V(H')$ und der eindeutige $x-y$–Pfad in T keine weiteren Knoten aus $V(H')$. Der $x-y$–Pfad p hätte dann $l(p)$ Kanten, die alle zu G_i gehörten, aber nur die $l-1$ inneren Knoten wurden „aufgelöst“ und haben jeweils höchstens eine Kante zu G_i beigetragen - Widerspruch. □

Der K_n zeigt, daß die Abschätzungen aus den beiden letzten Propositionen bis auf einen Faktor korrekt sind.

Arborizität. Ein der Dicke sehr ähnlicher Parameter ist die *Arborizität* $a(G)$ eines Graphen (auch Waldzerlegungszahl). Sie bezeichnet die minimale Anzahl kreisfreier Subgraphen von G, deren (Kanten-) Vereinigung G ist. Da ein Wald maximal $n-1$ Kanten hat, benötigt man also mindestens $m/(n-1)$ viele solcher Wälder. Daraus ergibt sich eine untere Schranke für $a(G)$, die erstaunlicherweise schon scharf ist:

Satz 6.6.8 (NASH-WILLIAMS 1964) *Für jeden Graphen G gilt*

$$a(G) \;=\; \max_H \left\lceil \frac{|E(H)|}{|V(H)|-1} \right\rceil, \tag{6.10}$$

wobei H über alle nicht-trivialen, induzierten Subgraphen von G läuft.

Die Arborizität des Graphen K_n beispielsweise berechnet sich mit Hilfe von Satz 6.6.8 unmittelbar zu $a(K_n) = \lceil n/2 \rceil$. In Übung 3.2.1a wurde eine entsprechende Wald- bzw. Baum-Zerlegung konstruiert (für n ungerade füge man dem Graphen zunächst einen Knoten hinzu).

Beweis von Satz 6.6.8:
Den Beweis führt man heute zumeist allgemeiner im Rahmen der Matroid-Theorie, siehe [Wel76, Chapter 8.4]. CHEN et al. [CMWZZ94] gaben jedoch einen kurzen graphentheoretischen Beweis für die Richtung „$\leq$“ in (6.10) durch Induktion über die Anzahl m der Kanten von G. Für Wälder ist die Aussage (6.10) offensichtlich richtig. Sei G also ein Graph mit m Kanten, und (6.10) für alle Graphen mit weniger als m Kanten richtig. Falls G nicht zusammenhängt, so folgt (6.10) sofort aus der Induktionsvoraussetzung. Ebenso, falls $a(G) = a(G-e)$ für eine Kante $e \in E$, da dann

$$a(G) \;=\; a(G-e) \;\overset{(IV)}{=}\; \max_{H \leq G-e} \left\lceil \frac{|E(H)|}{|V(H)|-1} \right\rceil \;\leq\; \max_{H \leq G} \left\lceil \frac{|E(H)|}{|V(H)|-1} \right\rceil .$$

Es genügt also, (6.10) für alle zusammenhängenden, a-kritischen (d.h. $a(G-e) = a(G)-1$ für alle Kanten $e \in E$) Graphen G (wie beispielsweise den K_3) zu zeigen. Dies wiederum leiten wir leicht aus der folgenden Behauptung ab:

Sei G zusammenhängend und a-kritisch mit $n \geq 2$. Dann ist für jede Kante $e \in E$ jede $(a(G)-1)$–Waldzerlegung von $G-e$ eine Zerlegung in $a(G)-1$ spannende Bäume von G.

Denn unter Benutzung dieser Behauptung haben wir

$$m-1 \;=\; |E(G-e)| \;=\; (a(G)-1)(n-1),$$

woraus sofort

$$\begin{array}{rclcl} a(G) & = & \left\lceil a(G) - 1 + \frac{1}{n-1} \right\rceil & = & \left\lceil \frac{(a(G)-1)(n-1)+1}{n-1} \right\rceil \\ & = & \left\lceil \frac{m}{n-1} \right\rceil & \leq & \max_H \left\lceil \frac{|E(H)|}{|V(H)|-1} \right\rceil \end{array}$$

folgt, was zu zeigen war.
Beweis der Behauptung. Angenommen, die Behauptung sei falsch. Sei dann G ein zusammenhängender und a-kritischer Graph, der eine Kante $e = \{u,v\}$ enthält, so daß

$G-e$ eine Waldzerlegung $W_1,\ldots,W_{a-1}$ besitzt, $a := a(G)$, bei der W_1 kein G spannender Baum ist. Wegen $a(G) > a(G-e)$ enthält jedes $W_i + e$, $1 \le i \le a-1$, (genau) einen Kreis. Die Komponente von W_1, die u und v enthält, spannt also nur eine echte Untermenge $U \subset V$ der Knoten. Wir betrachten nun die Menge $\mathcal{Z}$ aller Waldzerlegungen von G der Form $Z' = (W_1',\ldots,W_{a-1}',e')$, wobei eine Komponente von W_1' ein spannender Baum von $G[U]$ ist und $e' \in G[U]$. Wie die Waldzerlegung $(W_1,\ldots,W_{a-1},e)$ zeigt, ist $\mathcal{Z}$ nicht leer. Da G zusammenhängt, ist $E(G) \setminus E(G[U]) \neq \emptyset$. Da G a-kritisch ist, besitzt $G[U]$ also eine $(a-1)$–Waldzerlegung $A_1,\ldots,A_{a-1}$. Mithin gibt es auch ein Element $Z^* = (W_1^*,\ldots,W_{a-1}^*,e^*) \in \mathcal{Z}$, das die folgende Funktion $f : \mathcal{Z} \to \mathbb{N}$ maximiert:

$$f(W_1',\ldots,W_{a-1}',e') \;:=\; \sum_{i=1}^{a-1} |W_i' \cap A_i|.$$

Wir führen dies zum Widerspruch, indem wir ein Element aus $\mathcal{Z}$ konstruieren, daß einen noch größeren f-Wert als Z^* hat. Wie oben enthält wegen $a(G) > a(G-e)$ auch jedes $W_i^* + e^*$ (genau) einen Kreis C_i. Überlegen wir uns, daß alle diese Kreise C_i ganz in $G[U]$ verlaufen. Für $i = 1$ ist dies nach Definition von $\mathcal{Z}$ klar. Angenommen, ein Kreis C_i, $2 \le i \le a-1$, verläßt $G[U]$ und $f \in C_i \cap \langle U, V \setminus U\rangle$ sei eine Kreiskante, die aus $G[U]$ herausführt. Dann wäre $(W_1^* + f,\ldots,W_i + e^* - f,\ldots,W_{a-1}^*)$ eine Waldzerlegung von G im Widerspruch zu $a(G) = a$.
Da schließlich $e^* \in G[U]$, folgt $e^* \in A_i$ für ein $1 \le i \le a-1$. Da A_i kreisfrei ist, gibt es eine Kante $f \in C_i \setminus A_i$. Mithin ist

$$f(W_1^*,\ldots,W_i^* + e^* - f,\ldots,W_{a-1}^*,f) \;>\; f(Z^*). \qquad \square$$

Zusammen mit Korollar 6.1.10 ergibt sich aus diesem Satz, daß planare Graphen Arborizität höchstens drei haben, woraus folgt:

$$th(G) \;\le\; a(G) \;\le\; 3th(G).$$

Die Arborizität läßt sich daher ähnlich gegen das Geschlecht abschätzen wie die Dicke, siehe [Sch87a]. Anders als bei der Dicke gibt es jedoch (wie so oft, wenn Graphenparameter wohlcharakterisiert sind) polynomielle Algorithmen, um eine minimale Waldzerlegung eines Graphen zu berechnen, siehe [Kam76, RT85, GW92].

Buchdicke. Ein p-Buch ist eine Gerade L im $\mathbb{R}^3$ zusammen mit p Halbebenen, die L als gemeinsamen Rand haben. Die *Buchdicke* $bth(G)$ (auch Seitenzahl) eines Graphen G ist das minimale p, so daß G wie folgt (kreuzungsfrei) in ein p-Buch eingebettet werden kann: alle Knoten liegen auf L, und jede Kante von G ist in L oder aber in einer Buchseite enthalten.[10] Offensichtlich gilt $th(G) \le \lceil bth(G)/2 \rceil$. Zusammenhängende Graphen mit kleiner Buchdicke lassen sich leicht charakterisieren:

Proposition 6.6.9 *Für einen zusammenhängenden Graphen G gilt:*

- $bth(G) = 0 \;\Leftrightarrow\; G \cong P_n$;
- $bth(G) \le 1 \;\Leftrightarrow\; G$ *ist kreisartig planar;*

[10] Die Bedingung an den Verlauf der Kanten ist essentiell, da ohne sie jeder Graph bereits in ein 3-Buch einbettbar ist [BK79].

- $bth(G) \leq 2 \Leftrightarrow$ *G ist Subgraph eines Hamiltonschen planaren Graphen.* □

Aus letzterem folgt zusammen mit Satz 6.3.19, daß das Problem, die Buchdicke eines Graphen zu bestimmen, NP-schwer ist [Wig82]. Bereits unter den Graphen mit Buchdicke 3 gibt es Graphen beliebig hohen Geschlechts [BK79]. Die maximale Buchdicke planarer Graphen ist jedoch 4 [Yan86, Yan89]. Allgemein haben Graphen Buchdicke $\mathcal{O}(\sqrt{\gamma(G)})$ [Mal94b]. Für Anwendungen siehe [CLR87, Mal94a].

Übung 6.6.10 [BK79]

a) $\frac{m-n}{n-3} \leq bth(G) \leq \tau(G)$;

b) $bth(K_n) = \lceil n/2 \rceil$;

c) $\bar{d} := \frac{1}{n} \sum_{v \in V} d(v) < 2bth(G) + 2$;

d) $\chi(G) \leq 2bth(G) + 2$.

Kapitel 7

Perfekte Graphen

7.1 Einleitung

Wie wir in Abschnitt 5.3 gesehen haben, können i.a. die chromatische Zahl und die Cliquenzahl in einem Graphen beliebig weit auseinanderfallen. Andererseits ist auch bei Gleichheit der beiden Parameter über den Graphen i.a. nicht viel auszusagen - man füge einem beliebigen Graphen eine $\chi(G)$-Clique hinzu.[1] Das Bild ändert sich jedoch gewaltig, definiert man wie folgt.

Definition 7.1.1 (SHANNON 1956; BERGE 1961) *Ein Graph G heißt* perfekt, *falls für jeden induzierten Subgraphen H von G gilt: $\chi(H) = \omega(H)$.*

Bäume und allgemeiner bipartite Graphen G sind also beispielsweise perfekt, denn für sie gilt $\chi(G) = \omega(G) = 2$ (bzw. $= 1$, falls sie keine Kanten enthalten) und die Perfektheit folgt daraus, daß sich die Eigenschaft eines Graphen, bipartit zu sein, auf alle seine induzierten Subgraphen vererbt (Heredität). Eine schon etwas interessantere Klasse perfekter Graphen liefert die

Proposition 7.1.2 *Komplemente bipartiter Graphen sind perfekt.*

Beweis. Aufgrund der Heredität der Klasse genügt es, zu zeigen, daß für jeden bipartiten Graphen G $\chi(\overline{G}) = \omega(\overline{G})$ oder $\theta(G) = \alpha(G)$ gilt. Da für alle dreiecksfreien Graphen $\theta(G) = \rho(G)$ gilt, ist dies mit Hilfe des Satzes 4.1.9 von GALLAI äquivalent zum Satz 4.2.15 von KÖNIG. □

Übung 7.1.3 *Ungerade Kreise der Länge ≥ 5 und deren Komplemente sind nicht perfekt.*

Übung 7.1.4 a) *Die disjunkte Vereinigung perfekter Graphen ist perfekt.*
b) *Identifikation zweier perfekter Graphen in einer Clique (d.h. Identifikation der Knoten zweier gleichgroßer Cliquen in den beiden einzelnen Graphen) ergibt einen perfekten Graphen.*

Übung 7.1.5 *Die Kanten eines dreiecksfreien Graphen tragen die* HELLY*-Eigenschaft: wenn immer sich eine Menge von Kanten paarweise schneidet, so enthält auch schon der Schnitt über alle diese Kanten einen Knoten.*[2]

[1] vergleiche mit der Situation in Korollar 4.2.36!

[2] HELLY bewies eine analoge Aussage für Intervalle auf der reellen Achse.

Die letzte Übung erweist sich für das folgende Ergebnis als nützlich.

Proposition 7.1.6 *Linegraphen bipartiter Graphen sind perfekt.*

Beweis. Die Klasse ist wieder hereditär. Sei $L = L(G)$ der Linegraph eines bipartiten Graphen G. Aus Übung 7.1.5 entnehmen wir $\omega(L) = \Delta(G)$. Wegen $\chi(L) = \chi'(G)$ ist die Proposition äquivalent zu der Aussage, daß bipartite Graphen „Klasse 1" sind: $\chi'(G) = \Delta(G)$, vgl. Korollar 4.2.14. □

Übung 7.1.7 *Komplemente von Linegraphen bipartiter Graphen sind perfekt.*

Immer wieder von Nutzen ist die folgende einfache Charakterisierung.

Lemma 7.1.8 *Ein Graph G ist perfekt genau dann, wenn in allen induzierten Subgraphen H von G eine stabile Menge $S(H)$ existiert, die alle maximum Cliquen in H schneidet:*

$$\forall H \subseteq G \quad \exists S(H): \qquad \omega(H - S(H)) \leq \omega(H) - 1. \tag{7.1}$$

Beweis. „⇒": Wegen $\chi(H) = \omega(H)$ trifft eine beliebige Farbklasse einer $\chi(H)$-Färbung von H alle maximum Cliquen in H.
„⇐": durch Induktion nach $|H|$.

$$\chi(H) \;\leq\; \chi(H-S)+1 \;\overset{(IV)}{=}\; \omega(H-S)+1 \;\overset{(7.1)}{\leq}\; \omega(H).$$ □

Obwohl die folgende Graphenklasse nach Lemma 7.1.8 trivialerweise perfekt ist, enthält sie doch alle perfekten Graphenklassen, die wir in diesem Kapitel betrachten werden.

Definition 7.1.9 (Berge, Duchet 1984) *Ein Graph G heißt* stark perfekt, *falls jeder Subgraph H von G eine (stabile) Menge $S \subseteq V(H)$ besitzt, die alle maximalen Cliquen von H schneidet.*

Korollar 7.1.10 *Stark perfekte Graphen sind perfekt.*

Optimierung auf perfekten Graphen. Perfekte Graphen sind insbesondere aus algorithmischer Sicht von Interesse. Denn für perfekte Graphen werden wichtige NP-schwere Optimierungsprobleme zugänglich (siehe auch [GLS84, GLS88]):

Satz 7.1.11 (Grötschel, Lovász, Schrijver 1981)
Die mit den Graphenparametern $\chi(G), \alpha(G), \theta(G)$ und $\omega(G)$ verknüpften Suchprobleme sind für perfekte Graphen in polynomieller Zeit lösbar.

Übung 7.1.12
Für perfekte Graphen sind die Optimierungsprobleme $\chi(G), \omega(G), \alpha(G), \theta(G)$ in NP ∩ co-NP.

Die entsprechenden polynomiellen Algorithmen benutzen jedoch die Ellipsoidmethode aus der Linearen Optimierung, deren Darstellung den hier gesteckten Rahmen sprengen würde. Ein kombinatorischer Algorithmus ist bisher nicht bekannt. Für viele Teilklassen der perfekten Graphen kennt man aber effiziente kombinatorische Algorithmen. Wir erinnern z.B. an die

- *bipartiten Graphen:* Das Cliquenproblem ist offenbar trivial, das Färbungsproblem löst der Erkennungsalgorithmus aus Proposition 1.3.5. Mittels eines Maching-Algorithmus findet man ein $\nu(G)$-Matching M in polynomieller Zeit. Daraus konstruiert man wie im Beweis von Satz 4.2.15 von KÖNIG eine minimum Knotenüberdeckung C. Das Komplement $V \setminus C$ ist dann eine maximum stabile Menge. Die Matchingkanten aus M zusammen mit den einzelnen restlichen Knoten, die von M nicht überdeckt werden, stellt wegen $\theta(G) = \rho(G)$ nach dem Satz 4.1.9 von GALLAI eine minimum Cliquenüberdeckung dar.

- *Linegraphen bipartiter Graphen:* Die entsprechenden Optimierungsproblem werden hier sofort zugänglich, wenn man erst den zugrundeliegenden bipartiten Graphen B konstruiert hat, dessen Linegraph der Graph G ist. Dafür wurde in Abschnitt 12.2 ein polynomieller Algorithmus entwickelt. Jede mit einem Knoten $v \in B$ maximalen Grades inzidierende Kantenmenge stellt dann eine maximum Clique in $G = L(B)$ dar. Ein maximum Matching in B ergibt eine maximum stabile Menge in G, und eine minimale Knotenüberdeckung in B definiert eine Überdeckung der Knoten von G durch Cliquen, die man leicht zu einer Cliquenpartition macht. Eine optimale Knotenfärbung schließlich liefert der Kantenfärbungsalgorithmus aus Satz 5.6.4.

In den Abschnitten 7.2 und 7.3 wollen wir die Suchprobleme für $\chi(G), \alpha(G), \theta(G)$ und $\omega(G)$ bei chordalen und bei Vergleichbarkeitsgraphen untersuchen. Auch dort werden uns Strukturaussagen über die Graphenklasse effiziente Algorithmen ermöglichen.

Das Erkennungsproblem. Es ist unbekannt, ob perfekte Graphen in polynomieller Zeit erkannt werden können. Tatsächlich weiß man nicht einmal, ob PERFECT GRAPH RECOGNITION in NP liegt. In Abschnitt 7.6 zeigen wir allerdings, daß PERFECT GRAPH RECOGNITION $\in$ co-NP. Darüberhinaus kennt man wiederum für viele Teilklassen perfekter Graphen (wie z.B. für bipartite Graphen, Linegraphen bipartiter Graphen und deren Komplemente) effiziente Algorithmen. Auch dies wollen wir anhand der in den folgenden Abschnitten betrachteten Graphenklassen demonstrieren.

7.2 Chordale Graphen

Eine der ältesten Klassen perfekter Graphen ist die der chordalen Graphen.

Definition 7.2.1 (HAJNAL, SURÁNYI 1958) *Ein Graph heißt* chordal, *falls er keinen induzierten C_k, $k \geq 4$, enthält.*

Anders ausgedrückt: in einem chordalen Graphen besitzt jeder Kreis der Länge mindestens vier eine „Sehne“ (engl. chord), das ist eine Kante zwischen zwei auf dem Kreis nicht benachbarten Kreisknoten. Der vollständige Graph K_n ist also beispielsweise chordal, und ein bipartiter Graph ist chordal genau dann, wenn er ein Wald ist.
Chordale Graphen heißen auch triangulierte Graphen oder Dreiecksgraphen, doch wollen wir diese Bezeichnungen aus dem folgenden Grund nicht verwenden:

Übung 7.2.2 *Die Menge der chordalen Graphen ist unvergleichbar (d.h. weder Unter- noch Obermenge) mit der Menge der Triangulationen der Ebene (vgl. Definition 6.1.11).*

Anwendung. ROSE modellierte die Struktur dünnbesetzter, symmetrischer, positiv definiter Matrizen hinsichtlich der CHOLESKY-Faktorisierung mit Hilfe chordaler Graphen

[Ros70, Ros72]. Es zeigt sich, daß das Verfahren ohne Auffüllen der Matrix vonstatten gehen kann genau dann, wenn ein durch die Matrix definierter Graph chordal ist, siehe auch [Sim92, Liu92].

Wir sammeln zunächst einige Eigenschaften chordaler Graphen.

Lemma 7.2.3 (DIRAC 1961) *Ein Graph ist chordal genau dann, wenn jede inklusionsweise minimal trennende Knotenmenge eine Clique induziert.*

Beweis. „⇒": Sei S (o.B.d.A. $|S| > 1$) eine solche minimal trennende Knotenmenge in einem chordalen Graphen G. Es gibt also (mindestens) zwei Komponenten H_1 und H_2 in $G - S$, und alle Knoten aus S haben Nachbarn in beiden H_i, denn sonst würde schon ein kleineres S den Graphen zerfällen. Seien u und v zwei beliebige Knoten aus S. Nach Obigem gibt es für $i = 1, 2$ einen $u - v$-Pfad P_i minimaler Länge, der nur innere Knoten aus H_i benutzt. Der Kreis $C = P_1 \cup P_2$ hat dann Länge ≥ 4 und daher (mindestens) eine Sehne. Da es keine Kanten zwischen H_1 und H_2 gibt und die Pfade P_i kürzest möglich gewählt wurden, bleibt nur die Möglichkeit $\{u, v\} \in E$.
„⇐": Sei C ein Kreis der Länge ≥ 4 in G und seien $u, v \in C$ zwei auf C nicht benachbarte Knoten. Ist $\{u, v\} \in E$, so haben wir bereits eine Sehne gefunden. Andernfalls gibt es eine minimal $u - v$-trennende Knotenmenge S. S enthält dann mindestens je einen Knoten auf den beiden $u - v$-Pfaden auf C. Da S eine Clique ist, sind diese durch eine Kante verbunden. □

Definition 7.2.4 *Ein Knoten in einem Graph heißt* Simplizialknoten, *falls seine Nachbarschaft vollständig verbunden ist.*

Lemma 7.2.5 (DIRAC 1961) *Jeder chordale Graph besitzt einen Simplizialknoten, und falls er nicht der vollständige Graph ist, sogar zwei nicht benachbarte Simplizialknoten.*

Beweis. Durch Induktion nach $n = |V|$. Klar für $n = 1, 2, 3$ und falls der chordale Graph G vollständig ist. Andernfalls besitzt G zwei nicht benachbarte Knoten u und v. Sei S eine inklusionsweise minimale $u-v$-trennende Knotenmenge, so daß $G-S$ in Komponenten H_i, $i \in I$, $|I| \geq 2$, zerfällt. Da S nach Lemma 7.2.3 eine Clique induziert, besitzen die chordalen Graphen $G[H_i \cup S]$ nach Induktionsannahme mindestens je einen Simplizialknoten in H_i. Diese sind wegen $\Gamma(H_i) \subseteq H_i \cup S$ offenbar auch Simplizialknoten in ganz G. □

Daraus können wir nun leicht ableiten:

Satz 7.2.6 (BERGE 1961; HAJNAL, SURÁNYI 1958)
Chordale Graphen und ihre Komplemente sind perfekt.

Beweis. Zunächst bemerken wir wiederum, daß beide Klassen hereditär sind. Die Perfektheit chordaler Graphen folgt somit mittels Induktion über die Anzahl Knoten. Zum Induktionsschritt bemerken wir lediglich, daß ein chordaler Graph G, wenn er nicht vollständig ist - in welchem Fall er offenbar perfekt ist -, nach Lemma 7.2.3 eine trennende Clique C besitzt. Seien H_i, $i \in I$, die Komponenten von $G - C$. Dann sind nach Induktionsvoraussetzung die chordalen Graphen $G_i := G[H_i \cup C]$ perfekt und so auch G, da aus den disjunkten perfekten Graphen G_i durch Identifikation in einer Clique entstanden, vgl. Übung 7.1.4b.
Zum Nachweis der Perfektheit des Komplements eines chordalen Graphen G ist nach Lemma 7.1.8 eine Clique in G vorzuweisen, die alle maximum stabilen Mengen trifft. Genau

dies leistet ein Simplizialknoten s in G, wie ihn Lemma 7.2.5 garantiert, vereinigt mit seiner Nachbarschaft $\Gamma(s)$. □

Übung 7.2.7 [LPP89] *Ein Knoten ist simplizial genau dann, wenn er zu genau einer Clique gehört.*

8.2.1 Erkennung chordaler Graphen

Wir beginnen mit einer algorithmischen Charakterisierung chordaler Graphen. Da auch $G - s$ wieder chordal ist für einen Simplizialknoten s in G, kann man sukzessive Knoten auswählen, die simplizial im Restgraphen sind. Die Knoten eines chordalen Graphen lassen sich also wie folgt anordnen.

Definition 7.2.8 *Sei $G = (V, E)$ ein Graph mit n Knoten. Eine totale Ordnung $\sigma : V \to \{1, \ldots, n\}$ auf V heißt* perfektes Knoten-Eliminationsschema *(PES), falls jeder Knoten $v \in V$ Simplizialknoten im Graph $G[\{u \in V : \sigma(u) \geq \sigma(v)\}]$ ist, d.h. für alle $v \in V$ induziert die sogenannte* obere Nachbarschaft

$$ON(v) := \{u \in \Gamma(v) \mid \sigma(u) > \sigma(v)\}$$

von v bezüglich σ eine Clique in G.

Die Existenz eines solchen Schemas ist auch schon hinreichend für die Chordalität eines Graphen:

Proposition 7.2.9 (Fulkerson, Gross 1965) *Ein Graph ist chordal genau dann, wenn er ein perfektes Knoten-Eliminationsschema besitzt.*

Beweis. Es bleibt zu zeigen, daß nur chordale Graphen ein PES besitzen können. Sei G ein Graph mit PES σ und C ein Kreis in G. Ist $u \in C$ der erste Knoten des Kreises bezüglich σ (d.h. $u = \text{argmin}\{\sigma(v) : v \in C\}$), so sind nach Definition eines PES insbesondere seine beiden Nachbarn auf C verbunden. Also hat jeder Kreis eine Sehne. □

Einen Graphen kann man also durch vollständige Suche nach einem PES auf Chordalität testen - läßt sich in einem Schritt kein Simplizialknoten finden, so ist der untersuchte Graph offenbar nicht chordal, andernfalls wird ein PES bestimmt. Die Laufzeit dieses trivialen Algorithmus ist offenbar $\mathcal{O}(n^2 m)$, geht man von einer Codierung von G durch eine Adjazenzliste aus.

Einen linearen Algorithmus für das Erkennungsproblem bei chordalen Graphen entwickelten Rose, Tarjan und Lueker [RTL76]. Sie zerlegten die Aufgabe dazu in zwei Schritte:

1. Konstruktion einer totalen Ordnung σ auf den Knoten des vorgelegten Graphen G, die ein PES für G ist genau dann, wenn G chordal ist.

2. Testen, ob eine gegebene Ordnung σ auf einem Graphen ein PES darstellt.

Wir stellen eine auf [TY84] basierende, vereinfachte Variante vor.

Ad 1: Maximum-Adjazenz-Suche.
Wir wiederholen: Maximum-Adjazenz-Suche auf einem Graphen G (vgl. Abschnitt 2.1) erzeugt in Zeit $\mathcal{O}(n + m)$ eine totale Ordnung $v_1, \ldots, v_n$ der Knotenmenge wie folgt.

Ausgehend von einem beliebigen Knoten v_1 wählt man als nächsten Knoten stets einen (beliebigen) noch nicht numerierten Knoten, der zu maximal vielen, bereits numerierten Knoten benachbart ist.
Wir behaupten, daß dann für chordale Graphen die umgekehrte Ordnung $v_n, \ldots, v_1$ ein PES darstellt. Um dies zu zeigen, benutzen wir das folgende einfache Kriterium dafür, wann eine totale Ordnung auf V ein PES darstellt.

Proposition 7.2.10 *Eine totale Ordnung σ auf V ist ein PES genau dann, wenn alle Knotenpaare (v_i, v_j) aus V, die durch einen Pfad miteinander verbunden sind, dessen innere Knoten u alle $\sigma(u) < \min\{\sigma(v_i), \sigma(v_j)\}$ erfüllen, auch schon adjazent sind.*

Beweis. „$\Leftarrow$“: trivial.
„$\Rightarrow$“: seien v_i und v_j wie angegeben, $\{v_i, v_j\} \notin E$ und p ein kürzester solcher $v_i - v_j$–Pfad. Betrachte den inneren Knoten $u := \operatorname{argmin}\{\sigma(v) : v \in p\}$ von p, der in σ am weitesten vorne angeordnet ist. Nach Definition eines Simplizialknotens sind die Nachbarn von u auf p durch eine Kante verbunden, und p daher nicht kürzest möglich, Widerspruch. □

Proposition 7.2.11 *Sei G ein chordaler Graph. Eine* Maximum_Adjazenz_Suche *besuche die Knoten von G in der Reihenfolge $v_1, \ldots, v_n$. Dann bilden die Knoten von V in umgekehrter Reihenfolge ein PES für G.*

Der folgende neue Beweis läßt diese Aussage hoffentlich einleuchten.
Beweis. Sei $\sigma : V \to \{1, \ldots, n\}$, $\sigma[v_i] := n + 1 - i$. Wir überlegen uns zunächst, daß die Ordnung σ die folgende Eigenschaft hat:

($\star$) Angenommen, es gilt $\sigma(u) < \sigma(v) < \sigma(w)$ und $\{u, w\} \in E$, $\{v, w\} \notin E$.
Dann gibt es einen Knoten z, so daß $\sigma(v) < \sigma(z)$, $\{v, z\} \in E$ und $\{u, z\} \notin E$.

Der Grund ist einfach der, daß die Maximum_Adjazenz_Suche den Knoten v vor dem Knoten u numeriert hat, obwohl u im Gegensatz zu v mit dem bereits numerierten Knoten w benachbart ist. Der Knoten v hat also mindestens so viele, bereits numerierte Nachbarn wie u, und deshalb muß v mindestens einen bereits numerierten Nachbarn z besitzen, der nicht auch zu u adjazent ist.
Angenommen nun, σ sei kein PES. Dann gibt es nach Proposition 7.2.10 zwei nicht benachbarte Knoten v und w, die durch einen $v - w$–Pfad miteinander verbunden sind, der nur innere Knoten x benutzt, die $\sigma(x) < \min\{\sigma(v), \sigma(w)\}$ erfüllen. Ohne Einschränkung seien v und w so gewählt, daß $\sigma(w)$ maximal ist und, falls mehrere Möglichkeiten bestehen, auch $\sigma(v)$. Für solche Knoten v und w sei P ein kürzester $v - w$–Pfad, dessen innere Knoten alle „links von v und w liegen“. Der Pfad P enthält mindestens einen inneren Knoten; sei u der mit w verbundene innere Knoten des Pfades P. Dann gibt es aufgrund der Eigenschaft ($\star$) einen Knoten z, der (bezüglich σ) noch rechts von v liegt (d.h. $\sigma(v) < \sigma(z)$) und mit v verbunden ist, nicht aber mit u. Da also auch w und z durch einen Pfad verbunden sind, der nur Knoten „links von w und z“ benutzt, gilt nach Wahl von v und w: $\{w, z\} \in E$. Mithin bildet $P + \{v, z\} + \{z, w\}$ einen Kreis in G. Nach Wahl von P inzidiert jede Sehne dieses Kreises mit z. Falls mit z keine Sehne inzidiert, so bildet $P + \{v, z\} + \{z, w\}$ also einen induzierten Kreis auf mindestens 4 Knoten. Andernfalls sei x der letzte Knoten auf dem $v - w$–Pfad P, der mit z indizidiert. Wegen $\{u, z\} \notin E$ gilt $x \not\equiv u$, und $(x, \ldots, u, w, z)$ bildet einen solchen induzierten Kreis – im Widerspruch zur Chordalität von G. □

Ad 2: Testen der Eigenschaft PES.
Wir gehen von einer Codierung des vorgelegten Graphen durch eine Adjazenzliste aus. Zu einer totalen Ordnung σ auf V kann man dann die Mengen $ON(\sigma^{-1}(i))$, $i = 1, \ldots, n$, sicherlich in Zeit $\mathcal{O}(|V| + |E|)$ bestimmen. Schwierigkeiten bereitet der Test, ob die Knotenmengen $ON(\sigma^{-1}(i))$ wirklich Cliquen induzieren. Der Kunstgriff, der verhindert, daß das Vorhandensein ein und derselben Kante zu oft überprüft wird, besteht darin, den Auftrag zur Überprüfung der oberen Nachbarschaft $ON(v)$ auf vollständiges Verbundensein an den sogenannten *ON-Führer* f, den oberen Nachbarn von v mit der kleinsten Nummer in σ, weiterzureichen.
Der Test, ob $(ON(v) - f) \subseteq \Gamma(f)$, stellt einen Teil der geforderten Adjazenzen sicher. Er wird jedoch erst ausgeführt, wenn f an der Reihe ist - bis dahin werden alle Knoten, die aus verschiedenen Mengen $ON(v) - f$ stammen für verschiedene v, $\sigma(v) \leq \sigma(f)$, und deren Adjazenz mit f überprüft werden muß, in einer Menge $A(f)$ gesammelt.
Daß auch jeweils überprüft wird, ob $ON(v) - f$ eine Clique bildet, dafür sorgt dann schon f selbst - denn wegen $ON(v) - f \subseteq ON(f)$ ist dies Teil des Simplizialknotentests für f.
Die folgende Routine realisiert dieses Vorgehen. Der Wert der Funktion PES ist also genau dann `TRUE`, wenn der Parameter σ ein PES für den Graphen G darstellt.

```
FUNCTION PES (G,σ) : BOOLEAN;

BEGIN
  FOR v ∈ V DO A[v] := ∅;
  FOR i := 1 TO n DO BEGIN
    v := σ^-1[i];
    ON := {w ∈ Γ(v)| σ^-1[w] > σ^-1[v]};
    IF A[v] ⊈ ON THEN BEGIN
      PES:= FALSE;
      EXIT;
    END;
    IF ON ≠ ∅ THEN BEGIN
      f := argmin{σ[w]: w ∈ ON};
      A[f] := A[f] ∪ (ON \ {f});
    END;
  END; {for i}
  PES:= TRUE;
END;
```

Wir behaupten nun:

Lemma 7.2.12 *Der Algorithmus* PES *kann so implementiert werden, daß seine Laufzeit* $\mathcal{O}(|V| + |E|)$ *beträgt.*

Beweis. Verwendet man zur Verwaltung von ON eine einfach verkettete Liste, so ist offenbar die Gesamtzahl aller die Menge ON betreffenden Operationen (summiert über alle Durchläufe der (`FOR` i)-Schleife) durch $\mathcal{O}(|V| + |E|)$ beschränkt - bis auf den Test $A(v) \not\subseteq ON$. Hierfür benutzt man einen n-Vektor *test*, der eingangs auf 0 initialisiert wird, und geht wie folgt vor:

```
Teilmenge := TRUE;
FOR u ∈ ON DO test[u] := 1;
```

```
FOR u ∈ A[v] DO
  IF test[u] = 0 THEN Teilmenge := FALSE;
FOR u ∈ ON DO test[u] := 0;
```

Da die Anzahl aller in verschiedenen ONs auftretenden Knoten durch $|E|$ beschränkt ist und die Mengen $A(v)$ nur durch Elemente aus ONs gespeist werden, sind auch die $A(v)$ betreffenden Operationen insgesamt durch $|E|$ beschränkt. Man kann also $A(v)$ einfach als Multimenge darstellen - eine einfach verkettete Liste, an die man neue Elemente stets hinten anhängt, auch wenn sie bereits in der Liste enthalten sind. □

Wir fassen zusammen:

Satz 7.2.13 *Chordale Graphen können in linearer Zeit erkannt werden.*

Übung 7.2.14 (Bender, Richmond, Wormald 1985) *Ein chordaler Graph besitzt für jede seiner Cliquen ein perfektes Knoteneliminationsschema, das die Knoten der Clique am Schluß anordnet.*

8.2.2 Lineare Algorithmen für χ, ω, α und θ bei chordalen Graphen

Mit Hilfe eines PES ist es ein leichtes, für chordale Graphen effiziente (sogar lineare) Algorithmen anzugeben, die eine maximum Clique, eine optimale Knotenfärbung, eine maximum stabile Menge oder eine minimale Cliquenüberdeckung konstruieren.
Nach Definition eines PES induzieren die Mengen $\{v\} \cup ON(v)$ für alle $v \in V$ eine Clique. Umgekehrt ist aber auch schon jede Clique C in G von der Form $\{v\} \cup ON(v)$ – man wähle $v := \text{argmin}\{\sigma(u) : u \in C\}$. Diese einfache Beobachtung impliziert sofort:

Proposition 7.2.15 (Fulkerson, Gross 1965) *Ein chordaler Graph G auf n Knoten hat höchstens n maximale Cliquen (mit Gleichheit genau dann, wenn G leer ist).* □

Eine maximum Clique (und damit $\omega(G)$) in einem chordalen Graphen G findet man also einfach dadurch, daß man zunächst wie oben ein PES konstruiert und anschließend die größte Clique $\{v\} \cup ON(v)$ heraussucht.
Ebenso einfach ist es, mittels eines PES eine $\chi(G)$-Färbung zu finden:

Übung 7.2.16 *Sei σ ein PES für den chordalen Graphen G. Der Greedy-Algorithmus bezüglich der Anordnung $(\sigma^{-1}(n), \ldots, \sigma^{-1}(1))$ konstruiert eine optimale zulässige Knotenfärbung von G.*

Auch eine maximum stabile Menge S findet man greedy-mäßig [Gav72]. Definiere $s_1 := \sigma^{-1}(1)$ und induktiv s_k als den ersten Knoten hinter s_{k-1} bezüglich σ, der keinen Nachbarn in $\{s_1, \ldots, s_{k-1}\}$ hat:

$$s_k := \text{argmin}\left\{\sigma(v) | \sigma(v) > \sigma(s_{k-1}) \wedge v \notin \bigcup_{i=1}^{k-1} ON(s_i)\right\}.$$

Dann ist $S := \{s_1, \ldots, s_l\}$ offenbar stabil und $\{\{s_k\} \cup ON(s_k)\}_{k=1}^{l}$ eine Cliquenüberdekkung von G derselben Kardinalität. Wegen $\theta(G) \geq \alpha(G)$ sind beide schon optimal.

Übung 7.2.17 *Sei σ ein PES für den chordalen Graphen G. Eine Clique in G der Form $\{v\} \cup ON(v)$ bezüglich σ ist maximal genau dann, wenn in der Prozedur* PES *zum Zeitpunkt $v := \sigma[i]$ gilt $|A[v]| < |ON(v)|$.*

8.2.3 Schnittgraphdarstellung chordaler Graphen

Dieser Abschnitt gibt eine weitere interessante Charakterisierung chordaler Graphen, die uns insbesondere beim Beweis von Satz 9.1.9 von Nutzen sein wird. Den Beweis der folgenden hübschen Hilfsaussage überlassen wir dem Leser.

Übung 7.2.18 *Eine Familie $\mathcal{T} = \{T_i : i = 1, \ldots, n\}$ von Teilbäumen (aufgefaßt als Knotenmengen) eines Baumes B besitzt die* HELLY-Eigenschaft*: Wenn sich die Teilbäume aus $\mathcal{T}$ paarweise schneiden, so haben sie schon allesamt einen Knoten gemeinsam: d.h. es wird $\bigcap_{i=1}^{n} V(T_i) \neq \emptyset$.*

Satz 7.2.19 (WALTER 1978, GAVRIL 1974a, BUNEMAN 1974) *Sei $G = (V, E)$ ein Graph mit Knotenmenge $V = \{v_1, \ldots, v_n\}$. Dann sind die folgenden Aussagen äquivalent:*

(i) G ist chordal;

(ii) G ist der Schnittgraph einer Familie von Teilbäumen eines Baumes, d.h. es gibt einen Wirtsbaum $\mathcal{B}$ und eine Familie $T_1, \ldots, T_n$ von Teilbäumen von $\mathcal{B}$, so daß gilt:
$$\{v_i, v_j\} \in E \Leftrightarrow V(T_i) \cap V(T_j) \neq \emptyset;$$

(iii) Es gibt einen Baum $\mathcal{B}$ auf der Menge $\mathcal{K}$ der maximalen Cliquen von G, so daß für jedes Paar K', K'' von maximalen Cliquen in G gilt: die Clique $K' \cap K''$ ist in jeder maximalen Clique auf dem (eindeutig bestimmten) $K' - K''$−Pfad in $\mathcal{B}$ enthalten. Ein solcher Baum $\mathcal{B}$ heißt Cliquenbaum *für G.*

Beweis. (i) $\Rightarrow$ (ii): Wir induzieren nach n. Für $n = 1$ (und jeden vollständigen Graphen) wähle $\mathcal{B} = K_1$. Sei also G ein chordaler Graph auf den Knoten $v_1, \ldots, v_n$ und $s = v_n$ ein Simplizialknoten in G. Nach Induktionsvoraussetzung gibt es eine Schnittgraphdarstellung $\langle \mathcal{B}_{n-1}, \{T_1, \ldots, T_{n-1}\}\rangle$ für $G[V - s]$. Da $\Gamma(s)$ eine Clique in $G[V - s]$ ist, schneiden sich die Teilbäume T_v, $v \in \Gamma(s)$, paarweise. Aufgrund von Übung 7.2.18 gibt es also im Baum $\mathcal{B}_{n-1}$ einen Knoten $a \in \bigcap_{v \in \Gamma(s)} V(T_v)$. Hänge im Baum $\mathcal{B}_{n-1}$ ein Blatt b an den Knoten a an, definiere $T_s := \{a\}$ und verlängere alle Teilbäume T_v, $v \in \Gamma(s)$, bis nach b. Dann ist $\mathcal{B}_{n-1} + \{a, b\}$ der gewünschte Baum.

(ii) $\Rightarrow$ (iii): Zu einer Schnittgraphdarstellung $T_1, \ldots, T_n$ von $G = (V, E)$ in einem Wirtsbaum $\mathcal{B} = (V_\mathcal{B}, E_\mathcal{B})$ definiere für $b \in V_\mathcal{B}$

$$C_b := \{v_i \in V : T_i \ni b\}.$$

Dann bilden die Knotenmengen $C_b \subseteq V$, $b \in V_\mathcal{B}$, Cliquen in G, weil sich die Teilbäume T_i, $v_i \in C_b$, in b schneiden. Offensichtlich ist zudem für zwei maximale Cliquen $K', K'' \in \mathcal{K}$ jeder Knoten $v \in K' \cap K''$ nach Konstruktion in jeder Clique auf dem $K' - K''$−Pfad in $\mathcal{B}$ enthalten.

Für jede in G maximale Clique $C \in \mathcal{K}$ gibt es darüberhinaus aufgrund der HELLY-Eigenschaft von Teilbäumen eines Baumes einen Knoten $b \in V_\mathcal{B}$, so daß $C \subseteq C_b$ und mithin $C = C_b$ – wir sagen, die Clique C wird vom Knoten b repräsentiert. Angenommen nun, eine Clique C_a, $a \in V_\mathcal{B}$, ist nicht maximal in G. Sei $C = C_b$ eine C_a enthaltende maximale Clique in G. Dann gilt für alle Knoten c auf dem (eindeutigen) $a - b$−Pfad in $\mathcal{B}$: $C_a \subseteq C_c$. Durch Kontraktion der ersten Kante auf diesem Weg erhält man eine

hinsichtlich $|V_{\mathcal{B}}|$ kleinere Schnittgraphdarstellung von G. Analog verfährt man, falls eine Clique mehrfach repräsentiert wird. Folglich sind in einer hinsichtlich $|V_{\mathcal{B}}|$ minimalen Schnittgraphdarstellung von G alle Cliquen C_b, $b \in V_{\mathcal{B}}$, in G maximal und verschieden.

(iii) $\Rightarrow$ (ii): Setze $T_v := \mathcal{B}[\{K \in \mathcal{K} : K \ni v\}]$ für alle $v \in V$. Dann ist T_v nach Definition eines Cliquenbaumes zusammenhängend, d.h. ein Teilbaum des Wirtsbaumes $\mathcal{B}$, und die Familie der Teilbäume T_v ($v \in V$) von $\mathcal{B}$ bildet eine Schnittgraph-Darstellung von G:
Sei $\{u, v\} \in E$. Dann ist die Clique $\{u, v\}$ in einer maximalen Clique $K \in \mathcal{K}$ enthalten. Also ist die Clique K nach Definition der Teilbäume sowohl Knoten des Teilbaumes T_u wie des Teilbaumes T_v, und es folgt $V(T_u) \cap V(T_v) \neq \emptyset$.
Gelte nun umgekehrt $V(T_u) \cap V(T_v) \neq \emptyset$. Sei $K \in V(T_u) \cap V(T_v)$. Dann folgt $\{u, v\} \subseteq K$ und damit $\{u, v\} \in E$.

(ii) $\Rightarrow$ (i): Sei $C = (v_0, \ldots, v_{k-1})$ ein Kreis in G der Länge $k-1 \geq 4$, und seien $T_0, \ldots, T_{k-1}$ die zugehörigen Bäume der Schnittgraphdarstellung von G. Angenommen, C besäße keine Sehne. Dann schneiden sich diese Bäume genau dann, wenn sie in der Liste $T_0, \ldots, T_{k-1}$ aufeinanderfolgen (modulo k). Also gibt es in $\mathcal{B}$ einen Knoten $x_0 \in [V(T_0) \cap V(T_1)] \setminus V(T_{k-1})$ mit $dist_{T_0}(T_{k-1}, x_0)$ minimal und einen Knoten $x_1 \in [V(T_1) \cap V(T_2)] \setminus V(T_0)$ mit $dist_{T_1}(x_0, x_1)$ minimal. Sei W der (eindeutige) $x_0 - x_1$–Pfad in T_1. Setze nun für $i = 2, \ldots, k-1$ den Weg W jeweils zu einem Knoten $x_i \in [V(T_i) \cap V(T_{i+1})] \setminus V(T_{i-1})$ mit $dist_{T_i}(x_{i-1}, x_i)$ minimal fort. Ein kürzester $x_{k-1} - x_0$–Pfad in T_0 schließt den Weg W zu einem Zykel. Tatsächlich ist W jedoch ein Kreis in $\mathcal{B}$: einerseits sind die Knoten $x_0, \ldots, x_{k-1}$ nach Konstruktion alle verschieden, und andererseits liegen die Knoten in den Abschnitten auf W zwischen zwei Knoten x_{i-1} und x_i alle jeweils in T_i – es könnten sich also allenfalls benachbarte Abschnitte schneiden; da die Knoten eines jeden Abschnitts jedoch in $(T_{i-1} \cap T_i) \cup (T_i \setminus T_{i+1})$ liegen, ist auch das unmöglich. Also wurde im Baum $\mathcal{B}$ tatsächlich ein Kreis konstruiert – Widerspruch. □

Aus dem Beweis ergibt sich unmittelbar:

Korollar 7.2.20 *Eine Schnittgraph-Darstellung eines chordalen Graphen G durch Teilbäume eines Baumes $\mathcal{B}$ ist minimal bezüglich $|V_{\mathcal{B}}|$ genau dann, wenn $V_{\mathcal{B}} = \mathcal{K}$.*

Der vollständige Graph auf der Menge $\mathcal{K}$ der maximalen Cliquen von G mit Kantengewichten

$$w(\{K, K'\}) := |K \cap K'|$$

heißt der *Cliquenschnittgraph* $\mathcal{K}(G)$ von G.

Proposition 7.2.21 (Bernstein, Goodman 1981; Shibata 1988)
Ein spannender Baum $\mathcal{B}$ von $\mathcal{K}(G)$ ist genau dann ein Cliquenbaum für G, wenn er ein maximum spannender Baum des Cliquenschnittgraphen $\mathcal{K}(G)$ ist.

Beweis. „$\Rightarrow$“: Sei $\mathcal{B}$ ein Cliquenbaum. Betrachte eine Kante $\{K', K''\} \in \binom{\mathcal{K}}{2} \setminus E(\mathcal{B})$. Da $\mathcal{B}$ ein Cliquenbaum ist, gilt für den (eindeutigen) $K' - K''$–Pfad $K' = K_0, \ldots, K_\ell = K''$ in $\mathcal{B}$: $K' \cap K'' \subseteq K_i$ für alle $0 \leq i \leq \ell$. Mithin haben alle Kanten $\{K_i, K_{i+1}\}$, $0 \leq i \leq \ell - 1$, auf diesem Pfad Gewicht $w(\{K_i, K_{i+1}\}) \geq w(\{K', K''\})$. Nach Übung 1.3.17 ist $\mathcal{B}$ also maximum.
„$\Leftarrow$“: Da die Menge der maximum spannenden Bäume im Spannbaumgraphen $\mathcal{T}(G)$ von G zusammenhängt (vgl. Übung 1.3.18), genügt es, zu zeigen: wenn $\mathcal{B}$ ein Cliquenbaum von G ist, dann auch jeder maximum spannende Baum T von G mit $|E(T) \Delta E(\mathcal{B})| = 2$. Dies sieht

man wie folgt ein. Es sei $e = \{K', K''\} = E(T) \setminus E(\mathcal{B})$ und $f = \{K_i, K_{i+1}\} = E(\mathcal{B}) \setminus E(T)$. Da T wie $\mathcal{B}$ maximales Gewicht haben, gilt $w(e) = w(f)$. Da $\mathcal{B}$ zudem ein Cliquenbuam ist, folgt $K_i \cap K_{i+1} = K \cap K'$. Also werden alle die Teilbäume $T(v) = \mathcal{B}[\{K \in \mathcal{K} : K \ni v\}]$, $v \in V(G)$, von $\mathcal{B}$, die durch Herausnahme der Kante f aus $\mathcal{B}$ zerbrechen, durch die Hinzunahme der Kante e wieder zusammengefügt. Somit ist auch T ein Cliquenbaum von G. □

Proposition 7.2.22 (Blair, England, Thomason 1988) *Ein Cliquenbaum für einen chordalen Graphen kann in linearer Zeit konstruiert werden.*

Beweis. Sei G ein chordaler Graph. Wir führen eine leicht modifizierte Maximum_Adjazenz_Suche auf G durch. Die Maximum_Adjazenz_Suche besuche die Knoten von G in der Reihenfolge $v_1, \ldots, v_n$. Wie oben gesehen, bilden dann die Knoten von V in umgekehrter Reihenfolge ein PES für G. Sei $\sigma : V \to \{1, \ldots, n\}$, $\sigma[v_i] := n + 1 - i$, das so konstruierte PES für G. Entscheidend ist die folgende Beobachtung:

Maximum_Adjazenz_Suche *schöpft sukzessive maximale Cliquen aus,*

d.h. sie konstruiert zu Beginn eine maximale Clique K_1, dann wählt sie einen Knoten v, der nicht in dieser Clique liegt (aber maximal viele Nachbarn in K_1 hat) und fügt sukzessive die Knoten einer maximalen Clique K_2, die die Clique $ON(v) + v$ enthält, hinzu u.s.w. (wie in der Funktion PES seien $ON(v) := \{w \in \Gamma(v) | \ \sigma[w] > \sigma[v]\}$ die bereits besuchten Nachbarn von v). Dies läßt sich wie folgt einsehen. Es werde im $(k-1)$−ten Schritt ein Knoten u und im k-ten Schritt ein Knoten v gewählt. Dann ist, weil ein PES konstruiert wird, $ON(v) + v$ eine Clique in G. Aufgrund des Auswahlkriteriums der Maximum_Adjazenz_Suche gilt $|ON(v) + v| \leq |ON(u) + u| + 1$ mit Gleichheit genau dann, wenn $u \in ON(v) \Leftrightarrow ON(v) = ON(u) + u$. Also war $ON(u) + u$ eine maximale Clique in G genau dann, wenn $|ON(v) + v| \leq |ON(u) + u|$. Damit haben wir also sogar ein Kriterium an der Hand, um zu entscheiden, wann eine neue maximale Clique „betreten“ wird.

Die Knotenmenge $\mathcal{K}$ des Baumes $\mathcal{B}$ wird nun wie folgt parallel zur Maximum_Adjazenz_Suche konstruiert. Solange die Funktion $ON(u) + u$ streng monoton steigt, sammeln wir die besuchten Knoten in der maximalen Clique K_1. Sobald mit einem Knoten v eine neue Clique betreten wird, initialisieren wir eine neue Clique K_2 mit $K_2 := ON(v) + v$ und sammeln, solange die Funktion $ON(u) + u$ wieder streng monoton steigt, alle unterwegs besuchten Knoten in diese Clique, u.s.w.

Um die Kanten von $\mathcal{B}$ leichter definieren zu können, nehmen wir an, daß jeder Knoten $v \in V$ in dem Moment, in dem er besucht wird, eine Marke $MaxClique[v]$ erhält, die die Nummer der maximalen Clique angibt, in die er „gesammelt” wurde. Man sieht leicht ein, daß $MaxClique[v]$ diejenige maximale Clique $ON(\sigma^{-1}[i]) + \sigma^{-1}[i]$ von G ist, die v enthält und im Schema σ am weitesten „hinten“ steht, d.h. die größte Nummer i hat.

Wenn nun mit einem Knoten v eine neue Clique betreten wird, so fügen wir in die (zu Anfang leere) Kantenmenge $E_{\mathcal{B}}$ die Kante $\{MaxClique[v], MaxClique[f]\}$ ein, wobei $f := argmin\{\sigma[w] : w \in ON(v)\}$ wieder der ON-Führer von v sei. Da dies denselben Effekt hat, wie die Knoten der Clique $MaxClique[v]$ (gemäß dem Beweis zu Satz 7.2.19) einzeln zu $\mathcal{B}$ hinzuzufügen und die Kanten nachher wieder zu kontrahieren, liefert dieses Verfahren, wie gewünscht, einen Cliquenbaum. □

Azyklische Hypergraphen oder duale Hyperbäume, wie sie bei der Cliquen-Struktur von chordalen Graphen auftreten, werden auch von Brandstädt [Bra94a] näher untersucht; sie finden in der Theorie der relationalen Datenbanken Anwendung [BFM+81].

Übung 7.2.23 *Sei G ein chordaler Graph.*
a)[BET88] *Der obige Algorithmus zur Konstruktion eines Cliquenbaumes $\mathcal{B}$ in G kann interpretiert werden als der* PRIM*-Algorithmus zur Konstruktion eines spannenden Baumes maximalen Gewichts im Cliquenschnittgraphen $\mathcal{K}(G)$ von G.*
b)[Shi88] *Der Algorithmus von* KRUSKAL *zur Konstruktion eines spannenden Baumes maximalen Gewichts in $\mathcal{K}(G)$ liefert ebenso einen Cliquenbaum von G. Was hat er für eine Laufzeit?*

Übung 7.2.24 [HL89] *Sei $\mathcal{B}$ ein Cliquenbaum eines chordalen Graphen $G = (V, E)$. Dann ist eine Knotenmenge $S \subset V$ eine inklusionsweise minimal trennende Knotenmenge von G genau dann, wenn $S = K' \cap K''$ für eine Kante $\{K', K''\} \in E(\mathcal{B})$.*

7.3 Vergleichbarkeitsgraphen

Definition 7.3.1 *Eine zweistellige Relation „$\leq$“ auf einer Menge P heißt* teilweise Ordnung *(englisch partially ordered set), in Zeichen $P_\leq$, falls für alle $x, y, z \in P$ gilt:*

$$\begin{array}{lll} (1) & x \leq x & (Reflexivität) \\ (2) & x \leq y \wedge y \leq z \;\Rightarrow\; x \leq z & (Transitivität) \\ (3) & x \leq y \wedge y \leq x \;\Rightarrow\; x = y & (Antisymmetrie) \end{array}$$

Zwei Elemente $x, y \in P$ werden *vergleichbar* genannt, wenn $x \leq y$ oder $y \leq x$ gilt, sonst unvergleichbar. Eine teilweise Ordnung $P_\leq$ heißt *lineare Ordnung* oder auch *Kette*, falls je zwei Elemente von P vergleichbar sind, bzw. *Antikette*, falls je zwei Elemente von P unvergleichbar sind. Man sagt, y bedeckt x, in Zeichen $x <\cdot\, y$, falls $x \leq y$ gilt und $x \leq a \leq y \Rightarrow a \in \{x, y\}$.

Beispiele. (1) Die Menge der 2^n Teilmengen einer n-elementigen Menge M bildet bezüglich der Inklusion eine teilweise geordnete Menge. Jedes Niveau $\binom{M}{k}$ bildet eine Antikette.
(2) Die natürlichen Zahlen bilden mit der Teilbarkeitsrelation eine teilweise geordnete Menge. □

Definition 7.3.2 *Ein Graph $G = (V, E)$ heißt* Vergleichbarkeitsgraph, *falls es eine teilweise Ordnung $P_\leq$ und eine Bijektion $f : V \to P$ gibt, so daß für alle $u, v \in V$, $u \neq v$, gilt: $\{u, v\} \in E$ genau dann, wenn $f(u)$ und $f(v)$ in $P_\leq$ vergleichbar sind.*

Beispiel. Jeder bipartite Graph $B = (U, V, E)$ ist ein Vergleichbarkeitsgraph: definiere eine teilweise Ordnung „$\leq$“ auf $U \cup V$ durch

$$x \leq y \quad :\Leftrightarrow \quad (\{x, y\} \in E \wedge x \in U \wedge y \in V) \;\vee\; (x = y). \qquad \square$$

Lemma 7.3.3 *Ein Graph ist ein Vergleichbarkeitsgraphen genau dann, wenn er* transitiv orientierbar *ist, d.h. eine Orientierung $o : E \to V \times V$ seiner Kanten besitzt, so daß gilt:*

$$(x, y) \in o(E) \wedge (y, z) \in o(E) \;\Rightarrow\; (x, z) \in o(E).$$

Beweis. Sei $P_\leq$ eine teilweise Ordnung zu G. Interpretiere obige Funktion $f : V \to P$ wie folgt:

$$o(\{u, v\}) = (u, v) \;\Leftrightarrow\; f(u) \leq f(v). \qquad \square$$

Beispiel. Eine transitive Orientierung der Kanten eines bipartiten Graphen $B = (U, V, E)$ ist dadurch gegeben, daß man alle Kanten $e \in E$ von U nach V orientiert. □

Beachte, daß sich aus der Antisymmetrie (Bedingung (3)) einer teilweisen Ordnung ergibt, daß ein transitiv orientierter Graph insbesondere azyklisch ist, d.h. keine gerichteten Kreise enthält.

Es gibt einen $\mathcal{O}(n + m)$-Algorithmus, der eine Orientierung der Kanten eines ungerichteten Graphen G konstruiert, die transitiv ist, wenn G ein Vergleichbarkeitsgraph ist [Spi92, MS95]. Wie (schnell) kann man eine Orientierung auf Transitivität testen?

Satz 7.3.4 (MIRSKY 1971a) *Sei $P_\leq$ eine teilweise geordnete Menge. Dann ist die maximale Länge (Kardinalität) einer Kette in $P_\leq$ gleich der minimalen Anzahl von Antiketten, in die sich P partitionieren läßt.*

Beweis. Da jede Antikette nur höchstens ein Element einer Kette enthalten kann, ist die Ungleichung „$\leq$“ trivial.
„$\geq$“: Ein Element $y \in P_\leq$ heiße minimal, falls für alle $x \in P$ gilt: $x \leq y \Rightarrow x = y$. Damit ist die folgende, sogenannte Höhenfunktion $h : P \to \mathbb{N}$ von $P_\leq$ wohldefiniert:

$$h(y) := \begin{cases} 1, & \text{falls } y \text{ ein minimales Element von } P_\leq \text{ ist;} \\ \max\{h(x)+1 \mid x \in P \wedge x \leq_P y\} & \text{sonst.} \end{cases}$$

Der Wert $h(y)$ gibt damit die Kardinalität einer längsten Kette $x = z_1 \leq z_2 \leq \cdots \leq z_{h(y)} = y$ von y zu einem minimalen Element $x \in P_\leq$ an; $\omega(P_\leq) = \max\{h(y) : y \in P\}$ bezeichne die Länge einer längsten Kette in $P_\leq$. Wir geben eine Zerlegung von P in höchstens $\omega(P_\leq)$ viele Antiketten an. Für zwei vergleichbare Elemente x und y in $P_\leq$ gilt nach Definition von h offenbar entweder $h(x) > h(y)$ oder $h(x) < h(y)$. Also bilden für $1 \leq i \leq \omega(P_\leq)$ die „Niveaus“ $H_i := \{y \in P_\leq | h(y) = i\}]$ Antiketten in $P_\leq$. □

Korollar 7.3.5 *Vergleichbarkeitsgraphen sind perfekt.*

Beweis. Offensichtlich entsprechen Cliquen (bzw. stabile Mengen) in einem Vergleichbarkeitsgraphen den Ketten (bzw. Antiketten) in der zugehörigen teilweisen Ordnung. Also stellt die Höhenfunktion h eine $\omega(G)$-Färbung dar, d.h. für jeden Vergleichbarkeitsgraph gilt $\chi(G) \leq \omega(G)$. □

Die duale Version des Satzes 7.3.4 lautet:

Satz 7.3.6 (DILWORTH 1950) *Die maximale Länge (= Kardinalität) einer Antikette in einer teilweisen Ordnung $P_\leq$ ist gleich der minimalen Anzahl von Ketten, in die sich P partitionieren läßt.*

Beweis [Tve67]. Die Richtung „$\leq$“ ist trivial.
„$\geq$“: Wir induzieren über $|P|$, $P := P_\leq$. Für $|P| \leq 1$ ist die Aussage trivial. Bezeichne $\alpha(P)$ die Kardinalität einer maximum Antikette und $\theta(P)$ die minimale Anzahl Ketten in einer Kettenzerlegung von P. Sei C eine maximale Kette in P.
Falls C jede Antikette in $P \setminus C$ trifft, d.h. falls jede Antikette in $P \setminus C$ nur noch höchstens $\alpha(P) - 1$ Elemente hat, so läßt sich $P \setminus C$ nach Induktionsvoraussetzung in höchstens $\alpha(P) - 1$ viele Ketten partitionieren, und wir sind fertig.
Andernfalls gibt es eine Antikette $A := \{a_1, \ldots, a_\alpha\}$ der Länge $\alpha := \alpha(P)$ in $P \setminus C$.

Definiere $P^- := \{x \in P : \exists a \in A\ (x \leq_P a)\}$ und entsprechend P^+. Da A eine Antikette ist, gilt $P^- \cap P^+ = A$, und, da sie zudem eine maximale Antikette ist, gilt $P \subseteq P^- \cup P^+$. Da die Kette C maximal ist, aber kein Element aus A enthält, ist das maximale Element in C nicht in P^- enthalten. Also läßt sich die Induktionsvoraussetzung auf P^- anwenden, d.h. P^- läßt sich in höchstens $\alpha(P^-) = \alpha(P)$ viele Ketten $P^- = C_1^- \cup \cdots \cup C_\alpha^-$ zerlegen, wobei o.B.d.A. $a_i \in C_i^-$. Dann ist a_i für $1 \leq i \leq \alpha$ das maximale Element der Kette C_i^-, denn für jedes Element $x \in P^- \setminus A$ gibt es nach Definition ein $a_x \in A$ mit $x <_P a_x$; da A eine Antikette und $<_P$ transitiv ist, kann es also kein $a \in A$ mit $a <_P x$ geben.
Dieselbe Argumentation liefert eine Kettenzerlegung von P^+, die die Elemente aus A als minimale Elemente hat. Heftet man die Ketten C_i^- und C_i^+ jeweils in den Elementen a_i zusammen, so erhält man eine Kettenzerlegung von P aus $\alpha(P)$ Ketten. □

Der Satz von DILWORTH läßt sich damit äquivalent wie folgt formulieren.

Korollar 7.3.7 *Ein Vergleichbarkeitsgraph ist perfekt.* □

Die Optimierungsprobleme GRAPH COLORING und MAXIMUM CLIQUE lassen sich auf Vergleichbarkeitsgraphen wie folgt lösen.

Übung 7.3.8 a) *Eine topologische Ordnung (auch topologische Sortierung der Knoten) von G ist eine Permutation $\sigma : V \to \{1, \ldots, n\}$ der Knotenmenge, so daß gilt*

$$(u, v) \in A \ \Rightarrow\ \sigma(u) < \sigma(v).$$

Ein Digraph besitzt eine topologische Ordnung genau dann, wenn er azyklisch ist.
b) *Ein gerichteter Graph G ist azyklisch genau dann, wenn eine gerichtete Tiefensuche (die in jedem Knoten nur die ausgehenden Kanten berücksichtigt) in G keine Rückwärtskanten (sondern höchstens Querkanten) liefert.*
c) [Tar74] *Sei $G = (V, A)$ ein azyklischer Digraph. Die folgende Variante der gerichteten Tiefensuche produziert eine topologische Ordnung von G in linearer Zeit: Erhöhe t und setze $DFSnum[v]$ nicht am Beginn der Prozedur* TIEFENSUCHE, *sondern am Ende (nach der* `FOR`*-Schleife); rufe im Hauptprogramm* TIEFENSUCHE (G, v) *für alle bisher noch nicht besuchten Knoten auf, d.h. für alle Knoten $v \in V$ mit $DFSnum[v] = 0$. Dann ist $\sigma[v] := n + 1 - DFSnum[v]$ eine topologische Ordnung von G.*
d) [ELP72] *Sei ein Vergleichbarkeitsgraph G mit einer transitiven Orientierung auf E gegeben. Der Greedy-Algorithmus für* GRAPH COLORING, *angewandt auf eine Anordnung $DFSnum^{-1}(1), \ldots, DFSnum^{-1}(n)$ der Knoten von G, wie sie die gerichtete Tiefensuche aus Aufgabenteil c liefert, berechnet eine optimale Färbung von G. Durch leichte Modifikation dieses Algorithmus findet man eine maximum Clique in G in linearer Zeit.*

Die Optimierungsprobleme MINIMUM CLIQUE PARTITION und MAXIMUM INDEPENDENT SET (d.h. insbesondere die Bestimmung von $\theta(G)$ und $\alpha(G)$) reduzieren sich für Vergleichbarkeitsgraphen hingegen auf das bipartite Matching Problem. Etwas allgemeiner behandeln wir hier das Problem der Pfadzerlegung eines azyklischen Digraphen.

Pfadzerlegung. Ein *Digraph* (von engl. *di*rected *graph*) sei in unserem Zusammenhang ein gewöhnlicher Graph zusammen mit einer Orientierung seiner Kanten (die gerichteten Kanten heißen nun auch Bögen (engl. arcs)). Eine *Pfadzerlegung* eines Digraphen $D = (V, A)$ ist eine Menge von knotendisjunkten, gerichteten Pfaden (Ketten) in D, so daß jeder Knoten $v \in V$ von genau einem Pfad überdeckt wird (ein isolierter Knoten bildet

dabei einen trivialen Pfad der Länge 0). Beim Optimierungsproblem WEIGHTED PATH PARTITION ist ein kantengewichteter Digraph $D = (V, A, w)$, $w : A \to \mathbb{Q}$, gegeben und eine Pfadzerlegung $\mathcal{P} = \bigcup P$ von D maximalen Gewichts $w(\mathcal{P}) := \sum_{P \in \mathcal{P}} w(P)$ gesucht, wobei das Gewicht eines Pfades $w(P) := \sum_{a \in P} w(a)$ sei.

Eine Pfadzerlegung $\mathcal{P}$ eines Graphen G läßt sich auch als ein spannender Subgraph von G auffassen, dessen sämtliche Komponenten Pfade sind. Insbesondere ist dieser Subgraph also ein Wald auf V, so daß nach Korollar 1.1.27 für die Anzahl der in den Pfaden enthaltenen Kanten gilt:

$$|E(\mathcal{P})| \;=\; n - |\mathcal{P}|. \tag{7.2}$$

Schon in ungewichteten Digraphen (d.h. $w(e) \equiv 1$) ist mithin das Problem, eine Pfadzerlegung mit maximal vielen Kanten zu konstruieren, äquivalent damit, eine Pfadzerlegung mit minimal vielen Pfaden zu finden; das Problem PATH PARTITION ist somit in allgemeinen Digraphen NP-äquivalent (Reduktion von HAMILTONIAN PATH).

Vergleichbarkeitsgraphen sind jedoch transitiv orientierbar, und ein transitiv orientierter Digraph ist insbesondere azyklisch (d.h. es treten keine gerichteten Kreise auf): wenn ein Knoten x in einem gerichteten Kreis läge, so wäre aufgrund der Transitivität der Orientierung auch (x, x) ein Bogen – der zugrundeliegende Vergleichbarkeitsgraph besitzt jedoch keine Schleifen.

Proposition 7.3.9 (FULKERSON 1958; BOESCH, GIMPEL 1977) *In azyklischen Digraphen $D = (V, A, w)$ reduziert sich das Problem* WEIGHTED PATH PARTITION *auf das maximale gewichtete Matching Problem in bipartiten (ungerichteten) Graphen.*

Beweis. Sei $V = \{v_1, \ldots, v_n\}$. Betrachte den bipartiten Hilfgraphen $H = (X, Y, E, w_H)$ mit Teilen X und Y, definiert durch:

$$\begin{aligned} X &:= \{x_i |\; v_i \in V\} \\ Y &:= \{y_i |\; v_i \in V\} \\ E &:= \{\{x_i, y_j\} : (v_i, v_j) \in A\} \\ w_H(\{x_i, y_j\}) &:= w((v_i, v_j)). \end{aligned}$$

Dann entsprechen sich Matchings in H und Subgraphen in D mit Außen- und Innengrad höchstens 1 für alle $v \in V$ bijektiv. Da D azyklisch ist, treten in diesen Subgraphen keine gerichteten Kreise auf, d.h. sie stellen Pfadzerlegungen von D dar. □

Insbesondere gilt offensichtlich für eine Zerlegung $\mathcal{P}_{OPT}$ eines (ungewichteten) azyklischen Digraphen D in minimal viele Pfade:

$$|E(\mathcal{P}_{OPT})| \;=\; \nu(H). \tag{7.3}$$

In einem Vergleichbarkeitsgraphen G induzieren die Knoten eines gerichteten Pfades eine Clique, so daß eine Pfadzerlegung eine Cliquenzerlegung darstellt. Aus (7.2) und (7.3) folgt:

Korollar 7.3.10 *Für einen Vergleichbarkeitsgraphen G gilt*

$$\theta(G) \;=\; n - \nu(H), \tag{7.4}$$

wobei der bipartite Graph H wie oben aus einer transitiven Orientierung von G konstruiert wird. Insbesondere kann eine minimum Cliquenpartition von G mit Hilfe eines bipartiten Matching-Algorithmus gewonnen werden.

Anwendung. Eine Menge von Aufträgen mit festgelegten Anfangs- und Endzeiten ist durch eine minimale Anzahl von Maschinen, von denen jede jeden Auftrag bearbeiten kann, auszuführen. Der gerichtete Graph, der die Aufträge als Knoten hat und bei dem zwei Knoten x und y genau dann durch eine gerichtete Kante (x, y) verbunden sind, wenn der Auftrag x endet, bevor der Auftrag y beginnt, ist transitiv orientiert. Jede Kette von Aufträgen kann von einer Maschine bearbeitet werden. Dann stellt eine Kettenzerlegung mit minimal vielen Ketten eine Lösung des Scheduling-Problems an, die minimal viele Maschinen benötigt. □

Proposition 7.3.11 (FULKERSON 1958) *Das Optimierungsproblem* INDEPENDENT SET *(d.h. insbesondere die Bestimmung von $\alpha(G)$) reduziert sich für Vergleichbarkeitsgraphen auf das bipartite Matching-Problem.*

Beweis. Sei G ein Vergleichbarkeitsgraph und $D = (V, A)$ eine transitive Orientierung von G. Betrachte wie oben den bipartiten Graphen $H = (X, Y, E)$ zu D. Nach dem Satz 4.2.15 von KÖNIG berechnet man mit Hilfe eines Matching-Algorithmus eine minimum Knotenüberdeckung

$$T = \{x_{i_1}, \ldots, x_{i_k}, y_{j_1}, \ldots, y_{j_\ell}\}$$

von H. Setze nun $T' := \{v_{i_1}, \ldots, v_{i_k}, v_{j_1}, \ldots, v_{j_\ell}\}$. Dann gilt:

$$|T'| = |T|, \text{ d.h. alle Knoten in } T' \subseteq V \text{ sind verschieden.}$$

Angenommen, zwei Indizes wären gleich, also o.B.d.A. $i_1 = j_1 =: b$. Da T eine minimale Knotenüberdeckung von H ist, ist x_b zu mindestens einem Knoten $y_c \in Y \setminus T$ und y_b zu mindestens einem Knoten $x_a \in X \setminus T$ in H adjazent. Also gibt es die Kanten (v_a, v_b) und (v_b, v_c) in A. Aus der Transitivität der Orientierung in D folgt nun aber $(v_a, v_c) \in A$, ein Widerspruch, denn dann würde T die Kante $\{x_a, y_c\}$ nicht überdecken.

$$S := V - T' \text{ ist eine maximum unabhängige Knotenmenge in } G.$$

Zunächst ist S tatsächlich unabhängig, denn gäbe es eine Kante $\{v_a, v_b\} \in G[V - T']$ (o.B.d.A. gelte $(v_a, v_b) \in A$), so würde T die Kante $\{x_a, y_b\} \in E(H)$ nicht überdecken. Die Optimalität von S sieht man wie folgt ein. Einerseits ist

$$|S| \;=\; n - |T'| \;=\; n - |T| \;=\; n - \tau(H).$$

Nach dem Satz 4.2.15 von KÖNIG und Gleichung (7.4) gilt aber andererseits

$$\tau(H) \;=\; \nu(H) \;=\; n - \theta(G),$$

und es folgt insgesamt

$$|S| \;=\; \theta(G) \;\geq\; \alpha(G).$$ □

Übungen

Übung 7.3.12 a) *Leite den Satz 4.2.15 von* KÖNIG *(in seiner dualen Form aus Korollar 4.2.19) aus dem Satz 7.3.6 von* DILWORTH *ab.*
b) *Leite den Satz von* DILWORTH *aus dem Satz von* KÖNIG *ab.*

Übung 7.3.13 (SPERNER 1928) *Die maximale Anzahl von Teilmengen einer n-elementigen Menge, so daß keine Teilmenge in einer anderen enthalten ist, ist $\binom{n}{\lceil n/2 \rceil}$.*
[*Hinweis:* Sätze von DILWORTH und HALL]

7.4 Intervallgraphen

Definition 7.4.1 (Hajós 1957) *Ein Graph G heißt* Intervallgraph, *wenn er Schnittgraph einer Familie von abgeschlossenen Intervallen der reellen Zahlengeraden ist.*

Intervallgraphen treten beispielsweise beim Problem der Macro-Substitution auf, wo eine Byte-Folge dadurch komprimiert werden soll, daß häufig auftretende Teilstücke durch Schlüssel ersetzt werden. Wenn sich zwei Byte-Sequenzen überlappen, darf offenbar nur eine von beiden ersetzt werden. Als eine weitere Anwendung kann man sich das Problem der Platzreservierungen für Zugreisende denken: eine möglichst gute Auslastung der Sitzplätze entspricht einer Färbung mit minimal vielen Farben; In der Praxis ist dieses Problem jedoch „online“ zu lösen. Für weitere Anwendungen von Intervallgraphen siehe [Rob78, Gol80, DKL87].

Die Perfektheit dieser Graphenklasse folgt sofort aus der folgenden Beobachtung.

Proposition 7.4.2 *Intervallgraphen sind chordal.*

Beweis. Sei G ein Intervallgraph und $I_1, \ldots, I_n$ eine Schnittgraphdarstellung von G durch Intervalle $I_i = [a_i, b_i] \subset \mathbb{R}$, $i = 1, \ldots, n$. Wir geben eine Schnittgraphdarstellung wie in Satz 7.2.19 an. Der Baum B ist einfach ein Pfad auf der Knotenmenge $V(B) := \{a_1, \ldots, a_n\} \cup \{b_1, \ldots, b_n\}$ und der Teilbaum T_i für das Intervall I_i der Pfad auf den Knoten $\{v \in V(T) : a_i \leq v \leq b_i\}$. □

Der Beweis macht insbesondere deutlich, wie speziell Intervallgraphen in der Klasse der chordalen Graphen sind.

Übung 7.4.3 a) *Man zeige anhand der Definition, daß ein Intervallgraph chordal ist.*
b) *Für eine Intervalldarstellung $[a_i, b_i]$, $i = 1, \ldots, n$, eines Graphen G zeige man, daß die Anordnung der Intervalle aufsteigend sortiert nach b_i ein perfektes Knoteneleminationsschema darstellt.*
c) *Man zeige anhand der Definition, daß Intervallgraphen perfekt sind.*

Die Perfektheit der Komplemente von Intervallgraphen ergibt sich aus der folgenden Eigenschaft.

Übung 7.4.4 (Ghouila-Houri 1962)
Das Komplement eines Intervallgraphen ist ein Vergleichbarkeitsgraph.

Tatsächlich sind Intervallgraphen dadurch schon charakterisiert, chordal und Komplemente von Vergleichbarkeitsgraphen zu sein:

Satz 7.4.5 (Gilmore, Hoffman 1964) *Für einen Graphen G sind äquivalent:*

(i) G ist ein Intervallgraph;

(ii) G enthält keinen induzierten C_4 und $\bar{G}$ ist ein Vergleichbarkeitsgraph;

(iii) Die maximalen Cliquen von G können linear geordnet werden, so daß für jeden Knoten $v \in V$ die maximalen Cliquen, die v enthalten, aufeinanderfolgen.

Die Äquivalenz $(i) \Leftrightarrow (iii)$ wurde auch von FULKERSON und GROSS [FG65] bewiesen.
Beweis. □

Während Intervallgraphen als chordale Graphen und Komplemente von Vergleichbarkeitsgraphen natürlich in polynomieller Zeit erkannt werden können, entwickelten BOOTH und LUEKER [BL76] aufbauend auf (iii) einen linearen Algorithmus, um Intervallgraphen zu erkennen (siehe auch [Gol80, KM89]). Ihre Implementierung benutzt jedoch (wie auch ihr Planaritätstest) die Datenstruktur der PQ-Bäume. Ohne sie kommen SIMON [Sim91] und HSU [Hsu93] aus.

Die Tatsache, daß Intervallgraphen mehr Struktur als allgemeine chordale oder Vergleichbarkeits-Graphen tragen, schlägt sich auch komplexitätstheoretisch nieder. So ist beispielsweise das HAMILTONIAN CIRCUIT Problem - selbst für chordale Graphen [CS85] oder bipartite Graphen [Kri75] NP-vollständig - auf Intervallgraphen polynomiell lösbar [Kei85]. Dasselbe gilt für das Problem DOMINATING SET, siehe [Dew83] bzw. [BJ82].

Übung 7.4.6 The BERGE mystery story
Sechs Professoren waren in der Bibliothek gewesen an dem Tag, an dem das wertvolle Traktat gestohlen wurde. Jeder von ihnen hatte die Bibliothek betreten, war für eine Zeit lang geblieben und dann wieder gegangen. Wenn immer zwei zur gleichen Zeit in der Bibliothek gewesen waren, hatte zumindest einer von beiden den anderen im Blick. Der Inspektor befragte die Professoren, die folgende Angaben machten: Abe berichtete, Burt und Eddie in der Bibliothek gesehen zu haben; Burt sagte, er hätte Abe und Ida gesehen; Charlotte behauptete, Desmond und Ida begegnet zu sein, während Desmond widerum Abe und Ida gesehen haben wollte; Eddie bezeugte, er hätte Burt und Charlotte gesehen; Ida schießlich gab zu Protokoll, Charlotte und Eddie getroffen zu haben.
Einer der Professoren log! Wer war's?

7.5 Graphen mit perfekter Ordnung

Lemma 7.5.1 *Graphen mit perfekter Ordnung sind perfekt.*

Satz 7.5.2 (CHVÁTAL 1984) *Sei $G = (V, E)$ ein Graph. Eine Ordnung auf V ist perfekt genau dann, wenn sie hindernisfrei ist.*

Beweis. □

Korollar 7.5.3 *Chordale und Vergleichbarkeitsgraphen besitzen eine perfekte Ordnung.*

Satz 7.5.4 (CHVÁTAL 1984) *Graphen mit perfekter Ordnung sind streng perfekt.*

Beweis. □

Satz 7.5.5 (MIDDENDORF, PFEIFFER 1990) *Das Erkennungsproblem für Graphen mit perfekter Ordnung ist* NP*-vollständig.* □

HOÀNG [Hoa94] gab $\mathcal{O}(nm)$-Algorithmen für das Färbungproblem und das Maximum Clique Problem in Graphen mit perfekter Ordnung an.

7.6 Im Umfeld der Perfekte-Graphen-Vermutung

BERGE hatte den Begriff der perfekten Graphen auf einer Tagung 1961 im Zusammenhang mit zwei Vermutungen eingeführt, die dieses Gebiet zum Gegenstand intensiver Forschung gemacht haben. Die erste Vermutung zählt heute zu den prominentesten in der Diskreten Mathematik.

Die starke Perfekte-Graphen-Vermutung (SPGC)
Ein Graph ist perfekt genau dann, wenn weder er noch sein Komplement einen ungeraden induzierten Kreis der Länge ≥ 5 besitzt.

Graphen ohne induzierten C_{2l+1}, $\overline{C_{2l+1}}$, $l \geq 2$, werden BERGE zu Ehren auch BERGE-*Graphen* genannt. Wie aus Übung 7.1.3 hervorgeht, ist jeder perfekte Graph trivialerweise ein BERGE-Graph.
Die SPGC wurde bereits u.a. für folgende Graphenklassen bewiesen:

- planare [Tuc73],
- $K_{1,3}$-freie [PR76],
- $(K_4 - e)$-freie [PR79]
- K_4-freie [Tuc84]
- $\gamma(G) \leq 1$ oder $\Delta(G) \leq 6$ [Gri84]

Darüber hinaus konnte gezeigt werden, daß Gegenbeispiele höchstens eine Nullmenge in der Menge aller BERGE-Graphen darstellen können:

Satz 7.6.1 (PRÖMEL, STEGER 1992a) *Fast alle* BERGE-*Graphen sind perfekt.*

Die Situation bei den bipartiten und chordalen Graphen regte BERGE zu seiner zweiten Vermutung an, die besagt, daß die Klasse der perfekten Graphen unter Komplementbildung abgeschlossen ist. Weil deren Richtigkeit aus Symmetriegründen sofort aus der der SPGC folgt, wird sie entsprechend die schwache Perfekte-Graphen-Vermutung genannt. Sie wurde von FULKERSON [Ful71] und LOVÁSZ [Lov72a] bewiesen und wird heute zitiert als der

Satz 7.6.2 (Perfekte-Graphen-Satz)
Ein Graph ist genau dann perfekt, wenn sein Komplement perfekt ist.

Der Beweis dieses Satzes beruht auf einem auch für sich interessanten Lemma. Unter dem Ersetzen eines Knotens $x \in V(G)$ durch einen Graphen H wollen wir den Graphen verstehen, der entsteht, wenn in G der Knoten x durch den Graphen H ersetzt und jeder Knoten von H zudem mit allen vorherigen Nachbarn von x in G verbunden wird.

Lemma 7.6.3 (Ersetzungslemma) (LOVÁSZ 1972)
Wenn man in einem perfekten Graphen G Knoten durch perfekte Graphen ersetzt, so ist der resultierende Graph G^* ebenfalls perfekt.

Beweis. Nach Lemma 7.1.8 ist eine stabile Menge S in G^* zu konstruieren, die alle maximum Cliquen in G^* trifft. Sei S_G diejenige Farbklasse in einer $\chi(G)$-Färbung von G, die x

enthält, und S_H eine stabile Menge in H, die alle maximum Cliquen in H trifft (existiert nach Lemma 7.1.8). Dann ist

$$S := (S_G - x) \cup S_H$$

stabil in G^*. Sei C eine beliebige maximum Clique in G^*. Falls $C \cap V(H) = \emptyset$, so ist C auch eine maximum Clique in $G - x$. Wegen $\omega(G) \geq \chi(G)$ enthält daher C mindestens einen Knoten aus der Farbklasse S_G. Falls andernfalls $C \cap V(H) \neq \emptyset$, so muß C eine maximum Clique von H enthalten (sonst wäre C nicht maximum in G^*), und daher trifft S_H die Clique C. □

Beweis von Satz 7.6.2:
Es genügt offenbar, „⇒" zu zeigen. Wir induzieren über n. Für $n \leq 3$ ist die Aussage trivial. Sei G also ein perfekter Graph mit $n \geq 4$. Zu zeigen ist, daß auch sein Komplement $\overline{G}$ perfekt ist. Da auch alle echten induzierten Subgraphen von G perfekt sind, sind nach Induktionsvoraussetzung bereits alle echten induzierten Subgraphen von $\overline{G}$ perfekt, und es genügt, $\chi(\overline{G}) = \omega(\overline{G})$ oder genauer $\chi(\overline{G}) \leq \omega(\overline{G})$ zu zeigen. Seien $C_1, \ldots, C_k$ die inklusionsmaximalen Cliquen von G. Wir unterscheiden
Fall 1: Es gibt ein C_j, $1 \leq j \leq k$, das alle maximum stabilen Mengen in G schneidet.
Dann gilt wie in Lemma 7.1.8:

$$\chi(\overline{G}) \leq \chi(\overline{G - C_j}) + 1 \overset{(IV)}{=} \omega(\overline{G - C_j}) + 1 \leq \omega(\overline{G}).$$

Fall 2: Für alle $1 \leq j \leq k$ enthält $V \setminus C_j$ eine stabile Menge S_j in G mit $|S_j| = \alpha(G)$.
Für einen Knoten $x \in V(G)$ bezeichne $s(x)$ die Anzahl der maximum stabilen Mengen S_i, $1 \leq i \leq k$, die x enthalten:

$$s(x) := |\{i \mid 1 \leq i \leq k \wedge x \in S_i\}|.$$

Ersetze in G jeden Knoten $x \in V(G)$ durch einen $K_{s(x)}$ und erhalte einen, nach Lemma 7.6.3 wiederum perfekten Graphen G^*. Dann entsprechen sich die maximum Cliquen in G und G^* bijektiv und für die Cliquenzahl von G^* gilt

$$\omega(G^*) = \max_{1 \leq j \leq k} \sum_{x \in C_j} s(x).$$

Da eine Clique und eine stabile Menge sich nur in höchstens einem Knoten schneiden können, wird für jedes j in der Summe $\sum_{x \in C_j} s(x)$ jede stabile Menge S_i, $1 \leq i \leq k$, nur höchstens einmal gezählt. Außerdem gibt es nach Annahme für jede Clique C_j in G eine maximum stabile Menge $S_j \subseteq V \setminus C_j$, die folglich in $\sum_{x \in C_j} s(x)$ nicht mitgezählt wird. Also gilt

$$\omega(G^*) = \max_{1 \leq j \leq k} \sum_{x \in C_j} s(x) \leq k - 1.$$

Andererseits gilt wegen $|V(G^*)| = \sum_{i=1}^{k} |S_i| = k \cdot \alpha(G)$ und $\alpha(G^*) = \alpha(G)$:

$$\chi(G^*) \geq \frac{|V(G^*)|}{\alpha(G^*)} = \frac{k \cdot \alpha(G)}{\alpha(G)} = k$$

im Widerspruch zur Perfektheit von G^*. □

Angewandt auf spezielle perfekte Graphenklassen, verallgemeinert dieser Satz so unterschiedliche Ergebnisse wie die Sätze 4.2.15 und 4.2.14 von KÖNIG oder den Satz 7.3.6 von DILWORTH.

Übung 7.6.4 (SEINSCHE 1974) *Graphen ohne induzierten P_4 sind perfekt.*
[*Hinweis:* G oder $\overline{G}$ ist unzusammenhängend]

Um die SPGC zu beweisen, müßte die Perfektheit aller BERGE-Graphen nachgewiesen werden. Ein Ansatz zum Beweis der SPGC besteht daher darin, der BERGE-Bedingung immer schwächere Bedingungen hinzuzufügen, die noch die Perfektheit der dadurch definierten Graphenklasse garantieren.
Beispielsweise sind die chordalen Graphen (deren Perfektheit in Satz 7.2.6 nachgewiesen wurde), BERGE-Graphen: nach Definition enthalten sie überhaupt keinen induzierten Kreis der Länge ≥ 4, also insbesondere keine sehnenlosen C_{2k+1}, $k \geq 2$; wegen $\overline{C_5} = C_5$ und weil das Komplement eines jeden Kreises der Länge ≥ 6 einen C_4 als induzierten Subgraphen hat, enthalten chordale Graphen aber auch keinen induzierten $\overline{C_{2k+1}}$, $k \geq 2$.
Graphen ohne induzierte Kreise der Länge ≥ 5 und deren Komplemente heißen *schwach chordal*. Chordale Graphen sind also insbesondere schwach chordal. Schwach chordale Graphen sind perfekt und können in polynomieller Zeit erkannt werden [Hay85, SS95].

MEYNIEL [Mey76] bewies, daß alle Graphen, in denen nur jeder ungerade Kreis der Länge ≥ 5 mindestens zwei Sehnen hat (solche Graphen heißen ihm zu Ehren heute MEYNIEL-*Graphen*), schon perfekt sind. (Einen eleganten Beweis hierzu gab LOVÁSZ [Lov83].) RAVINDRA [Rav82] zeigte, daß auch MEYNIEL-Graphen stark perfekt sind; tatsächlich sind MEYNIEL-Graphen genau die *sehr stark perfekten* Graphen, siehe [Hoa87].

Definition 7.6.5 *Ein Graph G heißt kritisch imperfekt oder* minimal imperfekt, *falls G selbst zwar nicht perfekt ist, aber alle seine echten induzierten Subgraphen.*

Die SPGC läßt sich damit offenbar äquivalent auch so formulieren, daß die einzigen minimal imperfekten Graphen die ungeraden Kreise und ihre Komplemente sind. Ein Ansatz zum Beweis der SPGC besteht daher darin, soviele Eigenschaften minimal imperfekter Graphen herauszufinden, daß schließlich nur noch die ungeraden Kreise und deren Komplemente übrigbleiben.

LOVÁSZ konnte perfekte Graphen wie folgt charakterisieren.

Satz 7.6.6 (LOVÁSZ 1972) *Ein Graph G ist perfekt genau dann, wenn für alle induzierten Subgraphen H von G gilt:*

$$\omega(H) \cdot \omega(\bar{H}) \;\geq\; |H|.$$

Da die Bedingung offenbar symmetrisch in G und $\overline{G}$ ist, folgt aus dieser Charakterisierung perfekter Graphen insbesondere sofort der Perfekte-Graphen-Satz. Satz 7.6.6 läßt sich auch ganz analog wie der Perfekte-Graphen-Satz beweisen, siehe z.B. [Hal89]. Wir folgen deshalb ZAREMBA [Zar91] und zeigen, daß diese Charakterisierung perfekter Graphen äquivalent zur klassischen SCHWARTZ-Ungleichung für bestimmte Vektoren des $\mathbb{R}^n$ unter einer passenden Norm ist.
Beweis. □

Definition 7.6.7 *Ein Graph $G = (V, E)$ der Ordnung $|V| = \alpha\omega + 1$ mit $\alpha, \omega \in \mathbb{N}^{\geq 2}$ heißt* (α, ω)-partitionierbar, *wenn für alle $v \in V$ der Subgraph $G - v := G[V \setminus \{v\}]$ mit α vielen ω-Cliquen und genauso mit ω vielen α-stabilen Mengen (das seien stabile Mengen in G der Kardinalität α) überdeckt werden kann:*

$$\begin{aligned} \chi(\overline{G - v}) &= \alpha, \\ \chi(G - v) &= \omega. \end{aligned}$$

D.h. die Parameter α und ω sind für einen partitionierbaren Graphen eindeutig bestimmt.

Lemma 7.6.8 *Für einen (α,ω)-partitionierbaren Graphen G gilt:*

$$\begin{array}{lrcl} (i) & \alpha(G) & = & \alpha, \\ (ii) & \omega(G) & = & \omega, \\ (iii) & \chi(\overline{G}) & = & \alpha+1, \\ (iv) & \chi(G) & = & \omega+1. \end{array}$$

Beweis. □

Satz 7.6.9 (Padberg 1974) *Ein (α,ω)-partitionierbarer Graph G der Ordnung n hat folgende Eigenschaften:*

- (i) *G hat genau n ω-Cliquen.*
- (ii) *G hat genau n α-stabile Mengen.*
- (iii) *Jeder Knoten $v \in V$ ist in genau ω ω-Cliquen enthalten.*
- (iv) *Jeder Knoten $v \in V$ ist in genau α α-stabilen Mengen enthalten.*
- (v) *Jede ω-Clique schneidet genau eine α-stabile Menge nicht.*
- (vi) *Jede α-stabile Menge schneidet genau eine ω-Clique nicht.*

Beweis. Für eine α-stabile Menge $S_0 = \{s_1,\ldots,s_\alpha\}$ sei $\{S_{i1},\ldots,S_{i\omega}\}$ eine Partition von $G - s_i$ in α-stabile Mengen und S die $n \times n$ Matrix mit den Inzidizenzvektoren der Mengen $S_0, S_{11},\ldots,S_{\alpha\omega}$ bezüglich V in den Spalten. Analog erhalte eine $n \times n$ Matrix C mit Inzidenzvektoren von ω-Cliquen in den Spalten. J bezeichne die 1-Matrix. Formuliere $(i)-(vi)$ in der Sprache der Matrixalgebra. □

Übung 7.6.10 *Die Bedingungen (i) - (vi) aus Satz 7.6.9 sind auch hinreichend für die (α,ω)-Partitionierbarkeit eines Graphen G.*

Proposition 7.6.11 (Lovász 1972b) *Für einen minimal imperfekten Graphen G der Ordnung n gilt:*

- $n = \alpha(G)\omega(G)+1$;
- *G ist $(\alpha(G),\omega(G))$-partitionierbar.*

Beweis.

□

Aus Proposition 7.6.11 leitet sich immerhin ab, daß es für *nicht* perfekte Graphen ein Zertifikat gibt, das man in (in $|G|$) polynomieller Zeit überprüfen kann.

Korollar 7.6.12 Perfect Graph Recognition $\in$ *co-NP.*

Beweis.

□

Beachte, daß dieses Korollar unmittelbar aus der Richtigkeit der SPGC folgen würde.

Weitere Eigenschaften eines partitionierbaren Graphen G sind u.a.: $G-v$ ist eindeutig färbbar für alle $v \in V$ [BHT79], und G ist $(2\omega-2)$−zusammenhängend [Seb93].

Übung 7.6.13 (Star-Cutset-Lemma) (Chvátal 1985) *Kein minimal imperfekter Graph besitzt ein Star-Cutset, d.h. eine trennende Knotenmenge $S \subseteq V$, die einen Knoten $s \in S$ enthält, der zu allen anderen Knoten in S benachbart ist.*

7.7 Optimierung auf perfekten Graphen

Kapitel 8

Separatoren

8.1 Einleitung

Die "divide and conquer" Methode, klassisch auch "divide et impera" genannt, hat sich als nützlich zur Lösung zahlreicher Optimierungsprobleme erwiesen. Wie der Name es schon andeutet, besteht das Verfahren aus zwei Phasen. In der ersten Phase wird das Ausgangsproblem in zwei oder mehrere kleinere Teilprobleme unterteilt, die dann rekursiv mit derselben Methode gelöst werden. In der zweiten Phase werden die Lösungen der Unterprobleme zu einer Lösung des Ausgangsproblems zusammengefügt. Dieses Verfahren geht also eher „top down" vor im Gegensatz zu dem verwandten Paradigma der dynamischen Programmierung, das einen „bottom up"-Ansatz darstellt (vergleiche Abschnitte 1.3.2, 1.5). Voraussetzung für die Anwendbarkeit der "divide and conquer" Methode ist also, daß sich jede Probleminstanz entweder in (noch) kleinere Teilprobleme zerlegen läßt oder schließlich so klein ist, daß sie mit vertretbarem Aufwand (einfach) gelöst werden kann.[1]

Für Graphenprobleme stellen Separatoren einen geeigneten Teilungsbegriff dar. Ein Separator in einem Graphen ist eine Teilmenge der Knotenmenge des Graphen, nach deren Entfernung aus dem Graphen dieser in mehrere Komponenten möglichst kleiner Größe zerfällt. Dabei sind wir insbesondere daran interessiert, Graphenklassen zu bestimmen, für die es beispielsweise Separatoren der Größe $o(n)$ gibt. In diesem Kapitel wird ein Separatorensatz für planare Graphen vorgestellt, der im Jahre 1979 von LIPTON und TARJAN bewiesen wurde und die Forschung auf dem Gebiet der Separatoren initiierte. Aus dem Beweis werden wir die Existenz eines linearen Algorithmus zur Berechnung eines solchen Separators ableiten. Danach werden wir Separatoren für andere Graphenklassen und insbesondere Cliquenseparatoren, d.h. Separatoren, deren Knoten eine Clique bilden, betrachten. Außerdem zeigen wir, wie Separatoren bei der Lösung NP-schwerer Probleme, wie beispielsweise INDEPENDENT SET weiterhelfen können.

Definition 8.1.1 *Sei $0 < \alpha < 1$ eine Konstante und $f(n)$ eine Funktion. Dann ist ein* $(f(n), \alpha)$-Separator *in einem Graphen $G = (V, E)$ auf n Knoten eine Menge $C \subseteq V$ der Größe $|C| \leq f(n)$, so daß jede Komponente von $G[V \setminus C]$ höchstens αn Knoten enthält.*

Von einigen Autoren wird eine andere Definition bevorzugt, die eine Aufteilung in zwei

[1] Ein Gegenbeispiel wäre ein kleines gallisches Dorf, das einerseits intern eine verschworene Gemeinschaft bildet, die jede Teilung unmöglich macht, und das andererseits hartnäckig einer Eingliederung in das Imperium (Romanum) widerstrebt.

Mengen vorsieht, die untereinander durch keine Kante verbunden sind. Für den Fall $\alpha = \frac{2}{3}$ sind diese beiden Definitionen äquivalent:

Proposition 8.1.2 *Ein Graph G besitzt einen $(f(n), \frac{2}{3})$*-Separator *genau dann, wenn V in drei Mengen A, B und C partitioniert werden kann, so daß $|C| \leq f(n)$, $|A|, |B| \leq \frac{2}{3}n$, und es keine Kanten zwischen A und B gibt.*

Beweis. „$\Leftarrow$“: Da jede Komponente von $G[V \setminus C]$ entweder in A oder in B enthalten ist, kann sie höchstens $\frac{2}{3}n$ Knoten besitzen.
„$\Rightarrow$“: Falls es eine Komponente von $G[V \setminus C]$ gibt, die mindestens $\frac{1}{3}n$ Knoten enthält, sei A diese Komponente und B die Vereinigung der übrigen Komponenten. Wenn alle Komponenten $G - C$ weniger als $\frac{1}{3}n$ Knoten enthalten, dann vereinige sukzessive Komponenten zu einer Menge A, bis deren Größe erstmalig $\frac{1}{3}n$ überschreitet. □

Diese Proposition gilt offensichtlich für jedes $\alpha \geq \frac{2}{3}$. Für kleinere α wird sie jedoch falsch, wie das folgende Beispiel es zeigt. Sei T_k der Baum, der durch Vereinigung dreier Pfade der Länge k entsteht, die an einem ihrer Endknoten identifiziert werden. T_k besitzt offenbar einen $(1, \alpha)$-Separator für jedes $\alpha \geq \frac{1}{3}$, aber es gibt kein $\alpha < \frac{2}{3}$, so daß die Bäume T_k für k groß genug durch Entfernen eines Knotens in zwei nichtadjazente Mengen A und B der Größe höchstens $\alpha n(T_k)$ separiert werden könnten.

Da es Graphen gibt, die offensichtlich keinen $o(n)$-Separator besitzen, erhebt sich die Frage, ob es interessante Graphenklassen mit kleinen Separatoren gibt. Als ein einfaches Beispiel betrachten wir Bäume.

Proposition 8.1.3 *Jeder Baum besitzt einen $(1, \frac{1}{2})$-Separator.*

Beweis. Sei $T = (V, E)$ ein Baum auf n Knoten. Wähle einen beliebigen Knoten $c \in V$. Falls jede Komponente von $T - c$ höchstens $\frac{1}{2}n$ Knoten enthält, sind wir fertig. Ansonsten sei T' die eindeutige Komponente von $T - c$ mit mehr als $\frac{1}{2}n$ Knoten. Betrachte den Nachbarn c' von c in T'. Da $V(T) \setminus V(T')$ höchstens $\frac{1}{2}n$ viele Knoten enthält, enthalten auch die zu T' disjunkten Komponenten von $T - c'$ höchstens $\frac{1}{2}n$ viele Knoten. Die Größe der größten Komponente hat also beim Übergang von $T - c$ zu $T - c'$ um mindestens 1 abgenommen. Wenn man dieses Vorgehen iteriert, ist nach spätestens $n - 1$ Schritten ein $(1, \frac{1}{2})$-Separator c gefunden. □

Es läßt sich zeigen, daß die Bestimmung der Größe eines minimalen Separators C, nach dessen Entfernung zwei nichtadjazente Teile der Größe höchstens $\frac{1}{2}n$ verbleiben, sogar für reguläre Graphen mit festem Grad, NP-schwer ist, vgl. [Len90, Exercise 6.2]. Für planare Graphen ist die Komplexität dieses Problems nicht bekannt.

Übung 8.1.4 (LEISERSON 1980) *Jeder zweifach zusammenhängende, kreisförmig planare Graph besitzt einen $(2, \frac{1}{2})$-separator.* □

8.2 Separatoren in planaren Graphen

Da ein planarer Graph G einen Knoten vom Grad höchstens 5 besitzt, enthält also jeder planare Graph auf mindestens 5 Knoten eine trennende Menge der Größe höchstens 5. Falls wir aber fordern, daß die Komponenten etwa gleich groß sein sollen, dann wird die Aufgabe, einen geeigneten Separator zu finden, deutlich schwieriger. Wir werden den folgenden Satz beweisen.

Satz 8.2.1 (LIPTON, TARJAN 1979)
Jeder planare Graph auf n Knoten besitzt einen $(2\sqrt{2n}, \frac{2}{3})$-Separator.

Zunächst zeigen wir, daß die Größe des Separators bis auf eine multiplikative Konstante bestmöglich ist.

Proposition 8.2.2 [LT79] *Sei $G_{k\times k} = (V, E)$ der $k \times k$-Gittergraph, dessen $n = k^2$ Knoten den Gitterpunkten eines $k \times k$-Gitters entsprechen, wobei genau die benachbarten Gitterpunkte verbunden sind. Dann besitzt G für kein $0 < \alpha < 1$ und für kein $\beta < \min\{\frac{1}{2}, \sqrt{1-\alpha} - \frac{1}{2\sqrt{n}}\}$ einen $(\beta\sqrt{n}, \alpha)$-Separator.*

Beweis. Sei $C \subseteq V$ eine trennende Knotenmenge, so daß alle Komponenten von $G[V\backslash C]$ höchstens αn Knoten enthalten. Wir zeigen, daß dann $|C| > \beta\sqrt{n}$.

Zunächst bestimmen wir eine Menge $A \subseteq V\backslash C$ der Größe $\beta^2 n < |A| \leq \frac{1}{2}n$ mit der Eigenschaft, daß alle Nachbarn von A in G in $C \cup A$ enthalten sind. Falls eine Komponente B von $G[V\backslash C]$ mindestens die Größe $\frac{1}{2}n$ hat, dann definiere $A := V\backslash(B \cup C)$. Nach der Definition von β gilt dann $\frac{1}{2}n \geq |A| \geq n - \alpha n - \beta\sqrt{n} > \beta^2 n$. Andernfalls definiere A so als Vereinigung von Komponenten von $G[V\backslash C]$, daß $\frac{1}{4}n < |A| \leq \frac{1}{2}n$ gilt.

Sei z die Anzahl von Zeilen in G, die mindestens einen Knoten von A enthalten, und s die entsprechende Anzahl von Spalten. Wir können annehmen, daß $s \leq z$. Sei ferner z^* die Anzahl der Zeilen in G, die nur Knoten aus A enthalten. Offenbar muß C mindestens $z - z^*$ Knoten enthalten. Falls $z^* > 0$, dann ist wegen $s = k$ auch $z = k$. Aus $k \cdot z^* \leq |A| \leq \frac{1}{2}n$, also $z^* \leq \frac{1}{2}k$, folgt daher $|C| \geq z - z^* \geq k - \frac{1}{2}k > \beta\sqrt{n}$. Im Fall $z^* = 0$ folgt wegen $\beta^2 n < |A| \leq sz \leq z^2$ ebenfalls $|C| \geq z > \beta\sqrt{n}$. □

Aus der Proposition folgt also beispielsweise, daß es in planaren Graphen im allgemeinen keinen $((\frac{1}{2} - o(1))\sqrt{n}, \frac{2}{3})$-Separator gibt. DJIDJEV [Dji82] zeigte sogar die Nichtexistenz von $((\frac{1}{3} - \epsilon)\sqrt{4\pi\sqrt{3}}, \frac{2}{3})$-Separatoren für alle $\epsilon > 0$ in allgemeinen planaren Graphen. Die beste bekannte obere Schranke stammt von ALON, SEYMOUR und THOMAS [AST94], die zeigten, daß jeder planare Graph einen $(\frac{3}{2}\sqrt{2n}, \frac{2}{3})$-Separator besitzt. In Dezimaldarstellung haben diese Konstanten die Werte $\frac{1}{3}\sqrt{4\pi\sqrt{3}} \approx 1,555$ und $\frac{3}{2}\sqrt{2} \approx 2,121$.

Das nun folgende Lemma spielt eine zentrale Rolle im Beweis von Satz 8.2.1.

Lemma 8.2.3 *Sei G ein planarer Graph und $B = (V, E(B))$ ein spannender Baum in G vom Radius s. Dann besitzt G einen $(2s+1, \frac{2}{3})$-Separator C.*

Beweis. Sei $G = (V, E)$ in der Ebene eingebettet und o.B.d.A. trianguliert. Sei $e \in E\backslash E(B)$ eine Kante, die keine Baumkante ist. e bildet mit den Kanten aus $E(B)$ einen eindeutig bestimmten Kreis, den wir mit $C(e)$ bezeichnen. Die Länge eines solchen Kreises beträgt offenbar höchstens $2s + 1$. Wir bezeichnen für einen Kreis C in G mit $int(C)$ und $ext(C)$ die Anzahl der Knoten im Inneren beziehungsweise im Äußeren von C. Wir suchen nun eine Kante e, so daß sowohl $int(C(e)) \leq \frac{2}{3}n$ als auch $ext(C(e)) \leq \frac{2}{3}n$ gilt, denn dann ist die Knotenmenge von $C(e)$ eine trennende Menge mit den gewünschten Eigenschaften.

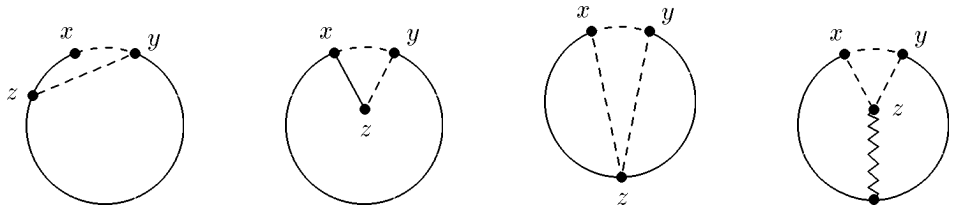
Wähle also eine beliebige Kante $e = \{x, y\} \in E\backslash E(B)$. Wir können annehmen, daß $int(C) \geq ext(C)$ gilt, wobei $C = C(e)$. Wir zeigen nun, wie wir, solange $int(C) > \frac{2}{3}n$ ist, konstruktiv eine zu e adjazente Kante e' wählen können, so daß das folgende gilt.

- Das Innere von $C(e')$ ist im Inneren von $C(e)$ enthalten und umfaßt mindestens eine Fläche weniger.

- $ext(C(e')) \leq \frac{2}{3}n$

Beim Ersetzen von $C(e)$ durch $C(e')$ muß zwar $int(C)$ nicht notwendigerweise abnehmen, aber da $int(C)$ nur $\mathcal{O}(n)$ viele Flächen enthalten kann, terminiert dieser Algorithmus nach linear vielen Schritten und gibt C als den gesuchten Separator aus.

Um e' zu bestimmen, betrachte das Dreick im Inneren von C, das die Kante $\{x, y\}$ enthält (vergleiche die untenstehende Abbildung). Sei z der dritte Knoten dieses Dreicks. Wegen $int(C) > 0$ können nicht sowohl $\{x, z\}$ als auch $\{y, z\}$ Kanten aus C sein.



Fall 1 Fall 2a Fall 2b' Fall 2b"

Durchgezogene Kanten sind Baumkanten, gestrichelte Nicht-Baumkanten.

Wir unterscheiden nun die folgenden Fälle:
Fall 1. Genau eine der Kanten $\{x, z\}$ und $\{y, z\}$ ist in C enthalten. Sei $\{x, z\}$ diese Kante. Dann liegt z auf C, und die Kante $\{y, z\}$ ist keine Baumkante. Der Kreis $C(\{y, z\})$ enthält im Innern die gleichen Knoten wie C, aber er umfaßt eine Fläche weniger. Wegen $int(C) > \frac{2}{3}n$ gilt $ext(C(\{y, z\})) \leq \frac{2}{3}n$, und daher setzen wir $e' := \{y, z\}$.
Fall 2. Weder $\{x, z\}$ noch $\{y, z\}$ ist in C enthalten. Dann können sie nicht beide Baumkanten sein.
Fall 2a. Genau eine der beiden Kanten ist Baumkante; sei dies $\{x, z\}$. Dann induziert $\{y, z\}$ einen Kreis $C(\{y, z\})$, dessen Inneres alle Knoten aus dem Inneren von $C(\{x, y\})$ bis auf den Knoten z enhält, und wir können wieder $e' := \{y, z\}$ wählen.
Fall 2b. Weder $\{x, z\}$ noch $\{y, z\}$ ist Baumkante. Sei ohne Beschränkung der Allgemeinheit $int(C(\{x, z\})) \geq int(C(\{y, z\}))$. Wir ersetzen e durch $e' := \{x, z\}$. Dabei nimmt die Anzahl der Flächen im Inneren von C mindestens um zwei ab. Ferner gilt

$$\begin{aligned} \frac{2}{3}n \;<\; int(C) \;&=\; int(C(\{x,z\})) + int(C(\{y,z\})) + |C(\{x,z\}) \cap C(\{y,z\})| - 1 \\ &\leq\; 2int(C(\{x,z\})) + |C(\{x,z\})|. \end{aligned} \tag{8.1}$$

Daraus folgt

$$ext(C(\{x,z\})) \;=\; n - |C(\{x,z\})| - int(C(\{x,z\})) \;<\; n - \frac{1}{3}n = \frac{2}{3}n. \qquad \square$$

Beweis von Satz 8.2.1 (nach [Ven87]): Ohne Beschränkung der Allgemeinheit sei G zusammenhängend. Wähle einen beliebigen Knoten $w \in V$ als Wurzel und teile die Knoten von G in Mengen S_i für $i = 1, \ldots, \ell$ auf, wobei S_i alle Knoten vom Abstand i von w enthalte und ℓ der maximale Abstand eines Knotens aus G zu w ist. Falls $2\ell + 1 \leq 2\sqrt{2n}$ garantiert schon das vorherige Lemma die Existenz des gesuchten Separators. Für eine später zu bestimmende natürliche Zahl s gruppieren wir die Mengen S_i nach ihrem Rest modulo s. Seien also

$$L_j = \bigcup \{S_i \mid i \equiv j \bmod s\} \qquad (0 \leq j < s).$$

Für ein k mit $0 \leq k < s$ gilt dann $|L_k| \leq \lfloor n/s \rfloor$. Angenommen, es gibt eine Komponente H von $G[V \backslash L_k]$, die mehr als $\frac{2}{3}n$ Knoten enthält. H ist in höchstens $s-1$ aufeinanderfolgenden Mengen S_i enthalten. Wir zeigen nun, daß H in einen planaren Graphen mit Radius höchstens $s-1$ eingebettet werden kann. Sei dazu $i_{\min}$ minimal mit $S_i \cap V(H) \neq \emptyset$. Falls $i_{\min} = 0$, dann enthält H den Knoten w, und wir sind bereits fertig. Wenn $i_{\min} > 0$, verbinden wir alle Knoten aus $S_{i_{\min}} \cap V(H)$ mit einem neuen Knoten w'; d.h. wir kontrahieren in G die Knoten aus $\bigcup_{i=0}^{i_{\min}-1} S_i$ zu einem Knoten. Der dadurch entstehende Graph ist offenbar planar und hat einen Radius von höchstens $s-1$. Nach dem vorhergehenden Lemma enhält H also einen $(2s-1, \frac{2}{3})$-Separator C'. Wenn wir dann $C := C' \cup L_k$ setzen, ist C ein Separator in G der Größe

$$|C| \;\leq\; \lfloor n/s \rfloor \;+\; 2s - 1.$$

Diese Summe wird für $s = \left\lceil \sqrt{n/2} \right\rceil$ minimal, und dann gilt $|C| \leq \sqrt{2n} + \sqrt{2n}$. □

Aus dem Beweis von Satz 8.2.1 können wir folgern, daß ein Separator sogar in linearer Zeit gefunden kann, was wichtig für Anwendungen des planaren Separatorensatzes ist.

Korollar 8.2.4 (LIPTON, TARJAN 1979) *In einem planaren Graphen G läßt sich in $\mathcal{O}(n)$ Zeit ein $(2\sqrt{2n}, \frac{2}{3})$-Separator konstruieren.*

Beweis. Wir gehen wie folgt vor:
1. Schritt: Berechne die Komponenten von G und die Mengen S_i aus dem Beweis von Satz 8.2.1 mit Hilfe von BREITENSUCHE. Wir können nun annehmen, daß G einen spannenden Baum B der Tiefe $s = \left\lceil \sqrt{n/2} \right\rceil$ besitzt.
2. Schritt: Bestimme für alle Knoten v ihren Vorgänger $p(v)$ in B und die Anzahl $n(v)$ der Nachfahren in T (Kinder, Enkel, etc.).
3. Schritt: Bette G in die Ebene ein.
4. Schritt: Trianguliere G, und halte für jeden Knoten die inzidenten Kanten im Uhrzeigersinn geordnet fest; diese Ordnung hängt natürlich von der gegebenen Einbettung ab.
5. Schritt: Wähle eine beliebige Nicht-Baumkante $e = \{x, y\}$. Bestimme $C(e)$. Dazu genügt es, den ersten Schnittpunkt der beiden Pfade von x beziehungsweise y zur Wurzel von B zu bestimmen. Gehe die Baumkanten auf beiden Seiten von $C(e)$ durch, und benutze $n(v)$, um $int(C(e))$ und $ext(C(e)$ zu berechnen. Dabei können wir annehmen, daß $int(C(e)) \geq ext(C(e)$.
6. Schritt: Solange $int(C(e)) > \frac{2}{3}n$, bestimme e' wie im Beweis von Lemma 8.2.3, und setze $e := e'$.

Es ist leicht einzusehen, daß die Schritte 1,2,4 und 5 nur $\mathcal{O}(n)$ Rechenschritte benötigen. Der 3. Schritt ist ebenfalls in linearer Zeit durchführbar, da es einen linearen Einbettungsalgorithmus für planare Graphen gibt (vgl. Abschnitt 6.2). Die Abschätzung der Laufzeit für den 6. Schritt erfordert genauere Überlegungen. Da G nach der Eulerschen Formel (Satz 6.1.9) nur linear viele Flächen besitzt und sich bei jedem Schleifendurchlauf die Anzahl der Flächen im Inneren von $C(e)$ mindestens um eins erniedrigt, wird die Schleife nur linear oft durchlaufen. Da die Kanten im Uhrzeigersinn geordnet sind, kann in konstanter Zeit entschieden werden, welcher der drei Fälle 1, 2a oder 2b aus Lemma 8.2.3 vorliegt. In den Fällen 1 und 2a können $C(e')$ und $int(C(e'))$ in konstanter Zeit berechnet werden.

Jetzt müssen wir also nur noch die Rechenzeit für alle Operationen zu Fall 2b betrachten. Die Rechenzeit für ein einmalige Anwendung von Fall 2b kann recht groß sein, denn ein

Pfad in B von z zu $C(e)$ kann beliebig lang sein. Wir werden nun zeigen, daß die Anzahl der gelesenen Kanten zu allen Schleifendurchläufen in Schritt 6, die in Fall 2b fallen, höchstens $m+2n = \mathcal{O}(n)$ beträgt. Jede Kante kommt in höchstens einem B-Pfad von einem Knoten z zu $C(e)$ vor, da die Pfadkanten im nächsten Schritt nicht mehr im Inneren von $C(e)$ liegen. Also werden zur Bestimmung von $C(e')$ mit Hilfe der p-Relation höchstens m Kanten gelesen. Um zu entscheiden, ob $int(C(\{x,z\}))$ oder $ext(C(\{y,z\}))$ größer ist, gehen wir wie folgt vor. Es ist klar, daß es genügt, einen dieser Werte zu bestimmen, da wir den anderen Wert dann mit (8.1) berechnen können, wobei $|C(\{x,z\}) \cap C(\{y,z\})|$ gerade die Anzahl der Kanten eines B-Pfades von z zu $C(e)$ ist. Nun lesen wir abwechselnd Baumkanten im Inneren von $C(\{x,z\})$ und $C(\{y,z\})$ und summieren die Nachfolgerfunktionen $n(v)$ auf. Wenn wir alle Baumkanten im Inneren eines der beiden Kreise gelesen haben, können wir mit (8.1) die gesuchten Werte bestimmen. Bei dieser Berechnung werden also höchstens zweimal so viele Knoten gelesen, wie der kleinere der beiden Kreise Kanten enthält. Da dieser Kreis daraufhin entfernt wird, werden diese Knoten später nicht mehr betrachtet. Insgesamt benötigen wir zur Bestimmung aller neuen ext- und int-Werte also höchstens $\mathcal{O}(n)$ Schritte. □

Nun wenden wir uns der naheliegenden Frage zu, wie groß wir den Separator wählen müssen, wenn wir in planaren Graphen Komponenten der Größe höchtens ϵn für ein $\epsilon > 0$ erhalten wollen.

Satz 8.2.5 [Ven87, LT80] *Sei $G = (V,E)$ ein planarer Graph auf n Knoten und $\epsilon \leq 1$ mit $\epsilon n \geq 1$. Dann enthält G einen $((2+\sqrt{\frac{6}{\epsilon n}})\sqrt{6}\sqrt{\frac{n}{\epsilon}}, \epsilon)$-Separator, der in Zeit $\mathcal{O}(n \log \frac{1}{\epsilon})$ bestimmt werden kann.*

Beweis. Definiere $s := \left\lceil \sqrt{\frac{\epsilon n}{6}} \right\rceil$. Wir wenden das folgende Verfahren an, das eine wiederholte Anwendung des Algorithmus aus Korollar 8.2.4 darstellt.

1. Schritt: Benutze wie im Beweis von Satz 8.2.1 BREITENSUCHE von einem beliebigen Knoten aus, um eine Menge L_k der Größe höchstens $\lfloor n/s \rfloor$ zu finden, so daß alle Komponenten von $G[V \backslash L_k]$ in planare Graphen mit Radius höchstens $s-1$ eingebettet werden können. Setze $C := L_k$.

2. Schritt: Solange es eine Komponente H in $G[V \backslash C]$ der Größe mindestens ϵn gibt, bestimme wie in Lemma 8.2.3 beziehungsweise Korollar 8.2.4 eine Menge $C' \subseteq V(H)$ der Größe höchstens $2s+1$, so daß H nach Entfernen von C' in Komponenten der Größe höchstens $\frac{3}{2}|V(H)|$ zerfällt, und setze $C := C \cup C'$.

Um zu zeigen, daß C höchstens die angegebene Größe hat, und um die Laufzeit des Algorithmus abzuschätzen, partitionieren wir alle Komponenten, die im Laufe des Algorithmus auftreten, folgendermaßen in Klassen $\mathcal{N}_i$. Sei

$$\mathcal{N}_0 \text{ die Menge aller Komponenten der Größe } \leq \epsilon n$$

und für $1 \leq i \leq \log_{3/2} \frac{1}{\epsilon}$

$$\mathcal{N}_i \text{ die Menge aller Komponenten der Größe } > \left(\tfrac{3}{2}\right)^{i-1} \epsilon n \text{ und } \leq \left(\tfrac{3}{2}\right)^{i} \epsilon n.$$

Falls im Laufe des Algorithmus eine Komponente H aus $\mathcal{N}_i$ in kleinere Komponenten aufgeteilt wird, so haben diese dann höchstens $\frac{2}{3}|V(H)|$ Knoten und gehören daher nicht zu $\mathcal{N}_i$. Also sind für jedes $i = 0, \ldots, \log_{3/2} \frac{1}{\epsilon}$ die Komponenten in $\mathcal{N}_i$ knotendisjunkt. Da

die Zeit für eine Teilung einer Komponente nach Korollar 8.2.4 linear in der Anzahl ihrer Knoten ist, beträgt daher die Gesamtlaufzeit des Algorithmus höchstens $\log_{3/2} \frac{1}{\epsilon} \cdot \mathcal{O}(n) = \mathcal{O}(n \log \frac{1}{\epsilon})$.

Nun schätzen wir die Größe von C ab. Definiere $\delta := s - \sqrt{\frac{\epsilon n}{6}}$. Da

$$|\mathcal{N}_i| \cdot \left(\frac{3}{2}\right)^{i-1} \epsilon n \;\leq\; n,$$

gibt es höchstens $\left\lfloor \left(\frac{2}{3}\right)^{i-1} \frac{1}{\epsilon} \right\rfloor$ Komponenten in der Klasse $\mathcal{N}_i$. Daher ist

$$\begin{aligned}
|C| &\leq \left\lfloor \frac{n}{s} \right\rfloor + \sum_{i=1}^{\log_{3/2} n+1} \left\lfloor \left(\frac{2}{3}\right)^{i-1} \frac{1}{\epsilon} \right\rfloor (2s-1) \\
&\leq \left\lfloor \frac{n}{s} \right\rfloor + \frac{2s}{\epsilon} \sum_{i=0}^{\infty} \left(\frac{2}{3}\right)^{i} \\
&\leq \frac{n}{\sqrt{\frac{\epsilon n}{6}}} + \frac{6}{\epsilon}\left(\sqrt{\frac{\epsilon n}{6}} + \delta\right) \\
&\leq \sqrt{6}\sqrt{\frac{n}{\epsilon}} + \sqrt{6}\sqrt{\frac{n}{\epsilon}} + \frac{6}{\epsilon} \\
&= \sqrt{6}\sqrt{\frac{n}{\epsilon}}\,(1 + 1 + \sqrt{\frac{6}{\epsilon n}}).
\end{aligned}$$

□

Anmerkungen

MILLER [Mil86] zeigte (vgl. auch [GM90]) zeigte die Existenz von $(2\sqrt{2n}, \frac{2}{3})$-Kreisseparatoren in triangulierten zweifach-zusammenhängenden planaren Graphen; d.h. von Separatoren, deren Knoten alle auf einem Kreis liegen. Verschiedene Autoren beschäftigten sich mit Kantenseparatoren [BP92, Rao87, DDSV93, GSV94, CY94]. Bei dieser Problemstellung werden nur Kanten, aber keine Knoten, aus dem Graphen entfernt.

8.3 INDEPENDENT SET auf planaren Graphen

Dieser Abschnitt widmet sich dem Problem der Bestimmung einer maximum unabhängigen Menge in einem planaren Graphen. Zunächst zeigen wir, daß das Problem NP-vollständig ist. Mit Hilfe der Technik der Separatoren können wir dann aber einen immerhin nur schwach exponentiellen Algorithmus für INDEPENDENT SET auf planaren Graphen entwerfen; er hat Laufzeit $2^{\mathcal{O}(\sqrt{n})}$.

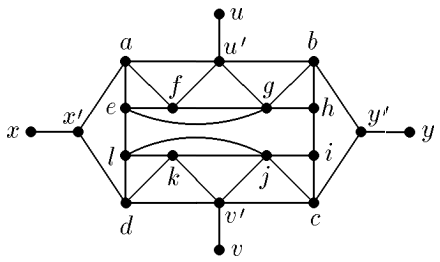
Satz 8.3.1 [GJS76, Lich82] PLANAR INDEPENDENT SET *ist* NP*-vollständig.*

Beweis [Kuc90]. Wir reduzieren von INDEPENDENT SET. Sei also ein Graph $G = (V, E)$ gegeben. Zunächst zeichnen wir G so in die Ebene, daß (vgl. Satz 6.5.9)

- die Kanten (d.h. die zugehörigen Jordankurven) nur ihre Endpunkte mit V gemeinsam haben,

- Kanten sich nicht selbst schneiden,
- Kanten andere Kanten höchstens einmal schneiden,
- sich in einem Punkt, der kein Knoten ist, höchstens zwei Kanten schneiden und
- die Anzahl der Überkreuzungen höchstens $\binom{n}{4}$ beträgt, vgl. Gleichung (6.6).

Falls sich dabei Kanten $\{u, v\}$ und $\{x, y\}$ aus E kreuzen, so ersetzen wir sie durch den folgenden *planaren* „Baustein-Graphen“:



Alle anderen Kanten, die in der ursprünglichen Zeichnung die Kante $\{x, y\}$ kreuzten, führen wir nun in der neuen Zeichnung so, daß sie die Kante $\{x, x'\}$ bzw. $\{y, y'\}$ kreuzen; analog bei den von $\{x, y\}$ verschiedenen Kanten von G, die in der ursprünglichen Zeichnung $\{u, v\}$ kreuzten. Wie wir gleich sehen werden, gilt für den entstandenen Graphen G':

$$\alpha(G') \;=\; \alpha(G) + 6. \tag{8.2}$$

Wenn wir diese Ersetzungsoperation also sukzessive für alle (sagen wir ℓ) Paare sich kreuzender Kanten durchführen, so erhalten wir einen planaren Graphen H mit $\alpha(H) = \alpha(G) + 6\ell$. Es gilt also

$$\alpha(G) \;\geq\; k \quad \Leftrightarrow \quad \alpha(H) \geq k + 6\ell.$$

Da die Transformation in polynomieller Zeit berechenbar ist, folgt die NP-Vollständigkeit von Planar Independent Set.

Beweis von Gleichung (8.2):
Sei $J := \{a, b, c, d, e, f, g, h, i, j, k, l, u', v', x', y'\} = V(G') \setminus V(G)$.

“$\geq$”: Sei S eine unabhängige Menge in G. Dann enthält S jeweils nur höchstens einen Knoten aus den Kanten $\{u, v\}$ und $\{x, y\}$. Falls $S \cap \{u, v, x, y\} \subseteq \{u, x\}$, so läßt sich die Knotenmenge $\{a, l, v', y', g, i\}$ zu S hinzufügen. Falls $S \cap \{u, v, x, y\} \subseteq \{u, y\}$, so läßt sich die Knotenmenge $\{b, i, v', x', f, l\}$ zu S hinzufügen. Aus Symmetriegründen findet man auch in den Fällen $S \cap \{u, v, x, y\}$ enthalten in $\{v, x\}$ bzw. $\{v, y\}$ sechs Knoten aus J, um die sich eine stabile Menge in G zu einer in G' erweitern läßt.

“$\leq$”: Sei T eine unabhängige Menge in G'. Wir zeigen im folgenden, wie wir aus T durch Streichen von höchstens sechs Knoten in $T \cap J$ eine stabile Menge in G erhalten. Wir unterscheiden vier Fälle:
Fall 1: $|T \cap \{u, v\}| \leq 1$ und $|T \cap \{x, y\}| \leq 1$.
Es gilt $|T \cap J| \leq 6$, denn T kann aus den Kreisen $\{x', a, e, l, d\}$ und $\{y', b, h, i, c\}$ jeweils nur höchstens zwei und aus den Kreisen $\{u', f, g\}$ und $\{v', k, j\}$ jeweils nur höchstens einen Knoten enthalten. Da in diesem Fall $T \setminus J$ in G stabil ist, genügt es, alle Knoten aus J aus T zu entfernen.
Fall 2: $u, v, x, y \in T$.

Da nun $T \cap \{u', v', x', y'\} = \emptyset$ gilt und T aus jedem der vier Dreiecke $\{a, e, f\}$, $\{b, g, h\}, \{l, k, d\}$ und $\{i, j, c\}$ nur höchstens einen Knoten enthalten kann, folgt $|T \cap J| \leq 4$. Wir setzen $T := T \setminus (J \cup \{x, u\})$.
Fall 3: $|T \cap \{u, v\}| \leq 1$ und $|T \cap \{x, y\}| = 2$.
Dann können x' und y' nicht in T enthalten sein. Wieder enthält T aus jedem der vier oben genannten Dreiecke nur höchstens einen Knoten. Falls also nur höchstens einer der Knoten u' und v' in T enthalten ist, so folgt unmittelbar $|T \cap J| \leq 5$. Falls aber u' und v' in T enthalten sind, so kann T an weiteren Knoten nur je höchstens einen aus den Cliquen $\{e, l\}$ und $\{h, i\}$ enthalten, es gilt also $|T \cap J| \leq 4$. Wir setzen $T := T \setminus (J \cup \{x\})$.
Fall 4: $|T \cap \{u, v\}| = 2$ und $|T \cap \{x, y\}| \leq 1$.
Die Knoten u' und v' können nun nicht in T enthalten sein. Wieder enthält T aus jedem der vier oben genannten Dreiecke nur höchstens einen Knoten. Falls also nur höchstens einer der Knoten x' und y' in T enthalten ist, so folgt unmittelbar $|T \cap J| \leq 5$. Falls aber x' und y' in T enthalten sind, so kann T an weiteren Knoten nur je höchstens einen aus den Cliquen $\{e, f, g\}$, $\{l, k, j\}$ und $\{h, i\}$ enthalten, und es folgt wieder $|T \cap J| \leq 5$. Wir setzen $T := T \setminus (J \cup \{u\})$. □

In Satz 1.4.13 haben wir gezeigt, daß das Problem INDEPENDENT SET schon für Graphen vom Maximalgrad 3 NP-vollständig ist. Die hierzu durchgeführte Reduktion von INDEPENDENT SET erhält die Planarität eines Graphen. Mithin hat Satz 8.3.1 das folgende

Korollar 8.3.2 PLANAR INDEPENDENT SET *mit* $\Delta \leq 3$ *ist* NP*-vollständig.* □

Nun wollen wir zeigen, wie das Problem PLANAR INDEPENDENT SET in nur schwach exponentieller Zeit exakt gelöst werden kann. Die Idee des Verfahrens besteht darin, zunächst einen $(\sqrt{8n}, 2/3)$-Separator C zu bestimmen. Für alle unabhängigen Teilmengen S von $G[C]$ wird dann in jeder Komponente von $G \setminus C$ rekursiv eine maximum unabhängige Menge bestimmt, die keinen Nachbarn in S hat. Diese unabhängigen Mengen und S werden zu einer Menge I zusammengesetzt, und die größte der Mengen I stellt offensichtlich eine maximum stabile Menge in G dar. Die folgende Funktion PLANAR_IND_SET (G) bestimmt damit in einem planaren Graphen G eine maximum unabhängige Menge.

```
FUNCTION PLANAR_IND_SET (G) : 2^V;

BEGIN
  IF |V(G)| ≤ 1 THEN
    I := V(G);
  ELSE BEGIN
    I := ∅;
    berechne einen (√(8n), 2/3)-Separator C von G, so daß G[V \ C] in
      nichtadjazente Mengen A und B der Größe höchstens (2/3)n zerfällt;
    FOR {S ⊆ C : S stabil in G[C]} DO BEGIN
      I_A := PLANAR_IND_SET (G[A \ Γ(S)]);
      I_B := PLANAR_IND_SET (G[B \ Γ(S)]);
      IF |S| + |I_A| + |I_B| > |I| THEN I := S ∪ I_A ∪ I_B;
    END; {for}
  END; {else}
  PLANAR_IND_SET:= I;                                  {Rückgabewert}
END; {PLANAR_IND_SET}
```

Proposition 8.3.3 (LIPTON, TARJAN 1980)
In einem planaren Graphen G auf n Knoten berechnet PLANAR_IND_SET (G) *in* $2^{\mathcal{O}(\sqrt{n})}$ *Schritten eine maximum stabile Menge in G.*

Beweis. Die Korrektheit des Algorithmus ist klar. Sei $t(n)$ die Zeitkomplexität von PLANAR_IND_SET für einen Graphen auf n Knoten. Die Berechnung von C erfordert nach Korollar 8.2.4 $\mathcal{O}(n)$ Rechenschritte. Die `FOR`-Schleife wird höchstens $2^{\sqrt{8n}}$-mal durchlaufen. Folglich genügt $t(n)$ der Rekursion

$$t(n) \;\leq\; \mathcal{O}(n) \;+\; 2^{\sqrt{8n}}\, \mathcal{O}(n)\, t(\frac{2}{3}n). \tag{8.3}$$

Seien K_1 bzw. K_2 Konstanten, so daß die Funktionen, für die die $\mathcal{O}$-Terme in (8.3) stehen, durch $K_1 n$ bzw. $K_2 n$ beschränkt sind. Sei ferner $n_0 \in \mathbb{N}$ derart, daß

$$(K_1 + K_2)\, n \;\leq\; 2^{\sqrt{8n}} \qquad \text{für alle } n \geq n_0, \tag{8.4}$$

und L eine Konstante, so daß $t(n) \leq L$ für alle $n < n_0$. Wir zeigen nun durch Induktion nach n, daß

$$t(n) \;\leq\; L \cdot 2^{\frac{2\sqrt{8n}}{1-\sqrt{2/3}}}.$$

Für alle $n < n_0$ ist diese Ungleichung trivialerweise erfüllt. Für $n \geq n_0$ gilt hingegen

$$\begin{array}{rcll} t(n) & \overset{(8.3)}{\leq} & K_1 n \;+\; 2^{\sqrt{8n}}\, K_2 n\, t(\frac{2}{3}n) & \\ & \overset{(8.4)}{\leq} & 2^{2\sqrt{8n}}\, t(\frac{2}{3}n) & \\ & \overset{(IA)}{\leq} & 2^{2\sqrt{8n}}\, L \cdot 2^{\frac{2\sqrt{8\frac{2}{3}n}}{1-\sqrt{2/3}}} & = \quad L \cdot 2^{\frac{2\sqrt{8n}}{1-\sqrt{2/3}}}. \end{array}$$

□

Anmerkungen

Mit einem ähnlichen Algorithmus läßt sich auch die chromatischen Zahl eines planaren Graphen in schwach exponentieller Rechenzeit bestimmen.

Übung 8.3.4 (LIPTON, TARJAN 1980) *Man entwerfe einen Algorithmus, der in* $2^{\mathcal{O}(\sqrt{n})}$ *Schritten die chromatische Zahl eines planaren Graphen bestimmt.*

In Abschnitt 10.8 wird ein polynomiales Approximationsschema für PLANAR INDEPENDENT SET vorgestellt, das auf Satz 8.2.5 beruht. RAVI und HUNT [RH87] geben eine Erweiterung der Funktion PLANAR_IND_SET an, die es erlaubt, in $2^{\mathcal{O}(\sqrt{n})}$ Zeit verschiedene Zählprobleme in planaren Graphen zu lösen, wie die Anzahl von maximum unabhängigen Mengen, die Anzahl von minimalen Knotenüberdeckungen und die Anzahl von Dreifärbungen. Weitere Anwendungen findet der planare Separatorensatz im VLSI-Layout [Lei80], bei der Lösung von dünnen linearen Gleichungssystemen, die von Differentialgleichungen auf zweidimensionalen Gebieten herrühren [LRT79], beim kürzesten-Pfad-Problem [Fre87] und bei MAX CUT [Bar90a] in planaren Graphen, bei planaren Netzwerkflußproblemen [JV83, NC88] und verschiedenen anderen Problemen [LT80, Chu90].

8.4 Separatoren für andere Graphenklassen

MILLER, TENG und VAVASIS [MTV91] konnten einen Separatorsatz für die Klasse der sogenannten d dimensionalen overlap-Graphen zeigen, die sich geometrisch über Schnitte von Kugeln im $\mathbb{R}^d$ definieren. Für diese Klasse von Graphen existieren Separatoren der Größe $\mathcal{O}(n^{\frac{d-1}{d}})$. Nach Satz 6.1.58 lassen sich planare Graphen über Schnitte von Kugeln im $\mathbb{R}^2$, d.h. Kreisen, definieren, und es folgt erneut die Existenz eines Separators der Größe $\mathcal{O}(\sqrt{n})$ in planaren Graphen.

Der Satz 8.2.1 über planare Separatoren läßt sich auch auf Graphen höheren Geschlechts verallgemeinern.

Satz 8.4.1 (GILBERT, HUTCHINSON, TARJAN 1984) *Jeder Graph G auf n Knoten vom Geschlecht $\gamma \geq 0$ besitzt einen $(6\sqrt{\gamma n}+2\sqrt{2n}+1, \frac{2}{3})$-Separator, der in $\mathcal{O}(n+\gamma)$ Schritten bestimmt werden kann.* □

Mit Hilfe von Expandergraphen konnten die Autoren ferner zeigen, daß die obere Schranke für die Größe des Separators bis auf einen konstanten Faktor optimal ist. Djidjev [Dji85a, Dji85b] zeigte ähnliche Resultate.

Ein Separatorsatz gilt ferner für jedwede Graphenklasse, die sich durch einen ausgeschlossenen Minor definiert.

Satz 8.4.2 (ALON, SEYMOUR, THOMAS 1990a, 1990b) *Sei H ein Graph. Jeder Graph G auf n Knoten ohne H-Minor besitzt einen $(|H|^{3/2}\sqrt{n}, \frac{2}{3})$-Separator.* □

Man beachte, daß Satz 8.4.2 keine direkte Verallgemeinerung von Satz 8.4.1 ist. Nach Satz 6.4.13 gilt nämlich $\gamma(K_\ell) = \Theta(\ell^2)$. Wenn wir G also als den Graphen definieren, der aus dem K_ℓ durch $\frac{n}{\binom{\ell}{2}}$ malige Unterteilung aller Kanten entsteht, dann gilt immer noch $\gamma(G) = \Theta(\ell^2)$. Satz 8.4.1 garantiert dann die Existenz eines $(\mathcal{O}(\ell\sqrt{n}), \frac{2}{3})$-Separators, aber Satz 8.4.2 mit $H = K_\ell$ nur die eines $(\mathcal{O}(\ell^{3/2}\sqrt{n}), \frac{2}{3})$-Separators.

Die beiden letzten Sätze könnten den Eindruck erwecken, daß alle Graphenklassen mit nur $\mathcal{O}(n)$ Kanten Separatoren der Größe $o(n)$ besitzen. Der folgende Satz zeigt, daß dies nicht der Fall ist.

Satz 8.4.3 (ERDŐS, GRAHAM, SZEMERÉDI 1975) *Für jede Konstante $c \geq 2$ haben fast alle Graphen aus $\mathcal{G}_{n,\lfloor cn \rfloor}$ keinen $(\frac{1}{70}n, \frac{2}{3})$-Separator.*

Beweis. Sei also $c \geq 2$ eine fest gewählte Konstante und $G_{n,\lfloor cn \rfloor} = (V_n, E(G_{n,\lfloor cn \rfloor}))$ ein zufälliges Element aus $\mathcal{G}_{n,\lfloor cn \rfloor}$, d.h.

$$\text{Prob}\,(G_{n,\lfloor cn \rfloor} = G) = \frac{1}{|\mathcal{G}_{n,\lfloor cn \rfloor}|} \qquad \text{für jedes } G \in \mathcal{G}_{n,\lfloor cn \rfloor}.$$

Sei $\mathcal{S}$ die Menge der möglichen Unterteilungen von V_n:

$$\mathcal{S} := \{(A,B,C) \mid\ V_n = A \cup B \cup C,\ |C| = \lfloor \frac{1}{70}n \rfloor,\ |A| \leq \frac{2}{3}n \text{ und } |B| \leq \frac{2}{3}n\}.$$

Für jedes $(A,B,C) \in \mathcal{S}$ sei $X_{A,B,C}$ die Indikatorvariable des Ereignisses, daß $A \cup B \cup C$ den Graphen $G_{n,\lfloor cn \rfloor}$ separiert, d.h., daß es keine Kanten zwischen A und B in $G_{n,\lfloor cn \rfloor}$ gibt.

Dann müssen also alle Kanten von $G_{n,\lfloor cn\rfloor}$ in den $\binom{n}{2} - |A||B|$ Kanten aus $V_n \times V_n \backslash A \times B$ liegen. Wegen $|A||B| \geq \frac{2}{3}n\left(\frac{1}{3} - \frac{1}{70}\right)n = \frac{67}{315}n^2$ gilt dann

$$\begin{aligned}\text{Prob}\,(X_{A,B,C} = 1) &= \frac{\binom{\binom{n}{2}-|A||B|}{\lfloor cn\rfloor}}{\binom{\binom{n}{2}}{\lfloor cn\rfloor}} = \prod_{i=0}^{\lfloor cn\rfloor - 1} \frac{\binom{n}{2} - |A||B| - i}{\binom{n}{2} - i} \\ &\leq \left(1 - \frac{|A||B|}{\binom{n}{2}}\right)^{\lfloor cn\rfloor} \leq \left(1 - \frac{\frac{67}{315}n^2}{\binom{n}{2}}\right)^{\lfloor cn\rfloor} \leq e^{c\cdot\ln(\frac{181}{315})\cdot n}.\end{aligned}$$

Andererseits gibt es höchstens 3^n Möglichkeiten, um V_n in $A \cup B \cup C$ zu partitionieren. Daher ist

$$\begin{aligned}&\text{Prob}\,(G_{n,\lfloor cn\rfloor} \text{ hat keinen } (\frac{1}{70}n, \tfrac{2}{3})\text{-Separator}) \\ &\geq 1 - \sum_{(A,B,C)\in\mathcal{S}} \text{Prob}\,(X_{A,B,C} = 1) \geq 1 - 3^n \cdot e^{c\cdot\ln(\frac{181}{315})\cdot n} = 1 - o(1),\end{aligned}$$

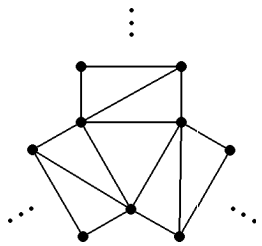
da $\ln 3 - c \cdot \ln(\frac{181}{315}) < 0$ wegen $c \geq 2$. □

Cliquenseparatoren in chordalen Graphen

Wir erinnern daran, daß ein Graph chordal (vgl. Abschnitt 7.2) heißt, falls er keinen C_k mit $k \geq 4$ als induzierten Subgraph enthält, d.h. jeder Kreis auf mindestens vier Knoten eine Sehne besitzt.
Da beispielsweise der K_n chordal ist, können chordale Graphen sicher nicht durch $o(n)$ Knoten separiert werden. Das folgende Ergebnis ist eine Verallgemeinerung von Proposition 8.1.3 über Separatoren in Bäumen.

Satz 8.4.4 [GRE84] *Sei G ein chordaler Graph mit Cliquenzahl $\omega(G)$. Dann besitzt G einen $\left(\omega(G), \frac{1}{2}\right)$-Separator, der als Clique gewählt werden kann.*

Für den K_n ist dieser Satz trivial. Ein (inklusions-)minimaler $(\omega(G), \frac{1}{2})$-Separator in einem chordalen Graphen G ist nicht notwendig eine Clique, wie es das Beispiel des Separators $\{v_2, v_4\}$ im $P_5 = (v_1, \ldots, v_5)$ zeigt.
Lemma 7.2.3 besagt, daß in einem chordalen Graphen jede minimal s-t-trennende Knotenmenge eine Clique ist, also insbesondere höchstens $\omega(G)$ viele Knoten hat. Ein $\left(\cdot, \frac{1}{2}\right)$-Separator ist nicht notwendigerweise minimal trennend, vgl. das mittlere Dreieck des folgenden Graphen:



Dieses Beispiel zeigt insbesondere, daß Satz 8.4.4 bestmöglich bezüglich der Größe des Separators ist.[2]

[2] In [GRE84] wird die Existenz eines $\left(\omega(G) - 1, \frac{1}{2}\right)$-Separators behauptet.

Beweis von Satz 8.4.4.
Sei $G = (V, E)$ chordal mit $|V| = n$ Knoten und $|E| = m$ Kanten. Wir zeigen, daß die Ausgabe C des folgenden Algorithmus das Gewünschte leistet:

```
PROCEDURE CHORDAL_SEPARATOR (G,C);

BEGIN
  C := ∅;
  WHILE ∃ Komponente A von G \ C mit |A| > n/2 DO BEGIN
    C := {c ∈ C : Γ(c) ∩ A ≠ ∅};
    bestimme ein a ∈ A mit C ⊂ Γ(a);
    C := C ∪ {a};
  END;
END;
```

Offenbar besitzt G nur höchstens eine Zusammenhangskomponente A mit $|A| > n/2$. Von dieser wird in jedem Durchlauf der `WHILE`-Schleife mindestens ein Knoten a entfernt. Da der gesamte übrige Graph dann höchstens $|V \setminus A| + 1 \leq n/2$ Knoten hat, muß die größte Komponente A' von $G\setminus(C \cup \{a\})$, falls sie mehr als $n/2$ Knoten hat, im nächsten Schleifendurchlauf die größte Komponente von $A - a$ sein, so daß der Algorithmus nach höchstens $\lceil n/2 \rceil$ Schleifendurchläufen terminiert. Es bleibt lediglich, die Existenz des Knotens $a \in A$ mit $C \subset \Gamma(a)$ zu zeigen. Im ersten Durchlauf der Schleife ist $a \in A$ frei wählbar. Der erste Schritt der Schleife sorgt dafür, daß die Menge C stets minimal mit der Eigenschaft ist, A von $V \setminus (C \cup A)$ zu trennen. Deshalb haben alle Knoten in C mindestens einen Nachbarn in A. Sei $\sigma = (a_1, \ldots, a_{|A|}, c_1, \ldots, c_{|C|})$ ein perfektes Knoteneliminationsschema für den chordalen Graphen $G[A \cup C]$, welches nach Übung 7.2.14 existieren muß. Betrachte den Knoten $a := a_{|A|} \in A$. Jeder Knoten $c \in C$ ist über einen seiner Nachbarn in A durch einen $c - a$-Pfad mit a verbunden, der als innere Knoten höchstens weitere Knoten aus A benutzt. Diese sind in σ vor a und c angeordnet, so daß nach Proposition 7.2.10 $\{c, a\} \in E$. Der Knoten a ist also vollständig zu C verbunden, und es folgt, daß die Menge C eine Clique in G induziert. □

Der Separator kann also ohne weiteres in Zeit $\mathcal{O}(nm)$ konstruiert werden. In [GRE84] findet man sogar einen linearen Algorithmus für dieses Problem.
Wegen $|E(G[C])| = \binom{|C|}{2} \leq m$ ist $|C| \leq \sqrt{2m}$, so daß der Satz effiziente divide-and-conquer Algorithmen für dünne, chordale Graphen ermöglicht. Für wichtige Graphenparameter wie $\chi(G)$, $\omega(G)$, $\alpha(G)$ und $\theta(G)$ haben wir jedoch sogar lineare, direkte Algorithmen bei chordalen Graphen kennengelernt.

8.5 Cliquenseparatoren

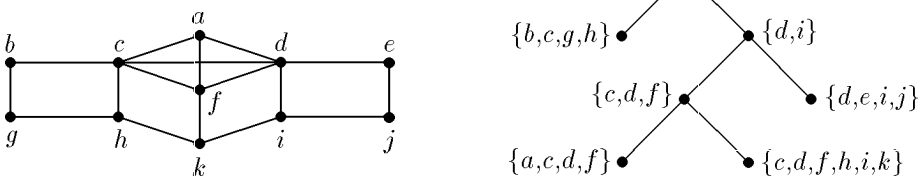
Nun wenden wir uns den Problemen GRAPH COLORING, INDEPENDENT SET und CLIQUE zu. Wenn es uns gelingt in einem Graphen einen Separator zu finden, der eine Clique ist, dann ist es besonders einfach, die Lösungen dieser Probleme für die einzelnen Komponenten zu einer Gesamtlösung zusammenzusetzen, vorausgesetzt wir können die Probleme für die Komponenten lösen. Die sukkzessive Zerlegung von Graphen durch Cliquenseparatoren, welche allerdings nicht eindeutig ist, läßt sich durch sogenannte Cliquenseparatorenbäume darstellen, die wie folgt definiert sind.

Definition 8.5.1
a) Ein Cliquenseparator *in einem Graphen $G = (V, E)$ ist eine Clique $C \subseteq V$, so daß $G[V \backslash C]$ unzusammenhängend ist.*
b) Ein Cliquenseparatorbaum *ist rekursiv wie folgt definiert*

- *Falls G keinen Cliquenseparator enthält, dann besteht ein Cliquenseparatorbaum für G nur aus einem Knoten, dem die Menge V zugeordnet ist und der gleichzeitig die Wurzel des Cliquenseparatorbaumes von G ist.*
- *Falls G einen Cliquenseparator C besitzt, dann seien $A_1, \ldots, A_\ell$ die Zusammenhangskomponenten von $G[V \backslash C]$. Ein Cliquenseparatorbaum für G besteht dann aus einem Knoten w, der Wurzel des Baumes, dem die Menge C zugeordnet wird. Ferner ist w mit den Wurzeln der Cliquenseparatorbäume für $G[A_i \cup C]$ für $i = 1, \ldots, \ell$ verbunden.*

c) Die den Blättern des Cliquenseparatorbaumes zugeordneten Mengen heißen die Atome *der Zerlegung.*

Die folgende Abbildung zeigt einen Graphen G und einen Cliquenseparatorbaum für G.



Die Atome einer Zerlegung besitzen offenbar keinen Cliquenseparator. Die folgende Übung gibt eine obere Schranke für die Anzahl der Atome eines Cliquenseparatorbaumes an.

Übung 8.5.2 *Ein Cliquenseparatorbaum für einen Graphen G kann höchstens $\binom{n}{2} - m$ Atome enthalten.*

Mit Hilfe von Lemma 7.2.3 läßt sich leicht die folgende Aussage über die Struktur von Cliquenseparatorbäumen für chordale Graphen zeigen.

Übung 8.5.3 *Jeder chordale Graph besitzt einen Cliquenseparatorbaum, dessen Atome alle Cliquen sind.*

Nun werden wir zeigen, wie man Cliquenseparatorbäume effizient bestimmen kann. Dazu müssen wir zunächst tiefer in die bereits im Abschnitt 7.2 behandelte Theorie der Eliminationsschemata einsteigen.

Definition 8.5.4 *Sei $G = (V, E)$ ein Graph auf n Knoten. Eine Bijektion $\sigma : V \to \{1, \ldots, n\}$ heißt* Ordnung *(der Knoten von G). Der* Fill-in *zu einer Ordnung σ ist folgendermaßen definiert.*

$$F_\sigma := \{\{v, w\} : v \neq w, \{v, w\} \notin E, \text{ und es gibt einen Pfad } v = x_1 \ldots x_k = w \text{ in } G \text{ mit } \sigma(x_i) < \min\{\sigma(v), \sigma(w)\}\ \forall i = 2, \ldots, k-1\}.$$

Nach Proposition 7.2.10 ist $F_\sigma = \emptyset$ äquivalent damit, daß σ ein perfektes Eliminationsschema (PES) ist. Mit $G_\sigma = (V, E \cup F_\sigma)$ gilt ferner die folgende Aussage.

Übung 8.5.5 *Jede Ordnung σ ist ein PES zu G_σ. Der Fill-in zu σ in G_σ ist also die leere Menge. Insbesondere ist G_σ chordal.*

Sei nun $G = (V, E)$ ein Graph und σ eine Ordnung der Knoten von G. Sei ferner $F \subset V \times V$ und v ein Knoten in G. Dann ist die *obere Nachbarschaft* $\Gamma_{\sigma,F}(v)$ von v in $G' = (V, E \cup F)$ wie folgt definiert.

$$\Gamma_{\sigma,F}(v) := \{w : \{v, w\} \in E \cup F \text{ und } \sigma(w) > \sigma(v)\}$$

Wir definieren ferner für einen Knoten v den Knoten $m_F(v)$ als den Knoten u, für den $\sigma(u) = \min\{\sigma(w) : w \in \Gamma_{\sigma,F}(v)\}$ gilt. Der Fill-in zu einer Ordnung σ läßt sich dann folgendermaßen charakterisieren.

Lemma 8.5.6 *Sei G ein Graph und σ eine Ordung. Dann ist der Fill-in F_σ die kleinste Menge $F \subseteq V \times V$, so daß für alle $v \in V$ gilt*

$$\Gamma_{\sigma,F}(v) \subseteq \Gamma_{\sigma,F}(m_F(v)) \cup \{m_F(v)\}. \tag{8.5}$$

Beweis. Wir zeigen zuerst, daß $F = F_\sigma$ (8.5) erfüllt. Sei also v ein Knoten, $w \in \Gamma_{\sigma,F_\sigma}(v)$ und $w \neq m_F(v)$. Dann ist $m_F(v), v, w$ ein Pfad in G_σ mit $\sigma(v) < \min\{\sigma(m_F(v)), \sigma(w)\}$. Da der Fill-in in G_σ zu σ nach Übung 8.5.5 gleich der leeren Menge ist, gilt $\{w, m_F(v)\} \in E \cup F_\sigma$ und daher $w \in \Gamma_{\sigma,F_\sigma}(m_F(v))$.
Sei nun F eine Menge, die (8.5) erfüllt. Dann ist nachzuweisen, daß $F_\sigma \subset F$. Dazu zeigen wir per Induktion nach i, daß für alle $\{v, w\} \in F_\sigma$ mit $\sigma(v) \leq i$ gilt $\{v, w\} \in F$. Sei diese Aussage also für alle $i \leq i_0$ bewiesen und $\{v, w\} \in F_\sigma$ mit $\sigma(v) = i_0 + 1 \leq \sigma(w)$. Dann gibt es einen Pfad $v = x_1 \ldots x_k = w$ in $G_\sigma = (V, E \cup F_\sigma)$ mit $k \geq 3$ und $\sigma(x_j) < \min\{\sigma(v), \sigma(w)\}$ für $j = 2, \ldots, k-1$. Sei ferner k minimal mit dieser Eigenschaft. Falls $k > 3$, so sei $\ell \in \{2, \ldots, k-1\}$ mit $\sigma(x_\ell) = \max\{\sigma(x_j) : j = 2, \ldots, k-1\}$, und ohne Einschränkung gelte $\ell > 2$. Dann ist $v = x_1 \ldots x_\ell$ ein Pfad in G_σ mit $\sigma(x_j) \leq \min\{\sigma(v), \sigma(x_\ell)\}$ für $j = 2, \ldots, \ell - 1$. Nach der Definition des Fill-in ist dann $\{v, x_\ell\} \in F_\sigma$, was bedeutet, daß wir den Pfad abkürzen können im Widerspruch zur Minimalität von k. Also ist $k = 3$, und es gibt einen Knoten $u = x_2$ mit $v, w \in \Gamma_{\sigma,F_\sigma}(u)$. Wir wählen u so, daß $\sigma(u)$ maximal mit dieser Eigenschaft ist. Nach Induktionsannahme gilt wegen $\sigma(u) < \sigma(v)$ auch $v, w \in \Gamma_{\sigma,F}(u)$. Wenn $v \neq m_F(u)$ wäre, dann wären nach (8.5) $v, w \in \Gamma_{\sigma,F}(m_F(u))$ im Widerspuch zur Maximalität von $\sigma(u)$. Also ist $v = m_F(u)$. Da (8.5) $w \in \Gamma_{\sigma,F}(m_F(u))$ impliziert, ist also $\{v, w\} = \{m_F(u), w\} \in F$, und es folgt per Induktion, daß $F_\sigma \subset F$. □

Der folgende Algorithmus benutzt Lemma 8.5.6, um für einen Graphen G und eine Ordnung σ den Fill-in F_σ zu berechnen.

```
FUNCTION FILL_IN (G, σ) : SET OF EDGES;

BEGIN
  FOR v ∈ V DO
    A(v) := Γ_{σ,∅}(v) = {w ∈ Γ(V) : σ(w) > σ(v)};
  FOR i := 1 TO n − 1 DO BEGIN
    v := σ^{-1}(i);
    m(v) := σ^{-1}(min {σ(u) : u ∈ A(v)});
```

```
      FOR {w ∈ A(v) : w ≠ m(v)} DO
        A(m(v)) := A(m(v)) ∪ {w};
    END; {for}
    F_σ := ∅;
    FOR v ∈ V DO FOR w ∈ A(v)\Γ(v) DO
      F_σ := F_σ ∪ {v, w};
    FILL_IN:= F_σ;
  END {FILL_IN}
```

Proposition 8.5.7 *Zu einem Graphen G und einer Ordnung σ berechnet* FILL_IN (G, σ) *den Fill-in F_σ. Die Prozedur kann mit einer Laufzeit von $\mathcal{O}(n + m + |F_\sigma|) = \mathcal{O}(n^2)$ implementiert werden.*

Beweis. Die Korrektheit des Verfahrens folgt direkt aus Lemma 8.5.6. Die Mengen $A(v)$ werden als Listen implementiert. Dabei müssen wir darauf achten, daß ein Knoten w beim Durchlaufen der inneren FOR-Schleife nicht doppelt in der Liste vorkommt, da sonst die Listen zu lang werden. Die Berechnung von $m(v)$ kann in einer FOR-Schleife realisiert werden. Dabei können auch doppelte Elemente aus $A(v)$ eliminiert werden, was beispielsweise unter Verwendung eines Hilfsfeldes $test(u)$ realisiert werden kann, das beim ersten Auftreten von u in $A(v)$ TRUE gesetzt wird, woraufhin alle anderen Kopien von u aus $A(v)$ entfernt werden. Die Mengen $A(v)$ können, als Listen aufgefaßt, im Verlauf des Algorithmus insgesamt nur $m+|F_\sigma|$ Elemente enthalten. Deshalb erfordert die Elimination der doppelten Elemente höchstens $m + |F_\sigma|$ zusätzliche Operationen. Details finden sich in [RTL76]. Die Gesamtlaufzeit des Algorithmus ist dann beschränkt durch $\mathcal{O}(n+\sum_{v\in V}|A(v)|) = \mathcal{O}(n+m+|F_\sigma|)$. □

Falls ein Graph G kein PES enthält, was nach Proposition 7.2.9 äquivalent dazu ist, daß G nicht chordal ist, dann sind wir zumindest an *minimalen* Eliminationsschemata interessiert, die wie folgt definiert sind.

Definition 8.5.8 *Eine Ordnung σ der Knoten eines Graphen G heißt* minimales Eliminationsschema (MES)*, falls der Fill-in F_σ minimal ist; d.h. es gibt kein Ordnung π, die $F_\pi \subset F_\sigma$ erfüllt.*

Die Bedeutung von MES liegt darin, daß ein Cliquenseparator für G auch ein Cliquenseparator für G_σ ist, falls σ ein MES ist, was wir in Lemma 8.5.10 zeigen werden. Die Bestimmung eines *minimum* Eliminationsschema, das also einen kardinalitätsminimalen Fill-in hat, ist NP-schwer (vgl. [Yan81b]). Nun wollen wir zeigen, wie ein MES in einem Graphen G in polynomieller Zeit bestimmt werden kann. Der Algorithmus stammt von ROSE, TARJAN und LUEKER und stellt eine Verallgemeinerung von BREITENSUCHE dar. Die in BREITENSUCHE verwendete Warteschlange wird hier durch eine sogenannte Vorrangwarteschlange (engl. priority queue) ersetzt, bei der jeder in der Schlange enthaltene Knoten eine bestimmte Priorität hat. Diese Pioritäten sind durch Untermengen von $\{1, \ldots, n\}$ gegeben und werden in dem Feld pr festgehalten. Eine Menge A hat eine höhere Priorität als eine Menge B, wenn die Elemente von A lexikographisch gesehen kleiner sind als die Elemente von B. So hat beispielsweise die Menge $\{2, 4, 5\}$ eine höhere Priorität als die Menge $\{2, 5\}$. Jede nicht-leere Menge hat eine höhere Priorität als die leere Menge.

```
FUNCTION LEX_BREITENSUCHE (G) : Ordnung;
```

```
BEGIN
  FOR v ∈ V DO BEGIN
    pr(v) := ∅;
   σ(v) := 0;
  END;
  FOR i := n DOWNTO 1 DO BEGIN
    Sei v ein Knoten mit höchster Priorität pr(v) und σ(v) = 0;
    σ(v) := i;
    FOR w mit σ(w) = 0 DO
      IF ∃ einen Weg v = v_1 ... v_{k+1} = w mit σ(v_j) = 0
                                    und pr(v_j) < pr(w) für j = 2...k THEN
        pr(w) := pr(w) ∪ {i};
  END; {for}
END;
```

Der Beweis dafür, daß die Ordnung σ :=LEX_BREITENSUCHE (G) tatsächlich ein MES für G ist, ist recht aufwendig und sehr technisch. Wir wollen deshalb an dieser Stelle auf ihn verzichten. Er kann in [RTL76] nachgelesen werden. Dort ist auch eine Implementierung des Verfahrens mit der Laufzeit $\mathcal{O}(n(m+n))$ angegeben, die wir nun beschreiben wollen.

Zunächst gilt es zu beachten, daß die durch die Mengen pr definierte Ordung im allgemeinen nicht linear ist, d.h. es kann verschieden Knoten gleicher Priorität geben. Seien in einem Schleifendurchlauf $k := |\{pr(v) : v \text{ mit } \sigma(v) = 0\}|$ verschiedene Prioritäten vorhanden. Dann erhalten alle unmarkierten Knoten ein Label aus $\{1, \ldots, k\}$, das ihrer Priorität entspricht. Als v wird dann ein Knoten mit Label k ausgewählt. Die in der `FOR`-Schleife einzufügenden Fill-in Kanten können mit einer Variante von TIEFENSUCHE in $\mathcal{O}(n+m)$ Schritten bestimmt werden. Dabei wird das Label eines Knotens w um $\frac{1}{2}$ erhöht, falls $\{v, w\}$ Fill-in Kante ist, d.h. $\sigma(v) = i$ der Menge $pr(w)$ zugefügt wird. Anschließend werden die Label durch Umsortieren wieder ganzzahlig gemacht, was offenbar in $\mathcal{O}(n)$ Schritten geschehen kann. Daher werden für einen Durchlauf der `FOR`-Schleife höchstens $\mathcal{O}(n+m)$ Rechenschritte benötigt.

Satz 8.5.9 [RTL76] *Sei G ein Graph. Dann kann der Algorithmus* LEX_BREITENSUCHE *in $\mathcal{O}(n(n+m))$ Schritten ein MES für G bestimmen.*

Unabhängig davon geben auch OHTSUKI, CHEUNG und FUJISAWA [OCF76] einen Algorithmus für das gleiche Problem an und diskutieren auch den Zusammenhang mit der effizienten Implementierung des GAUSSschen Verfahrens zur Lösung von (dünnen) linearen Gleichungssystemen.

Nun wenden wir uns wieder dem Problem der Bestimmung von Cliquenseparatorenbäumen zu.

Lemma 8.5.10 *Sei σ ein MES der Knoten eines Graphen $G = (V, E)$. Dann ist ein Cliquenseparator C für G auch ein Cliquenseparator für G_σ.*

Beweis. Sein $V_1, \ldots, V_k$ die Knotenmengen der Zusammenhangskomponenten von $G[V \backslash C]$. Wir entfernen aus F_σ alle Kanten mit Endknoten aus verschiedenen Mengen V_i und erhalten so eine Menge $F \subset F_\sigma$. Zunächst zeigen wir, daß $G' = (V, E \cup F)$ chordal ist.
Sei also K ein Kreis in G' der Länge mindestens 4. Falls $K \subseteq G[C \cup V_i]$ für ein i, so hat K eine Sehne in F_σ, da G_σ nach Übung 8.5.5 chordal ist. Diese Sehne liegt dann auch in

$E \cup F$. Falls K Knoten aus verschiedenen V_i enthält, dann besitzt K zwei nicht aufeinanderfolgende Knoten aus C. Da C eine Clique ist, sind diese beiden Knoten benachbart, und K besitzt auch in diesem Fall eine Sehne in G'. Also ist G' chordal und hat nach Proposition 7.2.9 ein PES π, das also $F_\pi = F$ erfüllt. Da σ ein MES ist, folgt $F = F_\pi = F_\sigma$, und F_σ enthält keine Kanten mit Endknoten aus verschiedenen V_i, was zu zeigen war. □

Der folgende Algorithmus zur Bestimmung eines Cliquenseparatorenbaumes eines Graphen G berechnet zuerst ein MES σ und den dazugehörenden Fill-in F_σ. Dann werden für alle Knoten v aus G Mengen $C(v) = \{w : \sigma(w) > \sigma(v) \text{ und } \{v,w\} \in E \cup F_\sigma\}$ definiert, die Kanditaten für Cliquenseparatoren sind. Ob dies der Fall ist, wird im zweiten Teil des Algorithmus getestet. Die Atome der Zerlegung werden im Feld *Atom* gespeichert. Zum Beweis der Korrektheit des Algorithmus werden wir die Aussage von Lemma 8.5.10 verwenden.

```
PROCEDURE CS_Baum (G);

BEGIN
  σ := Lex_Breitensuche (G);
  F_σ := Fill-in (G; σ);
  FOR v ∈ V DO BEGIN
    C(v) := ∅;
    FOR w ∈ V DO
      IF (σ(w) > σ(v)) AND ({v,w} ∈ E ∪ F_σ) THEN
        C(v) := C(v) ∪ {w};
  END; {for}
  k := 1;
  FOR i := 1 TO n − 1 DO BEGIN
    IF v := σ^{-1}(i) ∈ G THEN BEGIN
      Sei A die Knotenmenge der Zusammenhangskomponente von
              G[V\C(v)], die v enthält.
      B := V\(C(v) ∪ A);
      IF (C(v) ist Clique in G) AND (B ≠ ∅) THEN BEGIN
        Atom(k) := A;
        k := k + 1;
        G := G[B ∪ C(v)];
      END; {if}
    END; {if}
  END; {for}
  Atom(k) := V(G);
END;
```

Nun zeigen wir, daß der Algorithmus einen Cliquenseparator findet, falls G einen solchen besitzt.

Lemma 8.5.11 *Falls G einen Cliquenseparator enthält, dann ist eine der Mengen $C(v)$ für einen Knoten v aus G ein Cliquenseparator.*

Beweis. Sei σ das im Algorithmus berechnete MES. Sei C ein inklusionsminimaler Cliquenseparator in G und A und B zwei Zusammenhangskomponenten von $G[V\backslash C]$. Da C minimal ist, hat jeder Knoten aus C jeweils mindestens einen Nachbarn aus A und B. Seien

x und y die Knoten mit dem höchsten σ-Wert in A beziehungsweise in B. Angenommen, es gibt einen Knoten $z \in C$ mit

$$\sigma(z) < \min\{\sigma(x), \sigma(y)\}. \tag{8.6}$$

Sei $x = x_1 x_2 \ldots x_{j-1} x_j = z$ ein kürzester Pfad in G_σ von x nach z mit der Eigenschaft, daß $x_1, \ldots, x_{j-1} \in A$. Ein solcher existiert, da z einen Nachbarn in A besitzt. Falls es ein $i \leq j-1$ mit $\sigma(x_i) < \sigma(x_{i+1})$ gibt, dann sei i minimal mit dieser Eigenschaft. Nach Wahl von x gilt also $i \geq 2$. Dann ist aber $\{x_{i-1}, x_{i+1}\} \in F_\sigma$ wegen $\sigma(x_i) < \min\{\sigma(x_{i-1}), \sigma(x_{i+1})\}$ und der Definition des Fill-in. Dies ist ein Widerspruch zur Minimalität des Pfades. Also können wir folgern, daß es Pfade $x = x_1 x_2 \ldots x_{j-1} x_j = z$ und $y = y_1 y_2 \ldots y_{\ell-1} y_\ell = z$ in G_σ gibt mit $\sigma(x_i) > \sigma(x_{i+1})$ für $i = 1, \ldots, j-1$ und $\sigma(y_i) > \sigma(y_{i+1})$ für $i = 1, \ldots, \ell - 1$. Wenn wir die beiden Pfade zusammensetzen, erhalten wir $\{x, y\} \in F_\sigma$ im Widerspruch zu Lemma 8.5.10. Also ist (8.6) falsch.
Falls $\sigma(x) < \sigma(y)$, gilt dann

$$\max\{\sigma(v) : v \in A\} = \sigma(x) < \sigma(z) \tag{8.7}$$

für alle $z \in C$.
Wir behaupten nun, daß $C(x) = C$. Dazu ist zu zeigen, daß für alle Knoten $z \in C$ gilt $\{x, z\} \in E \cup F_\sigma$. Sei $x = x_1 x_2 \ldots x_{j-1} x_j = z$ ein kürzester $x - z$-Pfad in G_σ mit der Eigenschaft, daß $x_1, \ldots, x_{j-1} \in A$. Falls dieser Pfad die Länge $j \geq 3$ hätte, müßte nach obiger Argumentation $\sigma(x_i) > \sigma(x_{i+1})$ für $i = 1, \ldots, j-1$ sein. Das ist aber ein Widerspruch zu $\sigma(z) > \sigma(x)$. Also ist $j = 2$ und $\{x, z\} \in E \cup F_\sigma$. □

Satz 8.5.12 *Für einen Graphen G bestimmt der Algorithmus* CS_Baum *einen Cliquenseparatorbaum für G in $\mathcal{O}(n(n+m))$ Rechenschritten.*

Beweis. Sei v der Knoten mit dem niedrigsten σ-Wert, für den $C(v)$ ein Cliquenseparator ist, A die Komponente von $G[V \backslash C(v)]$, die v enthält, und $B := V \backslash (A \cup C(v))$. Wie in (8.7) folgt, daß $\sigma(x) < \sigma(z)$ für alle $x \in A$ und $z \in C(v)$ gilt. Sei $G_1 := G[A \cup C(v)]$ und $G_2 := G[B \cup C(v)]$. σ induziert Ordnungen σ_1 und σ_2 auf G_1 und G_2. Für $i = 1, 2$ und alle Knoten x aus G_i definieren wir

$$C_i(x) := \{w : \sigma(w) > \sigma(x) \text{ und } \{x, w\} \in E \cup F_{\sigma_i}\}.$$

Zunächst zeigen wir, daß G_1 keinen Cliquenseparator besitzt. Nach Lemma 8.5.10 gilt, daß der Fill-in F_{σ_1} des Graphen G_1 zur Ordnung σ_1 gerade $F_{\sigma_1} = F_\sigma \cap (A \cup C(v)) \times (A \cup C(v))$ ist. Für alle $x \in A$ ist $C_1(x) = C(x)$. Es läßt sich nun leicht überprüfen, daß $\sigma(x) < \sigma(v)$ für alle $x \in A \backslash \{v\}$, denn wenn dies nicht der Fall wäre, läge eines dieser x in $C(v)$. Wegen der Minimalität von $\sigma(v)$ kann dann $C_1(x) = C(x)$ für $x \in A$ kein Cliquenseparator in G_1 sein, weil diese Menge sonst schon Cliquenseparator in G wäre. Für $x \in C(v)$ ist $C_1(x) = \{w : w \in C(v) \text{ und } \sigma(w) > \sigma(x)\}$. Wegen der Minimalität von $\sigma(v)$ ist $C(v)$ ein inklusionsminimaler Cliquenseparator, und daher ist G_1 ist nach Entfernen von $C_1(x)$ für $x \in C(v)$ immer noch zusammenhängend. Insgesamt folgt, daß $C_1(x)$ für keinen Knoten aus G_1 Cliquenseparator ist, und deshalb besitzt G_1 nach Lemma 8.5.11 keinen Cliquenseparator.
Da für $x \in B \cup C(v)$ gilt $C_2(x) = C(x)$, muß der Algorithmus die Mengen $C(x)$ nicht mehr

neu berechnen, wenn G durch G_2 ersetzt wird. Die Behauptung des Satzes folgt nun per Induktion nach der Anzahl der Atome im Cliquenseparatorenbaum von G.
Die Berechnung eines MES σ benötigt nach Satz 8.5.9 $\mathcal{O}(nm + n^2)$ Schritte und die von F_σ und der Mengen $C(v)$ $\mathcal{O}(n^2)$ Schritte. Die `FOR`-Schleife im zweiten Teil wird $n - 1$ mal durchlaufen. Die Berechnung der Zusammenhangskomponenten erfordert $\mathcal{O}(n + m)$ Schritte, woraus die angegebene Schranke für die Rechenzeit folgt. □

Wenn wir einen Cliquenseparatorenbaum für einen Graphen G berechnet haben, dann können wir ihn verwenden, um die Probleme CLIQUE, INDEPENDENT SET und GRAPH COLORING für G exakt oder approximativ zu lösen. Dazu genügt es, exakte oder approximative Lösungen für die einzelnen Atome des Cliquenseparatorbaumes zu finden. Da die Separatoren Cliquen sind, können die Teillösungen recht einfach zu einer Gesamtlösung zusammengesetzt werden, was wir im folgenden für jedes der drei obengenannten Probleme einzeln ausführen.

CLIQUE. Eine maximum Clique in einem Graphen muß vollständig in einem einzigen Atom enthalten sein, da Cliquen nicht separiert werden können. Also genügt es, die Atome einzeln zu betrachten.
Eine Clique in einem Graphen entspricht einer unabhängigen Menge im Komplement des Graphen. Falls wir also das Problem CLIQUE lösen wollen, der Cliquenseparatorbaum des betrachteten Graphen jedoch nur wenige Atome besitzt, d.h. das Finden von großen Cliquen nicht deutlich vereinfacht wird, dann können wir versuchen, zum Komplement überzugehen und dort folgendermaßen eine unabhängige Menge suchen.

INDEPENDENT SET. Die Berechnung einer maximum unabhängigen Menge in einem Graphen $G = (V, E)$ reduzieren wir folgendermaßen auf die Berechnung von maximum unabhängigen Mengen in den Atomen.
1. Schritt: Sei $A \cup C$ eine Atom aus dem Cliquenseparatorbaum, wobei C der Cliquenseparator ist. Setze $B := V \backslash (A \cup C)$.
2. Schritt: Bestimme für alle $v \in C$ eine maximum unabhängige Menge $I_A(v)$ in $G[A \cup C]$ mit $v \in I_A(v)$.
Bestimme eine maximum unabhängige Menge I_A in $G[A]$. Entferne $A \cup C$ aus dem Cliquenseparatorbaum.
3. Schritt: Bestimme rekursiv für alle $v \in C$ eine maximum unabhängige Menge $I_B(v)$ in $G[B \cup C]$ mit $v \in I_B(v)$.
Bestimme eine maximum unabhängige Menge I_B in $G[B]$.
4. Schritt: Sei $w \in C$ mit $|I_A(w) \cup I_B(w)| = \max\{|I_A(v) \cup I_B(v)| : v \in C\}$. Falls $|I_A \cup I_B| > |I_A(w) \cup I_B(w)|$, dann ist $I_A \cup I_B$ maximum stabile Menge in G und andernfalls $I_A(w) \cup I_B(w)$.
Die Korrektheit des Verfahrens folgt leicht aus der Tatsache, daß eine unabhängige Menge nur einen Knoten aus der Clique C enthalten kann. Pro Atom sind $\mathcal{O}(n)$ Mengen $I_A(v)$ beziehungsweise I_A zu berechnen. Da ein Cliquenseparatorbaum nach Übung 8.5.2 höchstens $\binom{n}{2}$ Atome enthalten kann, folgt, daß der Algorithmus $\mathcal{O}(n^3)$ Berechnungen von maximum unabhängigen Mengen in den Atomen erfordert.

GRAPH COLORING. Die Färbungen der Knoten der einzelnen Atome können folgendermaßen zusammengesetzt werden.
1. Schritt: Sei $A \cup C$ ein Atom aus dem Cliquenseparatorbaum, wobei C der Cliquenseparator ist. Setze $B := V \backslash (A \cup C)$.

2. Schritt: Bestimme rekursiv eine Färbung χ_B von $G[B \cup C]$ mit einer minimalen Anzahl von Farben.
3. Schritt: Bestimme eine Färbung χ_A von $G[A \cup C]$ mit einer minimalen Anzahl von Farben mit $\chi_A(v) = \chi_B(v)$ für alle $v \in C$.
4. Schritt: Setze $\chi(v) := \chi_A(v)$ beziehungsweise $\chi_B(v)$ je nachdem, ob $v \in A$ oder $c \in B \cup C$.

Nun stellen wir eine Klasse von Graphen vor, für die sich die drei obengennanten Probleme besonders einfach lösen lassen. Die folgende Definition stammt von GAVRIL, der auch für die Probleme CLIQUE und GRAPH COLORING polynomielle Algorithmen auf dieser Graphenklasse vorstellte (vgl. [Gav77]).

Definition 8.5.13 *Ein Graph G heißt* cliquenseparabel, *falls die Atome eines Cliquenseparatorbaumes für G zu einem der beiden folgenden Typen $\mathcal{T}_1$ und $\mathcal{T}_2$ von Graphen gehören.*

$$\begin{aligned}\mathcal{T}_1 &:= \{G=(V,E): \text{ Es gibt eine Partition } V = V_1 \cup V_2, \text{ so daß } G[V_1] \text{ bipartit}\\ &\qquad \text{und } G[V_2] \text{ Clique ist und } \{v,w\} \in E \text{ für alle } v \in V_1 \text{ und } w \in V_2 \text{ gilt.}\}\\ \mathcal{T}_2 &:= \{G=(V,E): G \text{ ist vollständig } k\text{-partit, d.h. } \exists \text{ eine Partition } V = V_1 \cup \ldots \cup V_k,\\ &\qquad \text{so daß } \{v,w\} \in E \text{ genau dann, wenn es } i \neq j \text{ gibt mit } v \in V_i \text{ und } w \in V_j.\}\end{aligned}$$

Übung 8.5.14 *Graphen vom Typ 1 und 2 können in $\mathcal{O}(n^2)$ Zeit erkannt werden.*

Da ein Cliquenseparatorbaum höchstens $\binom{n}{2}$ Atome enthält, impliziert Satz 8.5.12 das folgende Korollar.

Korollar 8.5.15 *a) Cliquenseparable Graphen können in $\mathcal{O}(n^4)$ Zeit erkannt werden.*
b) Die Probleme CLIQUE, INDEPENDENT SET *und* GRAPH COLORING *besitzen polynomielle Algorithmen für die Klasse der cliquenseparablen Graphen.*

TARJAN [Tar85] behandelt außerdem die Graphenklasse der sogenannten EPT-Graphen, die zuerst in [GJ85] eingeführt wurde und folgendermaßen definiert werden kann. Sei B ein Baum. Dann ist die Knotenmenge eines EPT Graphen von B eine Menge von Pfaden in B, wobei zwei Knoten adjazent sind, wenn die beiden jeweiligen Pfade mindestens eine Kante gemeinsam haben. Einfache Beispiele von EPT-Graphen sind Linegraphen.

Übung 8.5.16 *Ein Graph ist ein Linegraph genau dann, wenn er der EPT-Graph eines Sterns, $K_{1,a}$ ist.*

Auf EPT-Graphen können mit Hilfe von Cliquenseparatorbäumen CLIQUE und INDEPENDENT SET exakt und GRAPH COLORING approximativ mit einer Güte von $\frac{3}{2}$ gelöst werden.

Weiterhin sind Cliquenseparatorbäume hilfreich bei der Erkennung von $K_{1,3}$-freien perfekten Graphen. Dabei heißt in diesem Zusammenhang ein Graph H-frei, wenn er keine induzierte Kopies von H besitzt. CHVÁTAL und SBIHI [CS88] zeigten, wie in polynomieller Zeit überprüft werden kann, ob $K_{1,3}$-freier Graph ohne Cliquenseparator BERGE ist, d.h. $C_{2\ell+1}$- und $\bar{C}_{2\ell+1}$-frei für alle $\ell \geq 2$. Um zu testen, ob ein $K_{1,3}$-freier Graph BERGE ist, können wir also zunächst einen Cliquenseparatorbaum für G bestimmen und dann diese Eigenschaft für jedes Atom überprüfen, denn ein induzierter $C_{2\ell+1}$ oder $\bar{C}_{2\ell+1}$ muß offenbar vollständig in einem Atom enthalten sein. PARTHASARATHY und RAVINDRA [PR76]

zeigten, daß die starke Perfekte-Graphen-Vermutung (vgl. Abschnitt 7.6) für $K_{1,3}$-freie Graphen korrekt ist. Also ist ein $K_{1,3}$-freier Graph genau dann perfekt, wenn er BERGE ist, und daher gilt:

Satz 8.5.17 [CS88] *$K_{1,3}$-freie Graphen können in polynomieller Zeit auf Perfektheit überprüft werden.*

Kapitel 9

Baumweite

Viele NP-vollständige Probleme werden effizient lösbar oder sogar trivial, wenn man sich auf die Klasse der Bäume einschränkt. Das Problem INDEPENDENT SET beispielsweise haben wir in Abschnitt 1.5 auf Bäumen selbst in seiner gewichteten Form durch einen dynamic-programming-Ansatz in linearer Zeit gelöst. In diesem Abschnitt betrachten wir eine Klasse von Graphen mit Baum-ähnlicher Struktur.

9.1 Baumweite und k-Bäume

Definition 9.1.1 (ROSE 1974) *Für $k \in \mathbb{N}$ ist die Menge der* k-Bäume *induktiv wie folgt definiert:*

- *Der K_{k+1} ist ein k-Baum.*
- *Ist G ein k-Baum und C eine k-Clique in G, so ist auch der Graph ein k-Baum, der aus G durch Hinzufügen eines neuen Knotens entsteht, der vollständig mit C verbunden wird.*

Ein Graph heißt Sub-k-Baum,[1] *falls er Subgraph eines k-Baums für ein $k \in \mathbb{N}$ ist.*

Die 0-Bäume sind offenbar gerade die leeren Graphen. Für $k \geq 1$ sind k-Bäume jedoch zusammenhängend. 1-Bäume sind zudem kreisfrei, d.h. Bäume. Da umgekehrt jeder Baum ein Blatt besitzt, entspricht die Klasse der 1-Bäume gerade der Menge der Bäume auf mindestens zwei Knoten, und die Sub-1-Bäume sind gerade die Wälder. Kreise sind Sub-2-Bäume, aber keine Sub-1-Bäume (Übung). Offenbar ist jeder Sub-k-Baum auch ein Sub-$(k+1)$-Baum. Jeder Subgraph (auf mindestens $k+1$ Knoten) eines Sub-k-Baums ist ein Sub-k-Baum. Jeder Graph auf n Knoten ist ein Sub-$(n-1)$-Baum. Man überlegt sich ferner, daß ein Sub-k-Baum auf mindestens $k+1$ Knoten schon Subgraph eines k-Baums auf derselben Knotenmenge ist, d.h. durch Einfügen von Kanten zu einem k-Baum erweitert werden kann.

Proposition 9.1.2 (ROSE 1974)
Ein k-Baum ist ein zusammenhängender, chordaler Graph mit $\omega(G) = k+1$.

[1] engl. partial k-tree. Beachte: in der Literatur wird der K_k zumeist als k-Baum definiert

Beweis. Klar für vollständige Graphen. Entsteht ein k-Baum G aus einem K_{k+1} durch sukzessives Anheften von Knoten $v_{k+2}, \ldots, v_n$ (d.h. sie werden jeweils vollständig mit bereits vorhandenen k-Cliquen verbunden), so bilden diese Knoten in umgekehrter Reihenfolge gelesen offenbar ein perfektes Knoteneliminationsschema für G. Also sind k-Bäume nach Satz 7.2.9 chordal. Betrachtet man den zuletzt eingefügten Knoten einer kardinalitätsmaximalen Clique in G, ersieht man auch $\omega(G) \leq k+1$. □

Umgekehrt entnimmt man Übung 7.2.14 sofort, daß jeder chordale Graph mit $\omega(G) = k+1$ ein Sub-k-Baum ist. Wir folgern daraus:

Lemma 9.1.3 *Ein k-Baum kann von jeder $(k+1)$-Clique aus konstruiert werden.*

Beweis. Ein Knoteneliminationsschema, das eine Clique C' am Schluß anordnet, stellt ein Konstruktionsschema für G dar, da ein k-Baum für einen chordalen Graphen mit $\omega(G) = k+1$ maximal viele Kanten hat. □

Definition 9.1.4 *Die* Kontraktionscliquenzahl $\omega_c(G)$ *eines Graphen G ist die größte Zahl $k \in \mathbb{N}$, so daß G einen K_k-Minor besitzt:*

$$\omega_c(G) := \max\{k \in \mathbb{N} : G \succ K_k\}.$$

Während $\omega_c(G)$ i.a. die Cliquenzahl $\omega(G)$ fast immer um einen Faktor $\Omega\left(\frac{n}{(\log n)^{3/2}}\right)$ übersteigt [BCE80], gilt für k-Bäume:

Proposition 9.1.5 *Ein k-Baum G erfüllt* $\omega_c(G) = \omega(G)$.

Beweis. Sei $G = (V, E)$ ein knotenminimaler k-Baum mit $\omega_c(G) > \omega(G)$. $S_1, \ldots, S_{k+2}$ seien disjunkte Teilmengen von V, so daß durch elementare Kantenkontraktionen innerhalb der S_i ein K_{k+2} auf der Knotenmenge $S := \{S_i\}_{i=1}^{k+2}$ entsteht. Sei $\sigma = (v_1, \ldots, v_n)$ ein perfektes Knoteneliminationsschema für G und v der erste Knoten in σ, der zu S gehört. v hat damit Grad höchstens k in $G[S]$, und ist $v \in S_j$, $j \in \{1, \ldots, k+2\}$, so folgt $|S_j| \geq 2$. Da $G[S_j]$ zusammenhängt, hat v noch mindestens einen Nachbarn in S_j. Daher ist für alle $i \neq j$ der Schnitt $\langle S_i, S_j\rangle := \{\{x, y\} \in E : x \in S_i \wedge y \in S_j\}$ nichtleer genau dann, wenn der Schnitt $\langle S_i, S_j \setminus \{v\}\rangle$ nichtleer ist. Hieraus folgt $G - v \succ K_{k+2}$, Widerspruch. □

Wie sich in Satz 9.1.9 manifestieren wird, lassen sich Sub-k-Bäume auch ganz anders ansehen. Robertson und Seymour führten im Zusammenhang mit dem „Graph Minors Project" den Begriff der Baumzerlegung und der Baumweite ein [RS83, RS86]. Er ist eine Verallgemeinerung der Charakterisierung chordaler Graphen als Schnittgraphen von Familien von Teilbäumen eines Baumes, vgl. Satz 7.2.19.

Definition 9.1.6 *Eine* Baumzerlegung *eines Graphen $G = (V, E)$ ist ein Paar $\langle \mathcal{S}, \mathcal{B}\rangle$, wobei $\mathcal{S} = \{S_i : i \in I\}$ eine Familie von Teilmengen von V ist und $\mathcal{B}$ ein Baum auf I, so daß die folgenden Bedingungen erfüllt sind:*

(1) Die Mengen S_i, $i \in I$, überdecken ganz V: $\bigcup_{i \in I} S_i = V$;

(2) $\forall \{u, v\} \in E\ \exists i \in I : \{u, v\} \subseteq S_i$;

(3) für jeden Knoten $v \in V$ induziert die Menge $\{i \in I : S_i \ni v\}$ einen Teilbaum von $\mathcal{B}$.

Die Weite einer Baumzerlegung ist $\max_{i \in I} |S_i| - 1$. *Die* Baumweite *eines Graphen G ist das minimale $w \geq 0$, so daß G eine Baumzerlegung der Weite w besitzt.*

Die ersten beiden Bedingungen besagen, daß die Teilgraphen $G[S_i]$ den Graphen G überdecken. Die dritte Bedingung wird oftmals äquivalent auch so formuliert:

(3') $\forall i,j \in I$: ($k \in I$ ist ein innerer Knoten des $i-j$–Wegs in $\mathcal{B} \Rightarrow S_k \supseteq S_i \cap S_j$).

Bei chordalen Graphen kann $\mathcal{S}$ gerade als die Familie der maximalen Cliquen gewählt werden, siehe Satz 7.2.19. Genau die leeren Graphen haben Baumweite 0.

Lemma 9.1.7 *Ein nicht-leerer Wald hat Baumweite 1.*

Beweis. Ist $T = (V,E)$ ein Baum, so setze man $I := E$ und $S_e := \{x,y\}$, falls $e = \{x,y\} \in E$, und definiere den Baum $\mathcal{B}$ auf E wie folgt: Man wähle ein Blatt $b \in V$. $\{a,b\}$ sei die mit b in T inzidierende Kante. Nun laufe man T gemäß Breadth-First-Search mit Wurzel b ab. Wann immer man über eine Kante $\{u,v\} \neq \{a,b\}$ auf einen neuen Knoten $v \in V \setminus \{a,b\}$ stößt, verbinde man $\{u,v\}$ in $\mathcal{B}$ mit der Kante, über die man u erreichte.
Für einen Wald W aus Bäumen T_j, $j = 1,..,w$, fügt man zwischen den Bäumen $\mathcal{B}_j$ der Baumzerlegungen der Bäume T_j, $j > 1$, und $\mathcal{B}_1$ beliebig je eine Kante ein und erhält einen Baum $\mathcal{B}$ für W. □

Die Stärke der Bedingung (3) aus Definition 9.1.6 macht die folgende Aussage klar.

Lemma 9.1.8 *Für einen Graphen G mit Baumweite höchstens k gilt:* $\omega(G) \leq k+1$.

Beweis. Sei $G = (V,E)$ ein Graph mit Baumweite höchstens k, $\langle \mathcal{S}, \mathcal{B} \rangle$ eine Baumzerlegung der Weite höchstens k für G und $C \subseteq V$ eine Clique in G. Es genügt offenbar, zu zeigen, daß es eine Menge $S_i \in \mathcal{S}$ gibt, so daß $C \subseteq S_i$. Dies gilt aber, da die Teilbäume $T_v := \mathcal{B}[\{i \in I : S_i \ni v\}]$, $v \in C$, der HELLY-Bedingung genügen, so daß es ein $i \in I$ gibt mit $i \in \bigcap_{v \in C} V(T_v)$. □

Die Begriffe Baumweite und k-Baum hängen nun wie folgt zusammen.

Satz 9.1.9 (SCHEFFLER 1987; WIMER 1987)
Ein Graph hat Baumweite $\leq k$ genau dann, wenn er ein Sub-k-Baum ist.

Beweis. "$\Leftarrow$": Da die Baumweite höchstens abnimmt, wenn wir zu einem Subgraphen übergehen, sei der Graph $G = (V,E)$ ohne Einschränkung ein k-Baum. Wir beweisen durch Induktion nach $|V|$, daß G eine Baumzerlegung $\langle \{S_i\}_{i \in I}, \mathcal{B} \rangle$ der Baumweite höchstens k besitzt, in der jede k-Clique von G in einem S_i enthalten ist [vergleiche den Beweis von Satz 7.2.19, (i) $\Rightarrow$ (ii)]. Der Fall $|V| = k+1$ ist trivial: man nehme $\mathcal{B} = K_1$. Andernfalls sei $v \in V$ ein Simplizialknoten in G. Wegen $d(v) \geq k$ folgt, daß $\Gamma(v)$ eine k-Clique in G ist. Nach Induktionsvoraussetzung besitzt $G - v$ eine Baumzerlegung $\langle \mathcal{S}, \mathcal{B} \rangle$ der Baumweite höchstens k mit $\Gamma(v) \subseteq S_a$ für ein $a \in I$. Hänge in $\mathcal{B}$ an den Knoten a ein Blatt b an mit $S_b := \Gamma(v) + v$. Dann erfüllt $\langle \mathcal{S} + S_b, \mathcal{B} + b \rangle$ die Bedingungen (1),(2) und (3) einer Baumzerlegung von G [zu (3): nach Induktionsvoraussetzung genügt es, den Fall $j = b$ zu betrachten. Hier wiederum ist der Fall $k = a$ trivial; der Fall $k \neq a$ folgt aufgrund der Konstruktion und der Induktionsvoraussetzung: $S_i \cap S_b \subseteq S_i \cap (S_b - v) \subseteq S_i \cap S_a \subseteq S_k$].
„$\Rightarrow$": Ohne Einschränkung sei $G = (V,E)$ ein Graph mit Baumweite $= k$ und $\langle \{S_i\}_{i \in I}, \mathcal{B} \rangle$ eine entsprechende Baumzerlegung. Wir geben einen chordalen Graphen H mit $\omega(H) = k+1$ an, der G als Subgraphen enthält. Wie oben gesehen, ist H dann ein Sub-k-Baum

(vgl. Übung 7.2.14) und damit auch G.
Betrachte den Graphen $H = (V, F)$ definiert durch

$$F := \left\{ \{v, w\} \in \binom{V}{2} : \exists i \in I \text{ mit } \{v, w\} \subseteq S_i \right\}.$$

Dann folgt nach Bedingung (2) einer Baumzerlegung, daß $E \subseteq F$. Aus Lemma 9.1.8 folgt $\omega(H) \leq k+1$; da G Baumweite $= k$ hat, gilt mithin $\omega(H) = k+1$ Die Chordalität von H folgt unmittelbar aus Satz 7.2.19, da die Bäume $T_v := \mathcal{B}[\{i \in I : S_i \ni v\}]$, $v \in V$, eine Schnittgraphdarstellung von G definieren. □

Aus Satz 9.1.9 leiten wir nun eine Reihe einfacher, aber aufschlußreicher Folgerungen ab.

Korollar 9.1.10 *Die Graphen mit Baumweite 1 sind gerade die nicht-leeren Wälder. Kreise haben Baumweite 2.*

(Man beweise dies zur Übung ohne Zuhilfenahme von Satz 9.1.9 und damit Satz 7.2.19.)

Korollar 9.1.11 *Jeder Graph mit Baumweite $\leq k$ besitzt eine Baumzerlegung $\langle \mathcal{S}, \mathcal{B} \rangle$, so daß $\mathcal{B}$ höchstens $n-k$ Knoten hat.* □

Da jeder k-Baum einen Knoten vom Grad höchstens k besitzt, gilt:

Korollar 9.1.12 *Jeder Graph mit Baumweite höchstens k ist $(k+1)$-choosable (d.h. insbesondere $(k+1)$-färbbar).*

Weiter erhalten wir in Verallgemeinerung der Aussage, daß Graphen mit Baumweite ≤ 1 keinen Kreis enthalten, aus Satz 9.1.9 zusammen mit Proposition 9.1.5 das

Korollar 9.1.13 *Ein Graph mit Baumweite $\leq k$ besitzt keinen K_{k+2}-Minor.*[2] □

Wie im Fall $k = 1$ charakterisiert dies auch Graphen mit Baumweite ≤ 2 bereits:

Satz 9.1.14 (Dirac 1960; Wald, Colbourn 1983)
Ein Graph hat Baumweite ≤ 2 genau dann, wenn er keinen K_4-Minor besitzt.

Beweis. Es bleibt die Rückrichtung zu zeigen. Sei $G = (V, E)$ ein Graph ohne K_4-Minor. Da der $K_4 - e$ ein 2-Baum ist, ist die Aussage für $|V| \leq 4$ sicherlich richtig. Sei also $|V| \geq 5$. Wir unterscheiden nach der Zusammenhangszahl $\kappa(G)$.
Ist G unzusammenhängend, so enthalten selbstverständlich auch seine Komponenten keinen K_4-Minor und besitzen nach Induktionsannahme eine Baumzerlegung. Diese setzt man wie in Lemma 9.1.7 zu einer solchen für G zusammen.
$\kappa(G) = 1$: Sei v ein Artikulationsknoten in G. Seien H_i, $i = 1..k$, die Komponenten von $G - v$ jeweils vereinigt mit v. Jedes H_i ist K_4-Minor-frei. Nach Induktionsannahme besitzen sie also Baumzerlegungen $\mathcal{B}_i$. Sei $S_v^{(i)}$ ein beliebiger Knoten in $\mathcal{B}_i$, der v enthält. Füge wieder zwischen $S_v^{(i)}$, $i > 1$, und $S_v^{(1)}$ sternförmig Kanten ein.
$\kappa(G) = 2$: Seien die Knoten x und y trennend in G. Sind C_i die Komponenten von $G[V \setminus \{x, y\}]$, so enthalten auch die Graphen $H_i := G[C_i \cup \{x, y\}]$ keinen K_4-Minor und besitzen daher nach Induktionsvoraussetzung eine Baumzerlegung $\mathcal{B}_i$. Verfahre wie oben mit Knoten $S_{xy}^{(i)}$ in $\mathcal{B}_i$, die x und y enthalten.

[2] Hat ein Graph dagegen Baumweite $\geq k^{3/2}\sqrt{n}$, so enthält er einen K_k-Minor [AST90b].

$\kappa(G) \geq 3$: Dieser Fall kann nicht auftreten. Seien u und v zwei nicht benachbarte Knoten. Nach den Sätzen 2.2.1 und 2.2.3 von MENGER bzw. WHITNEY existieren drei knotendisjunkte u-v-Wege P_i, $i = 1, 2, 3$. Auf mindestens zwei Wegen gibt es innere Knoten. Da $G[V \setminus \{u, v\}]$ noch zusammenhängt, gibt es zwei innere Knoten x und y auf verschiedenen P_i, die durch einen Weg verbunden sind, der den dritten Weg nicht kreuzt. Also enthält G eine Unterteilung des K_4 auf den Knoten u, v, x und y. □

Anwendungen. (1) Sei G ein Graph ohne K_4-Minor. Dann enthält G höchstens $2n - 3$ viele Kanten, und diese Schranke ist scharf für 2-Bäume. Da sowohl der K_5 als auch der $K_{3,3}$ den K_4 als Minor enthalten, sind Graphen mit Baumweite ≤ 2 planar.
(2) Ein Graph G mit $\delta(G) \geq 3$ enthält den K_4 als Minor.
(3) Da ein Graph mit $\chi(G) \geq 4$ einen kritisch 4-chromatischen, induzierten Subgraphen enthält (vgl. Übung 5.1.2), ergibt dies einen neuen Beweis der HADWIGER-Vermutung im Fall $n = 4$ (vgl. Satz 6.3.21), der zudem verdeutlicht, daß – während (H_5) äquivalent zum 4-Farben-Satz ist – für (H_4) die Färbbarkeit nicht die tragende Rolle spielt.

Als vierte Anwendung dieses Satzes finden wir:

Korollar 9.1.15 *Es gibt einen linearen Algorithmus, um Sub-2-Bäume zu erkennen und sie in einen 2-Baum einzubetten.*

Beweis. Nach Satz 9.1.14 kann man sich beim Erkennungsproblem auf 2-zusammenhängende Graphen beschränken. Es gelte also $\delta(G) \geq 2$. Ein 2-Baum besitzt mindestens einen Knoten vom Grad 2 (den zuletzt angehängten). Gilt also $\delta(G) \geq 3$, so ist G kein Sub-2-Baum. Andernfalls enthält G mindestens einen Knoten x vom Grad genau 2. Falls die beiden Nachbarn u und v von x keine Kante bilden, so füge diese Kante zu G hinzu. Dann enthält $G - x + \{u, v\}$ genau dann einen K_4-Minor, wenn G einen solchen enthält. Iteration dieses Verfahrens endet also genau dann in einem K_3, wenn G ein Sub-2-Baum war.
Der sich aus diesem Beweis ergebende Algorithmus, um einen G enthaltenden 2-Baum zu konstruieren, testet zunächst, ob G höchstens $2n - 3$ viele Kanten enthält, berechnet dann aus der Adjazenzliste die Knotengrade von G und hält sich eine Queue aller Knoten vom Grad 2 (im Restgraphen). Der Update der Knotengrade und der Queue beim „Vergessen“ des Knotens x ist mittels einer zusätzlichen Zeigerliste sicherlich in konstanter Zeit durchführbar, die Laufzeit des Algorithmus also linear.
Falls G nicht 2-zusammenhängend ist, so setzt man die gewonnenen 2-Bäume für die Blöcke von G ohne Schwierigkeiten zu einem G-enthaltenden 2-Baum zusammen. □

Schon Graphen mit Baumweite ≤ 3 lassen sich nicht mehr analog dadurch charakterisieren, daß sie keinen K_5-Minor enthalten. Der $K_{2,2,2}$ beispielsweise besitzt keinen K_5-Minor (da planar), ist aber auch kein Sub-3-Baum (wegen $\delta(K_{2,2,2}) = 4$). Eine Minor-Charakterisierung von Sub-3-Bäumen findet sich in [APC90, ST90], ein Erkennungsalgorithmus vermittels einer sicheren Reduktionsregel wie oben bei den Sub-2-Bäumen in [AP86].

Leider kennt man keine große Graphenklasse mit kleiner Baumweite, die nicht über den Begriff der Baumweite definiert wäre. Wir wollen in diesem Zusammenhang nur die beiden folgenden Resultate erwähnen.

Wie die nächste Übung zeigt, folgt aus Satz 8.4.4 unmittelbar, daß Graphen von kleiner Baumweite auch kleine Separatoren besitzen.

Übung 9.1.16 *Ein Graph mit Baumweite $\leq k$ besitzt einen $(k + 1, \frac{1}{2})$-Separator.*

Aus Proposition 8.2.2 folgt mithin:

Korollar 9.1.17 *Es gibt planare Graphen beliebig hoher Baumweite.* □

Planaren Graphen sind von besonderer Bedeutung für die Baumweite, denn sie sind gewissermaßen sogar schon die einzige Ursache für große Baumweite, wie das folgende, tiefliegende Resultat zeigt (für einen Beweis siehe auch [SW89]):

Satz 9.1.18 (ROBERTSON, SEYMOUR 1984) *Sei $\mathcal{C}$ eine bezüglich Minorbildung abgeschlossene Graphenklasse. Wenn $\mathcal{C}$ nicht alle planaren Graphen enthält, dann ist $\mathcal{C}$ von beschränkter Baumweite.* □

Anmerkung und Übungen

Nach Übung 6.1.30 hat ein Graph keinen K_4-Minor genau dann, wenn er keine Unterteilung des K_4 besitzt. Die Graphen mit Baumweite ≤ 2 sind daher gerade die sogenannte *reihe-parallel* Graphen nach DUFFIN [Duf65].

Übung 9.1.19 *Kreisartig planare Graphen haben Baumweite ≤ 2* [vgl. Übung 6.1.35].

Übung 9.1.20 [CLR87] *Ein Graph G mit Baumweite ≤ 2 hat Buchdicke $bth(G) \leq 2$.* [vgl. Proposition 6.6.9]

Übung 9.1.21 (WIMER 1987) HALIN-*Graphen (vgl. Übung 2.4.12) sind (planare) Sub-3-Bäume.*

9.2 Optimierung auf Graphen beschränkter Baumweite

Alle Optimierungsalgorithmen auf Graphen beschränkter Baumweite basieren darauf, daß eine Baumzerlegung des untersuchten Graphen gegeben ist. Das Problem, die Baumweite eines Graphen zu bestimmen, ist jedoch NP-schwer [ACP87].[3] Beachte: wir konnten zwar für chordale Graphen in linearer Zeit einen Cliquenbaum konstruieren, doch ist das Problem, zu einem nicht chordalen Graphen $G = (V, E)$ eine kleinste Menge von Nicht-Kanten $F \subseteq \binom{V}{2} \setminus E$ zu finden, so daß der Graph $(V, E \cup F)$ chordal ist, ebenfalls NP-vollständig [Yan81b].

Für festes $k \in \mathbb{N}$ gibt es allerdings einen sogar linearen Algorithmus, der Graphen mit Baumweite $\leq k$ erkennt und gegebenenfalls eine entsprechende Baumzerlegung konstruiert [Bod93a] (leider ist dieser Algorithmus jedoch reichlich komplex und die Konstante sehr groß). Mit Hilfe dieser Baumzerlegung lassen sich dann durch einen Dynamische-Programmierung-Ansatz viele NP-schwere Optimierungsprobleme effizient (oft sogar in linearer Zeit) lösen. Da jeder Graph auf n Knoten Baumweite $\leq n-1$ hat, muß die Laufzeit eines solchen Algorithmus allerdings exponentiell in der Baumweite sein (falls P $\neq$ NP). Zu den Problemen, die eingeschränkt auf Graphen mit beschränkter Baumweite polynomiell lösbar werden, gehören u.a. CHROMATIC NUMBER, CLIQUE PARTITION, CLIQUE, INDEPENDENT SET, VERTEX COVER, MAX CUT, DOMINATING SET, HAMILTONIAN CIRCUIT, CHROMATIC INDEX, STEINER TREE, DISJOINT PATHS, siehe [AP89, Sch89a]. Tatsächlich konnte man sehr allgemeine, z.B. aussagenlogisch definierte Problemklassen angeben, deren Probleme sich auf Graphen von beschränkter Baumweite durch einen generischen Algorithmus in linearer Zeit lösen lassen, siehe [BLW87, Bod88, Sch89a, ALS91, CM93, Bor95].

[3] Es gibt aber immerhin einen Approximationsalgorithmus mit Güteratio $\mathcal{O}(\log n)$ [BGHK95].

Um die Technik der Dynamischen Optimierung auf Graphen mit beschränkter Baumweite zu illustrieren, greifen wir nochmals das Problem INDEPENDENT SET auf und verallgemeinern den Algorithmus für Bäume aus Abschnitt 1.5.

Satz 9.2.1 [AP89] *Es gibt einen linearen Algorithmus, der für Graphen mit beschränkter Baumweite die Stabilitätszahl $\alpha(G)$ sowie eine maximum stabile Menge berechnet.*

Beweis. Wir skizzieren einen Algorithmus, der für Graphen G mit Baumweite $\leq k$ bei gegebener Baumzerlegung $\langle \mathcal{S}, \mathcal{B} \rangle$ in Zeit $\mathcal{O}(2^k n)$ den Parameter $\alpha(G)$ berechnet und eine maximum-stabile Menge in G konstruiert.
Dazu werden ein beliebiger Knoten $w \in I$ von $\mathcal{B}$ als Wurzel ausgezeichnet und die Kanten von $\mathcal{B}$ zur Wurzel gerichtet. Für einen Knoten $i \in I \setminus \{w\}$ sei der Nachfolger von i in $\mathcal{B}$ mit $\nu(i)$ bezeichnet. Ein Knoten $i \in I$ induziert einen Teilbaum B_i von $\mathcal{B}$. Für $i \neq w$ heißt der Graph $G[\bigcup_{j \in B_i} S_j]$ der Zweig G_i der Baumzerlegung. Beachte, daß er aufgrund der Bedingung (3.) der Definition einer Baumzerlegung nur über die Knoten $C_i := S_{\nu(i)} \cap S_i$ mit dem übrigen Graphen kommuniziert. Wir bestimmen daher eine kardinalitätsmaximale stabile Menge in G, indem wir zunächst alle stabilen Mengen in $G[S_w]$ auflisten (das sind höchstens 2^{k+1} viele) und dann für jeden Zweig G_i, $i \in \Gamma^-(w)$, nachsehen, wie sich die Einschränkungen der stabilen Mengen auf $C_i = S_w \cap S_i$ größtmöglich in dem Zweig fortsetzen lassen. In jedem Zweig G_i wiederum gehen wir genauso vor, außer daß uns nun nicht mehr die Fortsetzungen aller stabilen Mengen von S_i interessieren, sondern nur noch aller stabilen Mengen in C_i. Wir speichern uns daher in einer Tabelle T für jeden Zweig G_i und für jede stabile Menge c in $C_i = S_{\nu(i)} \cap S_i$ die maximale stabile *Fortsetzung* (d.h. ohne c selbst!) von c in diesem Zweig im Eintrag $T^i[c]$. Aufgebaut wird die Tabelle von den Blättern her: in einem Blatt S_i ist die maximale Fortsetzung s einer stabilen Menge $c \subseteq C_i$ lediglich in $S_i \setminus C_i$ zu suchen - in einem inneren Knoten i des Baumes $\mathcal{B}$ hingegen hat man jede Fortsetzung s von c in S_i mit allen maximalen und konsistenten Fortsetzungen von $s \cup c$ in den darunter liegenden Zweigen G_j, $j \in \Gamma^-(i)$, zu vereinen und dann die größte als die maximale Fortsetzung von c im Zweig G_i auszuwählen:

```
PROCEDURE FILL_TABLE (i, C);

BEGIN
  {Schritt 1: Initialisierung}
  {→ Tabelle IS der unabhängigen Mengen von S_i aufbauen}
  FOR c ⊆ C : c stabil DO
    FOR s ⊆ S_i \ C : G[c ∪ s] stabil DO BEGIN
      IS[c,s].MaxIS   := s;
      IS[c,s].MaxSize := |s|;
  END;

  {Schritt 2: Informationen aus den Sub-Zweigen zusammentragen}
  {→ die maximalen Erweiterungen in jedem Sub-Zweig an IS anhängen}
  FOR j ∈ Γ⁻(i) DO BEGIN
    FILL_TABLE (j, S_i ∩ S_j);
    FOR c ⊆ C : c stabil DO
      FOR s ⊆ S_i \ C : G[c ∪ s] stabil DO BEGIN
        IS[c,s].MaxIS   := IS[c,s].MaxIS   ∪ T^j[(c ∪ s) ∩ S_j].MaxIS;
        IS[c,s].MaxSize := IS[c,s].MaxSize + T^j[(c ∪ s) ∩ S_j].MaxSize;
    END;
  END;

  {Schritt 3: Optimierung}
  {→ die maximalen Erweiterungen im Zweig G_i berechnen}
  FOR c ⊆ C : c stabil DO BEGIN
    T^i[c].MaxIS   := IS[c,∅].MaxIS;
    T^i[c].MaxSize := IS[c,∅].MaxSize;
    FOR ∅ ≠ s ⊆ S_i \ C : G[c ∪ s] stabil DO
      IF IS[c,s].MaxSize > T^i[c].MaxSize THEN BEGIN
        T^i[c].MaxIS   := IS[c,s].MaxIS;
        T^i[c].MaxSize := IS[c,s].MaxSize;
    END;
  END;

END; {Fill_Table}
```

Damit bestimmt sich nach einem Aufruf der Prozedur der Form FILL_TABLE (w, S_w) der Parameter $\alpha(G)$ durch

$$\alpha(G) := \max \{ |c| + T^w[c].MaxSize : c \subseteq S_w \text{ stabil } \}$$

und für einen entsprechenden Eintrag c von T^w ist

$$c \cup T^w[c].MaxIS$$

eine maximum-stabile Menge in G.
Bezüglich der Laufzeit überlegt man sich, daß ein Aufruf der Prozedur im ersten und dritten Schritt jeweils nur höchstens $\mathcal{O}(2^{k+1})$, also konstant viele Operationen zur Folge hat. Da die Prozedur für jedes $i \in I$ einmal aufgerufen wird und die Größe $|I|$ des Baumes $\mathcal{B}$ als linear in der Eingabegröße angenommen werden kann (vgl. Korollar 9.1.11), ergibt

dies linear viele Operationen im ersten und letzten Schritt. Im zweiten Schritt werden die Informationen eines Zweiges jeweils von der Wurzel des Zweiges an deren Nachfolger weitergereicht. Da der Baum $\mathcal{B}$ gerade $|I|-1$ Kanten hat, hat auch dieser Abschnitt nur insgesamt linear viele Operationen zur Folge. □

Übung 9.2.2 *Auf ähnliche Art entwickle man einen polynomiellen Algorithmus, um die chromatische Zahl eines Graphen mit Baumweite $\leq k$ zu bestimmen und eine $\chi(G)$-Färbung zu konstruieren.*

9.3 Randomisierung I: Subgraph Isomorphismus

Randomisierte Algorithmen. Randomisierte Algorithmen dürfen quasi eine Münze werfen, um zwischen verschiedenen Alternativen zu entscheiden. Sie finden zwar nicht immer eine Lösung oder versagen manchmal einfach, doch ist die Wahrscheinlichkeit eines Versagens (bezüglich der Münzwürfe, d.h. unabhängig von der Instanz) beschränkt, so daß sie sich durch Wiederholung des Algorithmus unter derselben Eingabe schnell unter die Wahrscheinlichkeit beispielsweise eines Hardwarefehlers drücken läßt. Beim Primzahlproblem [Rab76, SS77, AH87] (von dem man allerdings nicht weiß, ob es NP-schwer ist), bei parallelen Algorithmen für maximum Matching [MVV87] (wobei man hier nicht weiß, ob das Problem P-vollständig ist) sowie bei (u.a. graphentheoretischen) Zählproblemen [Sin93, Wel93] konnten hiermit große Erfolge erzielt werden. Doch selbst in Fällen, wo man effiziente deterministische Algorithmen kennt, bestechen randomisierte Algorithmen oft durch ihre einfache Struktur. Wir wollen dies an dem folgenden Problem exemplarisch demonstrieren. Das SUBGRAPH ISOMORPHISM Problem lautet: gegeben Graphen G und H – enthält G einen zu H isomorphen (schwachen) Subgraphen, d.h. gibt es eine Injektion $f : V(H) \to V(G)$ mit $\{u,v\} \in E(H) \Rightarrow \{f(u),f(v)\} \in E(G)$? Dieses Problem verallgemeinert so unterschiedliche Probleme wie CLIQUE, HAMILTONIAN CIRCUIT, HAMILTONIAN PATH und MATCHING; insbesondere ist es also NP-vollständig. Für festes H ist das Problem jedoch polynomiell lösbar: vollständige Enumeration aller Injektionen $f : V(H) \to V(G)$ liefert offenbar einen $\mathcal{O}(\binom{k}{2} n^k)$-Algorithmus, wobei $k := |V(H)|$ (eine Konstante) und $n = |V(G)|$. Wir geben in diesem Abschnitt einen randomisierten Algorithmus für SUBGRAPH ISOMORPHISM mit Laufzeit $2^{\mathcal{O}(k)} n^{t+1}$, wo $t = t(H)$ die Baumweite von H bezeichnet. SUBGRAPH ISOMORPHISM ist also polynomiell lösbar für alle Graphen H der Ordnung $\mathcal{O}(\log n)$ von beschränkter Baumweite. Der Status des Problems SUBGRAPH ISOMORPHISM eingeschränkt auf (beliebige) Graphen H der Ordnung $\mathcal{O}(\log n)$ ist allerdings unbekannt, ebenso wie der des Problems LOG CLIQUE: „gegeben ein Graph G, enthält G eine Clique der Größe $\log n$?“.

Satz 9.3.1 (ALON, YUSTER, ZWICK 1995) *Das Problem* SUBGRAPH ISOMORPHISM *besitzt für Graphen H von beschränkter Baumweite $t(H) \leq t$ einen (deterministischen) $2^{\mathcal{O}(k)} n^{t+1}$-Algorithmus.* □

Übung und Anmerkung

Als einen sehr speziellen Fall des Satzes 9.3.1 findet man:

Korollar 9.3.2 *Das Problem* LOG PATH*: „gegeben ein Graph G, besitzt G einen Pfad der Länge $\log n$?“ liegt in* P.

Vergleiche dies mit Übung 1.4.18c. Den ersten polynomiellen Algorithmus für SUBGRAPH ISOMORPHISM bei Graphen mit Baumweite höchstens t gaben PLEHN und VOIGT [PV90]; er hat allerdings Laufzeit $k^{\mathcal{O}(k)}n^{t+1}$. Im Fall, daß G und H Bäume sind, gab schon REYNER [Rey77] einen polynomiellen Algorithmus.

Kapitel 10

Approximationsalgorithmen

10.1 Einleitung

Unter der Annahme P $\neq$ NP gibt es keine polynomiellen Algorithmen, um NP-schwere (Such-) Probleme wie NODE COVER, INDEPENDENT SET oder COLORABILITY exakt zu lösen. Da man in der Praxis solche Probleme trotzdem handhaben muß, kann man sich fragen, wie nahe man denn mit polynomiellen Algorithmen an die optimale Lösung herankommen kann. Wir fragen also nach Approximationsalgorithmen. Wir haben implizit bereits etliche Approximationsalgorithmen kennengelernt.

Kantenfärbung. Aus dem Beweis von Satz 5.6.3 von VIZING ergibt sich unmittelbar ein polynomieller Algorithmus, der die Kanten eines jeden Graphen mit höchstens $\Delta(G)+1$ vielen Farben färbt, also mit höchstens einer Farbe mehr als nötig.

Färben planarer Graphen. Der Beweis zum 4-Farben-Satz lieferte, wie erwähnt, einen zwar polynomiellen, aber nicht praktikablen Algorithmus, um jeden planaren Graphen mit vier Farben zu färben. Wenn man zunächst den planaren Graphen daraufhin testet, ob er bipartit ist, und gegebenenfalls 2-färbt, ergibt sich aber aus dem HEAWOODschen 5-Farben-Satz immerhin ein $\mathcal{O}(n)$-Algorithmus (vgl. Übung 6.3.6b), der jeden planaren Graphen mit höchstens zwei Farben mehr als nötig färbt.

Man sagt, die Probleme EDGE COLORING und PLANAR GRAPH COLORING haben absolute Approximationsalgorithmen oder Approximationsalgorithmen mit Differenzgarantie (vgl. auch Übung 3.2.19c). Bei diesen beiden Problemen besteht die Schwierigkeit darin, den optimalen Wert aus einer kleinen Menge von Werten herauszufiltern. Man beachte, daß in beiden Fällen jede genauere Bestimmung NP-schwer ist.

Nur die wenigsten Optimierungsprobleme haben Approximationsalgorithmen mit Differenzgarantien. Betrachten wir beispielsweise die Probleme CLIQUE (oder äquivalent INDEPENDENT SET).

Proposition 10.1.1 *Unter der Hypothese P $\neq$ NP gibt keinen Approximationsalgorithmus für* CLIQUE *mit endlicher Differenzgarantie.*

Beweis. Sei G ein Graph. Für $k \in \mathbb{N}$ sei der Graph G^k wie folgt definiert. G^k besteht aus k disjunkten Kopien von G, die untereinander vollständig verbunden sind. Dann gilt offenbar

$$\omega(G) = s \quad \Leftrightarrow \quad \omega(G^k) = k \cdot s.$$

Angenommen nun, es gäbe einen Approximationsalgorithmus $\mathcal{A}$ für CLIQUE mit Differenzgarantie k für ein $k \in \mathbb{N}$. Angewandt auf den Graphen G^{k+1} gilt also für die von $\mathcal{A}$ gelieferte Clique $\mathcal{A}(G^{k+1})$:

$$\begin{array}{rcccccc} & \omega(G^{k+1}) & - & |\mathcal{A}(G^{k+1})| & \leq & k & \\ \Leftrightarrow & \omega(G) & - & \frac{|\mathcal{A}(G^{k+1})|}{k+1} & \leq & \frac{k}{k+1} & < 1. \end{array}$$

In mindestens einer der $k+1$ Kopien von G in G^{k+1} ist aber die Restriktion C der Clique $\mathcal{A}(G^{k+1})$ auf diese Kopie von der Größe

$$|C| \geq \frac{|\mathcal{A}(G^{k+1})|}{k+1},$$

und es folgt $|C| - \omega(G) < 1$ oder $|C| = \omega(G)$. Da diese Clique C in polynomieller Zeit gefunden werden kann, ergäbe sich insgesamt ein polynomieller Algorithmus, um $\omega(G)$ zu bestimmen. Also läge das NP-vollständige Entscheidungsproblem CLIQUE in P, und es folgte P = NP. □

Wir wollen daher unsere Erwartungen zurückschrauben, gegen unendlich strebende absolute Fehler zulassen und nach relativen Approximationsalgorithmen suchen, die eine optimale Lösung immerhin noch bis auf einen bestimmten Faktor approximieren (sogenannte relative Approximationsalgorithmen).

Wir wiederholen: ein Optimierungsproblem Π ist eine binäre Relation $\rho \subseteq D_\Pi \times \Sigma^*$ zusammen mit einer Zielfunktion $c : \rho \to \mathbb{Q}$. Wir interpretieren $I \in D_\Pi$ als Eingabe oder Instanz des Optimierungsproblems Π und die Menge

$$Sol(I) := \{\sigma \in \Sigma^* : \langle I, \sigma \rangle \in \rho\}$$

als die Menge der Lösungen zur Instanz I von Π. Die Zahl $c(I,\sigma) := c(\langle I, \sigma \rangle)$ heißt dann der *Wert* der Lösung $\sigma \in Sol(I)$.

$$OPT(I) := \begin{cases} \min\{c(I,\sigma) : \sigma \in Sol(I)\}, & \text{falls } \Pi \text{ ein Minimierungsproblem,} \\ \max\{c(I,\sigma) : \sigma \in Sol(I)\}, & \text{falls } \Pi \text{ ein Maximierungsproblem,} \end{cases}$$

bezeichnet den *Optimalwert* und ein $\sigma^* \in Sol(I)$ mit $c(I,\sigma^*) = OPT(I)$ heißt eine *Optimallösung* der Instanz $I \in D_\Pi$.

Definition 10.1.2 *Sei Π ein Optimierungsproblem mit Definitionsbereich D_Π Zielfunktion c und Optimalwert OPT. Ein (relativer)* Approximationsalgorithmus *ist ein polynomieller Algorithmus $\mathcal{A} : D_\Pi \to \Sigma^*$, der für jede Instanz I von Π eine Lösung $\mathcal{A}(I) \in Sol(I)$ ausgibt. Die Funktion*

$$R_\mathcal{A}(n) := \begin{cases} \max\left\{\frac{c(\mathcal{A}(I))}{OPT(I)} : |I| \leq n\right\}, & \textit{falls } \Pi \textit{ ein Minimierungsproblem,} \\ \max\left\{\frac{OPT(I)}{c(\mathcal{A}(I))} : |I| \leq n\right\}, & \textit{falls } \Pi \textit{ ein Maximierungsproblem,} \end{cases} \tag{10.1}$$

heißt Approximations- *oder* Güteratio *von $\mathcal{A}$. $\mathcal{A}$ selbst heißt dann ein $R_\mathcal{A}(n)$-*Approximationsalgorithmus.

Manchmal gestattet man Approximationsalgorithmen auch eine schlechtere Approximationsgüte auf kleinen Instanzen und interessiert sich lediglich für die sogenannte *asymptotische Approximationsratio* $R_{\mathcal{A}}^{\infty}(n)$. Für ein Suchproblem Π mit $OPT(I) \to \infty$ für $|I| \to \infty$ liege ein Algorithmus $\mathcal{A} : D_\Pi \to \Sigma^*$ vor. Dann ist im Falle, daß Π ein Minimierungsproblem ist, $R_{\mathcal{A}}^{\infty}(n)$ definiert als das

$$\min \{r : \exists C \in \mathbb{R}^+ \, \forall I \in D_\Pi \ (z(\mathcal{A}(I)) \ \leq \ r \cdot OPT(I) + C)\},$$

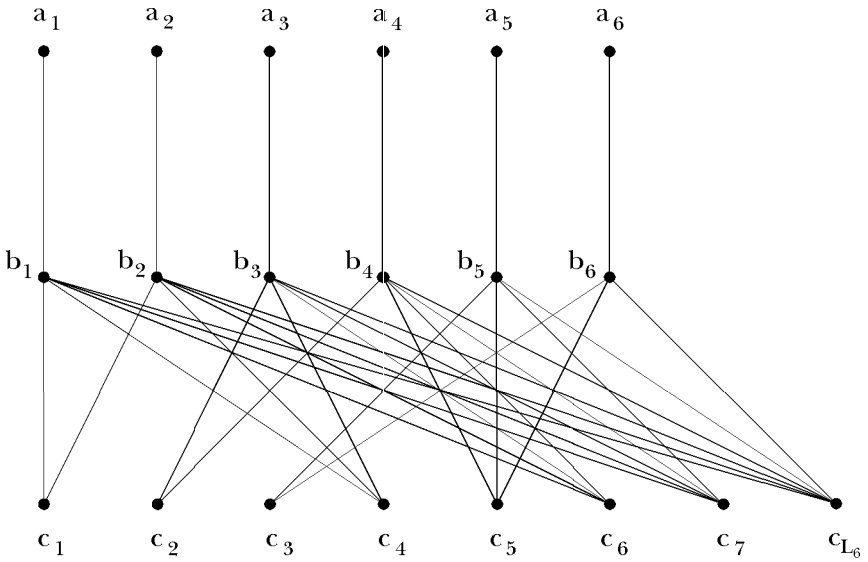
bzw. im Falle eines Maximierungsproblems analog als das

$$\min \{r : \exists C \in \mathbb{R}^+ \, \forall I \in D_\Pi \ (OPT(I) \ \leq \ r \cdot z(\mathcal{A}(I)) + C)\}.$$

Wir wollen im folgenden zunächst ein paar einfache Beispiele für Approximationsalgorithmen betrachten.

10.2 Knotenüberdeckung

Die wohl naheliegendste Heuristik für NODE COVER ist der Greedy-Ansatz, stets einen Knoten maximalen Grades aus G zu entfernen und in die Knotenüberdeckung C_{Greedy} aufzunehmen. Überraschenderweise hat dieser Algorithmus aber selbst auf bipartiten Graphen, wo NODE COVER $\in$ P (vgl. Korollar 4.2.17), keine endliche Güteratio! Betrachte dazu die folgende Graphenfamilie $\{U_k\}_{k \in \mathbb{N}}$ [PS82]. Der Graph U_k hat Knotenmenge $A_k \cup B_k \cup C_k$, wobei $U_k[A_k \cup B_k]$ ein Matching mit k Kanten ist. C_k hat $L_k := \sum_{i=2}^{k} \lfloor k/i \rfloor$ viele Knoten, die in Mengen C_k^i, $i = 2, \ldots, k$, der Größe $\lfloor k/i \rfloor$ eingeteilt sind. Die Knoten $c_1, \ldots, c_{\lfloor k/2 \rfloor}$ sind mit je zwei (anderen) Knoten aus B_k verbunden, die nächsten $\lfloor k/3 \rfloor$ mit je dreien u.s.w. und der letzte Knoten c_{L_k} schließlich mit allen k Knoten aus B_k. Jeder Knoten aus C_k^i ist also zu genau i Knoten in B_k adjazent, so daß kein Knoten in B_k zu mehr als einem Knoten in C_k^i adjazent ist für $2 \leq i \leq k$. Die Knoten in B_k haben also maximalen Grad $\Delta(U_k) = k$, ebenso wie der Knoten c_{L_k} in C_k^k.



Während die Menge B_k offensichtlich eine minimum Knotenüberdeckung C_{opt} von U_k darstellt, wählt der Greedy-Algorithmus im schlechtesten Fall der Reihe nach die Knoten

$c_{L_k}, \ldots, c_1$ und dann noch weitere k Knoten für die Menge C_{Greedy} aus. Für seine Güteratio gilt also

$$\begin{aligned} R_{Greedy}(n) &\geq \frac{|C_{Greedy}(U_k)|}{|C_{opt}(U_k)|} \\ &= \frac{1}{k} \sum_{i=1}^{k} \lfloor k/i \rfloor \\ &\geq \frac{1}{k} \left(\sum_{i=1}^{k} \frac{k}{i} - (k-2) \right) \\ &\geq \sum_{i=1}^{k} \frac{1}{i} - 1 \\ &\geq \int_1^{k+1} \frac{dx}{x} - 1 \\ &\geq \ln k - 1 \qquad = \ln \Delta(U_k) - 1 \end{aligned}$$

Wesentlich schlechter als $\ln \Delta(G)$ (wie im Fall der Graphenfamilie $\{U_k\}_{k \in \mathbb{N}}$) kann das Verhältnis $\frac{|C_{Greedy}(U_k)|}{|C_{opt}(U_k)|}$ für den Greedy-Algorithmus für NODE COVER allerdings nicht werden, wie in Satz 12.4.2 bewiesen werden wird.

Wie irreführend die Intuition beim Entwurf guter Heuristiken sein kann, zeigt die folgende, scheinbar ungünstigere Strategie: Sei M ein maximales Matching, wie es beispielsweise der Greedy-Algorithmus MAXIMAL_MATCHING aus Kapitel 4 liefert, und $C_{MaxMatch}$ die Knotenmenge gebildet aus den von M überdeckten Knoten. Dann ist $C_{MaxMatch}$ eine Knotenüberdeckung, denn gäbe es eine Kante, die mit keinem Knoten aus $C_{MaxMatch}$ inzidierte, so wäre diese Kante unabhängig von M und daher M nicht maximal gewesen. Da wir den Wert einer Optimallösung nicht kennen, können wir die Güte der von diesem Approximationsalgorithmus konstruierten Lösung nicht direkt mit OPT vergleichen. Wie in vielen Fällen gelingt es aber, beide Größen mit einer dritten zu vergleichen. Hier folgt wegen $|C_{MaxMatch}| = 2|M|$ und $\tau(G) \geq |M|$

$$\frac{|C_{MaxMatch}|}{\tau(G)} \leq \frac{2|M|}{|M|} = 2.$$

Wie Matchings oder die vollständig bipartiten Graphen zeigen, ist die Güteratio dieses Algorithmus sogar gleich 2.

Proposition 10.2.1 (GAVRIL 1974b) *Der lineare Algorithmus, der die von einem maximalen Matching in einem Graphen G überdeckten Knoten ausgibt, ist ein Approximationsalgorithmus für* NODE COVER *mit Güteratio 2.*

Beachte, daß für diesen Algorithmus ein minimum maximales Matching die kleinste Knotenüberdeckung liefern würde (ohne allerdings die Güteratio zu beeinflussen); ein solches Matching zu finden ist jedoch bereits wieder NP-schwer [YG80]. Auf zufälligen Graphen liefern beide Algorithmen Knotenüberdeckungen, die bis zu etwa 5% vom Optimum entfernt sind.

Approximation und polynomielle Transformationen. Die Probleme INDEPENDENT SET und NODE COVER lassen sich, wie in Satz 1.4.12 gesehen, durch polynomielle Transformationen ineinander überführen. Sie sind polynomiell äquivalent in dem Sinn, daß es

einen polynomiellen Algorithmus für eines dieser Probleme gibt genau dann, wenn es auch für das andere einen solchen gibt. Per definitionem gibt sogar für jedes beliebige NP-vollständige Problem einen polynomiellen Algorithmus genau dann, wenn es einen polynomiellen Algorithmus für NODE COVER gibt. Dies ist allerdings *keine* Aussage über die Approximierbarkeitseigenschaften der verschiedenen NP-vollständigen Probleme! Eine polynomielle Reduktion überträgt in der Regel nicht die Güteratio eines Approximationsalgorithmus. Tatsächlich können sich NP-vollständige Probleme *drastisch* hinsichtlich ihrer Approximierbarkeit unterscheiden. INDEPENDENT SET und NODE COVER stellen da ein illustratives Beispiel dar. Denn obwohl beispielsweise eine Knotenmenge in G genau dann eine Knotenüberdeckung ist, wenn ihr Komplement eine unabhängige Menge ist, erhält man deshalb aus einer guten Approximation für eine minimale Knotenüberdeckung noch lange keine gute Approximation für eine maximum stabile Menge. Für einen Graphen G, der ein Matching ist, beispielweise gilt $\tau(G) = n/2 = \alpha(G)$. Der Approximationsalgorithmus MAXMATCH für NODE COVER liefert die Knotenüberdeckung $C_{MaxMatch} = V$ mit $|C_{MaxMatch}| = 2 \times \tau(G)$, eine 2-Approximation. Das leere Komplement $V - C_{MaxMatch} = \emptyset$ aber ist wohl eine denkbar schlechte Approximation für eine maximum stabile Knotenmenge in G. Wir werden später sehen, daß es für INDEPENDENT SET keinen Approximationsalgorithmus mit endlicher Güteratio gibt, falls P $\neq$ NP.

Übungen und Anmerkungen

Übung 10.2.2 (SAVAGE 1982) *Sei $G = (V, E)$ ein zusammenhängender Graph und T ein von einer Tiefensuche in G konstruierter spannender Baum von G. Dann bildet die Menge NB der Nicht-Blätter von T eine Knotenüberdeckung von G mit $|NB| \leq 2\tau(G)$.* [*Hinweis:* konstruiere ein Matching aus einer maximum stabilen Menge $S \subseteq V$ in T.]

Approximationsalgorithmen mit besserer Güteratio als 2 für das Problem NODE COVER zu finden ist ein schwieriges Unterfangen. Bis heute ist für kein $r < 2$ ein r-Approximationsalgorithmus für NODE COVER bekannt. Die bislang besten Ergebnisse sind Algorithmen mit Ratio $2 - \frac{\log\log n}{2\log n}$, siehe [BE85, MS85].
Auch für das Knoten-gewichtete NODE COVER Problem gibt es Approximationsalgorithmen mit Güteratio 2, siehe Korollar 12.3.3. Für bipartite Graphen ist hingegen auch das gewichtete Problem durch Reduktion auf ein Maximum-Fluß-Problem in polynomieller Zeit exakt lösbar, siehe [Law76a].

Übung 10.2.3 *Durch „Nachbesserung" der vom Algorithmus von* GAVRIL *konstruierten Knotenüberdeckung zeige man, daß es einen linearen Approximationsalgorithmus $\mathcal{A}$ für* NODE COVER *gibt, der zu einem Graphen G eine Knotenüberdeckung $\mathcal{A}(G)$ mit $|\mathcal{A}(G)| < 2\tau(G)$ ausgibt und der bei Matchings und vollständig bipartiten Graphen eine Optimallösung findet.*

Pfadzerlegung. Mit Hilfe der Reduktion aus Proposition 7.3.9 erhält man für das Problem WEIGHTED PATH PARTITION in allgemeinen Digraphen immerhin noch einen Approximationsalgorithmus mit Güteratio $w(\mathcal{P}_{OPT})/w(\mathcal{P}_{APPROX}) \leq 3/2$. Ein Subgraph S in D mit Außen- und Innengrad höchstens 1 für alle $v \in V$ kann nun auch gerichtete Kreise enthalten. Aus diesen entfernt man lediglich eine Kante kleinsten Gewichts und erhält eine Pfadzerlegung $\mathcal{P}_S$ von D mit $w(\mathcal{P}_S) \geq 2/3 \cdot w(S)$. Eine minimale Pfadzerlegung $\mathcal{P}^*$ induziert aber ein Matching M in H gleichen Gewichts, so daß $w(\mathcal{P}^*) = w_H(M) \leq w_H(M^*) = w(S)$, wobei M^* ein Matching maximalen Gewichts in H sei.

Mit Hilfe eines Algorithmus für maximales gewichtetes (allgemeines) Matching erhält man einen Approximationsalgorithmus mit Güteratio 3/2 für das Pfadzerlegungsproblem in *ungerichteten* Graphen.

Übung 10.2.4 (MORAN, NEWMAN, WOLFSTAHL 1990) *Der folgende Algorithmus ist ein* 3/2*-Approximationsalgorithmus für das* WEIGHTED PATH PARTITION *Problem in ungerichteten Graphen: Berechne einen* 2*-beschränkten Subgraphen* S *von* G *maximalen Gewichts wie in Satz 4.3.9 und entferne aus jedem Kreis in* S *eine Kante minimalen Gewichts.*

10.3 Steiner-Bäume

Sei G ein durch eine Funktion $c : E \to \mathbb{Q}^+$ kantengewichteter Graph und $S \subseteq V$ eine Menge von ausgezeichneten Knoten in G, die sogenannten Terminale. Ein STEINER-*Baum* für S in G ist dann ein Subgraph von G, der ein Baum ist, die Knoten aus S enthält und dessen Blätter Knoten aus S sind. Das Problem STEINER-TREE fragt nun nach einem STEINER-Baum T für S in G minimalen Gewichts $c(T) := \sum_{e \in T} c(e)$. Das Problem läßt sich aus so formulieren, daß eine Knotenmenge $\Sigma \subseteq V \setminus S$ (die sogenannten STEINER-Punkte) gesucht ist, so daß das Gewicht eines minimal spannenden Baumes von $G[S \cup \Sigma]$ minimal wird. Dieses Problem tritt beispielsweise im VLSI-Chip-Design bei der Verdrahtung von Netzen zwischen verschiedenen Anschlüssen (Pins) auf, siehe [Len90].
Während wir in den Spezialfällen $S = V$ (Problem des minimal spannenden Baumes, vgl. Abschnitt 1.3.3) und $|S| = 2$ (kürzeste-Wege-Problem, vgl. Abschnitt 1.3.2) polynomielle Algorithmen kennengelernt haben, ist das allgemeine Problem NP-schwer:

Satz 10.3.1 (KARP 1972) *Das Problem* STEINER TREE *für bipartite Graphen und konstante Gewichtsfunktion ist* NP*-äquivalent.*

Beweis. Die Entscheidungsproblemversion zu STEINER TREE liegt offensichtlich in NP. Beim ungewichteten Problem reduziert sich überdies das Suchproblem auf das Optimalwertproblem und dieses wiederum auf das Entscheidungsproblem (Übung). Wir reduzieren das NP-vollständige Entscheidungsproblem NODE COVER auf die Entscheidungsproblemversion von STEINER TREE. Sei also $G = (V, E)$ ein Graph (eine Instanz von NODE COVER). Der Graph $G' = (V', E')$ gehe aus G hervor, indem zum einen ein zusätzlicher Knoten z vollständig mit G verbunden wird und zum anderen auf jeder ursprünglichen Kante $e \in E$ ein neuer Knoten v_e eingefügt wird (d.h. jede Kante $e \in E$ wird durch einen Pfad P_3 der Länge 2 ersetzt). G' hat mithin $n + m + 1$ viele Knoten und $n + 2m$ viele Kanten (wobei $n = |V|$ und $m = |E|$). G' sei ungewichtet, d.h. $c(e) := 1$ für alle $e \in E'$. Offenbar ist G' bipartit mit Teilen V und $S := \{v_e : e \in E\} + z$. Bezeichne OPT das minimale Gewicht eines STEINER-Baumes in G' für die Terminalmenge S. Dann gilt:

$$OPT = \tau(G) + m.$$

„$\leq$“: Wir wählen die Knoten einer minimum Knotenüberdeckung C von G als STEINER-Knoten. Nach Definition einer Knotenüberdeckung ist dann jeder Knoten v_e, $e \in E$, in G' zu einem Knoten $v \in C$ benachbart. Die Knoten $v \in C$ sind in G' wiederum zu z benachbart. Also ist $G'[S \cup C]$ zusammenhängend. Ein spannender Baum in $G'[S \cup C]$ hat Gewicht (= Anzahl Kanten) höchstens $|C \cup S| - 1 = \tau(G) + m$. Ein solcher enthält aber einen Steinerbaum für S.
„$\geq$“: Sei T ein STEINER-Baum für S in G' mit Gewicht OPT. Die Menge der STEINER-Knoten in T sei mit $C := V(T) \setminus S$ bezeichnet. Da jeder der Knoten v_e, $e \in E$, in T zu mindestens einem STEINER-Knoten $v \in C$ benachbart ist, bildet C eine Knotenüberdeckung von G. Da T ein Baum ist und $c(\cdot) \equiv 1$, gilt $|V(T)| = OPT + 1$. Also wird $\tau(G) \leq |C| = (OPT + 1) - (m + 1) = OPT - m$. □

Bemerkung. Es ist nicht klar, ob auch das allgemeine STEINERbaum-Problem (mit rationaler Kostenfunktion $c : E \to \mathbb{Q}^+$) oder auch nur seine Optimalwert-Version NP-leicht ist. Nur wenn man eine untere Schranke für die Differenz der Gewichte zweier Lösungen kennt (wie beispielsweise bei einer natürlichen Kostenfunktion $c : E \to \mathbb{N}$) kann man mit Hilfe eines Algorithmus für das Entscheidungsproblem den Optimalwert durch binäre Suche in polynomieller Zeit berechen.

Der Distanzgraph D_G eines zusammenhängenden Graphen G ist der vollständige Graph auf $n = |V(G)|$ Knoten mit Kantengewichten $c(\{u, v\}) := dist_G(u, v)$.

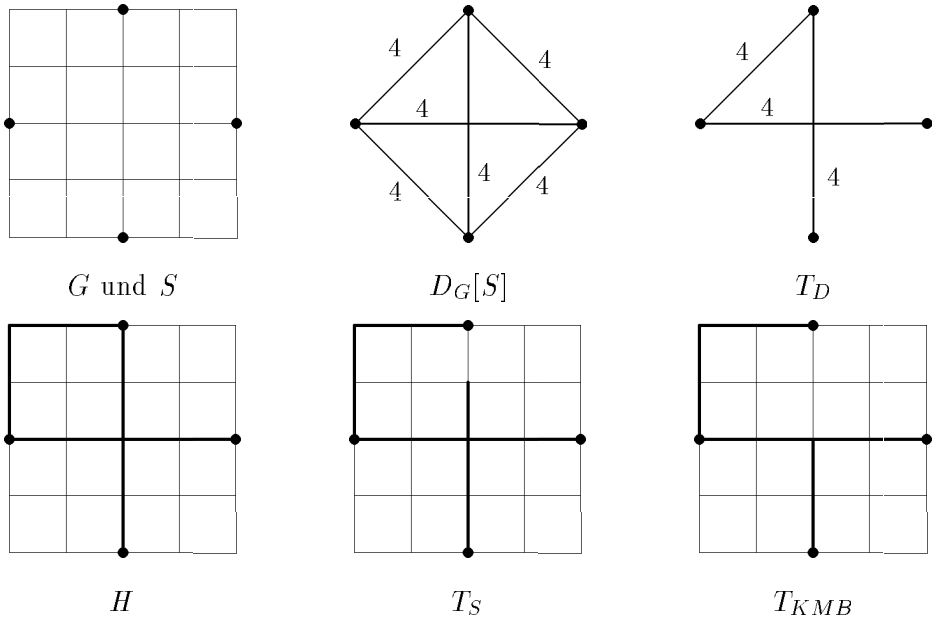
Satz 10.3.2 (KOU, MARKOWSKY, BERMAN 1981) *Sei G ein zusammenhängender Graph und $S \subseteq V$. Der folgende $\mathcal{O}(n^3)$-Algorithmus berechnet einen Steiner-Baum T_{KMB} für S in G mit Approximationsratio höchstens 2:*

- *Berechne den Distanzgraphen D_G.*
- *Berechne einen minimal spannenden Baum T_D für $D_G[S]$.*
- *Transformiere T_D in einen Subgraphen H von G durch Ersetzen der Kanten aus T_D durch Pfade in G:*

$$H := (V, \bigcup_{\{s,t\} \in T_D} P_{st}),$$

  wobei P_{st} die Kantenmenge eines kürzesten $s-t$−Pfads in G bezeichne.
- *Berechne einen minimal spannenden Baum T_S für H.*
- *Entferne sukzessiv Kanten aus T_S, die mit Blättern inzidieren, die nicht zu S gehören, und erhalte einen* STEINER*-Baum T_{KMB}.*

Das folgende Beispiel (mit konstanter Gewichtsfunktion $c(\cdot) \equiv 1$) demonstriert, daß die beiden letzten Schritte wirklich notwendig sind.



Beweis von Satz 10.3.2:
Offensichtlich ist $c(T_{KMB}) \leq c(H) \leq c(T_D)$. Sei nun T^* ein minimaler STEINER-Baum für S in G. Versuchen wir also, $c(T^*)$ nach unten durch eine Funktion in $c(T_D)$ abzuschätzen. Dazu betten wir den Baum T^* in die Ebene ein und umfahren T^* einmal außen herum. Wir erhalten einen (geschlossenen) Zykel Z in G, der alle Knoten von S enthält. Da jede Kante von T^* genau zweimal in Z auftritt, gilt $c(Z) = 2c(T^*)$. Z definiert nun wie folgt einen spannenden Baum für $D_G[S]$: Beginnend in einem Blatt $s_1 \in S$ von T^*, laufe man den Zykel Z einmal ab. Jedem Knoten aus S begegnet man dabei genau einmal zum ersten Mal. Seien $s_1, \ldots, s_{|S|}$ die Knoten von S in der Reihenfolge, in der man ihnen zum ersten Mal begegnet. Dann stellt die Kantenmenge $\{\{s_i, s_{i+1}\} : i = 1, \ldots, |S| - 1\}$ einen spannenden Baum T_Z für $D_G[S]$ dar, es gilt also $c(T_D) \leq c(T_Z)$. Da nun aber jede Kante $\{s_i, s_{i+1}\}$, $i = 1, \ldots, |S| - 1$, des spannenden Baum T_Z nur höchstens so schwer ist wie der zugehörige Abschnitt zwischen s_i und s_{i+1} auf Z, folgt

$$c(T_Z) \leq c(Z) = 2c(T^*), \tag{10.2}$$

und es gilt insgesamt für die Qualität des STEINER-Baumes T_{KMB}:

$$c(T_{KMB}) \leq c(T_D) \leq c(T_Z) \leq 2 \cdot c(T^*). \qquad \square$$

Übung und Anmerkung

Übung 10.3.3 [TM80, KMB81] *Sei T^* ein minimaler* STEINER*-Baum für S in G mit $b \leq |S|$ Blättern und T_D ein minimaler spannender Baum von $D_G[S]$. Man verbessere die Abschätzung (10.2) [und damit die Güte von T_{KMB}] zu $c(T_Z) \leq 2(1 - \frac{1}{b})c(T^*)$ und zeige anhand des Rades $C_{|S|} * 1$, dessen Kreiskanten mit $2 - \epsilon$, die „Speichen" mit 1 gewichtet sind, daß diese Abschätzung bestmöglich ist.*

Während bisher alle Approximationsalgorithmen für das STEINER-Baum-Problem in Graphen auf der Abschätzung (10.2) beruhten, hat ZELIKOVSKI [Zel93a] mit einer neuen Idee einen Approximationsalgorithmus für STEINER TREE mit Güteratio 11/6 gefunden (siehe auch [BR94, Zel93b, KZ95]). Zur Approximation des gewichteten STEINER-Baum-Problems siehe [KR95]. Mehr über das Steinerbaumproblem in [Win87, KPS90, HR92b, HRW92, IT94].

10.4 Das Traveling Salesman Problem

Man stelle sich vor, ein Handelsreisender (rsp. ein Laserbohrer) muß n Orte (rsp. Bohrlöcher) anfahren. Für je zwei Orte ist die Entfernung zwischen ihnen gegeben. Gesucht ist eine kürzeste Tour, die alle Orte abfährt und wieder an den Ausgangspunkt zurückkehrt. Oder: für eine Maschine sind Umrüstzeiten gegeben, d.h. die Zeiten, die nötig sind, um die Maschine von der Produktion eines Produktes v_i auf die Produktion eines Produktes v_j umzustellen. In der Sprache der Graphentheorie formuliert handelt es sich hier beide Male um das Problem, den kürzesten Hamiltonkreis im vollständigen Graphen K_n bezüglich einer Kantenbewertungsfunktion $c : E \to \mathbb{Q}^+$ zu finden. Jede Lösung des Problems TSP besteht mithin aus einer Permutation der n Knoten des K_n. Dieses sogenannte TRAVELING SALESMAN PROBLEM oder kurz TSP:

INSTANZ: ein vollständiger Graph K_n mit Kantengewichten $c : E \to \mathbb{Q}^+$ und ein $B \in \mathbb{Q}^+$,
FRAGE: gibt es eine TSP-Tour C in (K_n, c) der Länge $c(C) := \sum_{e \in C} c(e) \leq B$?

ist eines der meist studierten Probleme der Kombinatorischen Optimierung, siehe [LLRS85, Joh90c, Lap92a, Pot93, Rei94, JRR95]. In vielen praktischen Anwendungen erfüllt die Gewichtsfunktion c (z.B. aus geometrischen Gründen) zusätzlich die Dreiecksungleichung:

$$c(u,w) \leq c(u,v) + c(v,w) \qquad \forall \{u,v,w\} \in \binom{V}{3}. \tag{10.3}$$

Man spricht in diesem Fall vom Problem ΔTSP.

Übung 10.4.1 ΔTSP *für Instanzen mit Gewichtsfunktion* $c : E \to \{1,2\}$ *ist* NP*-äquivalent. [Hinweis:* Reduktion von HAMILTONIAN CIRCUIT]. □

Die naheliegende Greedy-Heuristik für ΔTSP, NEAREST NEIGHBOUR genannt, startet in einer beliebigen Knoten v_1 und fügt im i-ten Schritt, $i = 2, \ldots, n$, einen Knoten $v_i = argmin\{dist(v_{i-1}, u) : u \in V \setminus \{v_1, \ldots, v_{i-1}\}\}$ an die Tour an [Gav65]. Die Güteratio dieses Algorithmus ist jedoch $\Theta(\log n)$ [RSL77]. Das folgende Lemma gibt einen Hinweis, wie man an einen Approximationsalgorithmus für ΔTSP mit endlicher Güteratio gelangen kann.

Lemma 10.4.2 *Sei* $K_n = (V, E, c)$ *eine Instanz von* ΔTSP. *Wenn ein Eulerscher, zusammenhängender (Multi-) Graph* (V, Z) *auf der Knotenmenge* V *bekannt ist, so läßt sich eine* TSP*-Tour* C *in* K_n *mit Gewicht* $c(C) \leq c(Z)$ *in Zeit* $\mathcal{O}(m+n)$ *konstruieren.*

Beweis. Mittels des HIERHOLZER-Algorithmus konstruiert man in linearer Zeit einen Eulerschen Kantenzug $Z = (e_1, \ldots, e_{|Z|})$ in (V, Z). Setze $C := (v)$ für einen beliebigen Knoten $v \in V$. Man laufe nun Z ab und füge jeden Knoten, dem man zum ersten Mal begegnet, hinten an C an. Da C dann alle Knoten von V genau einmal enthält, definiert diese Knotenpermutation sicher eine TSP-Tour in (K_n, c). Die Konstruktionsvorschrift für C kann man andererseits aber auch so verstehen, daß man Z abläuft und, wann immer man auf eine Folge von bereits besuchten Knoten stößt, den Kantenzug Z verkürzt, indem man solche Knoten ausläßt und direkt mit dem nächsten unbesuchten Knoten fortfährt. Aus der Dreiecksungleichung folgt, daß die verkürzte Kantenfolge höchstens so lang wie die ursprüngliche ist und somit $c(C) \leq c(Z)$. □

Satz 10.4.3 (ROSENKRANTZ, STEARNS, LEWIS 1977) *Der folgende* $\mathcal{O}(n^2)$*-Algorithmus* $\mathcal{A}$ *ist ein Approximationsalgorithmus für* ΔTSP *mit Güteratio* < 2. *Sei* (K_n, c) *eine Instanz von* ΔTSP.

- *Berechne einen minimal spannenden Baum* T *in* (K_n, c).
- *Erhalte durch Verdopplung aller Kanten in* T *einen Eulerschen Graphen* T'.
- *Konstruiere mit Hilfe von* T' *eine* TSP*-Tour* C *in* (K_n, c) *gemäß Lemma 10.4.2.*

Beweis. Der Graph T' ist nach Definition Eulersch und zusammenhängend auf V. Also gilt für die nach Lemma 10.4.2 konstruierte TSP-Tour C: $c(C) \leq c(Z) = 2c(T)$. Sei andererseits C^* eine kürzeste TSP-Tour in (K_n, c). Dann ist $C^* - e$ ein spannender Baum von (K_n, c) für alle $e \in C^*$, es folgt $c(T) \leq c(C^* - e) < c(C^*)$. Schaltet man beide Abschätzungen hintereinander, so findet man

$$R_{\mathcal{A}}(K_n, c) = \frac{c(C)}{c(C^*)} < \frac{2c(C^*)}{c(C^*)} = 2. \qquad \square$$

Die sogenannte CHRISTOFIDES-Heuristik verfeinert diesen Ansatz, indem sie, um einen Eulerschen, zusammenhängenden Graphen auf V zu erhalten, statt die Kanten in T zu verdoppeln, ein perfektes Matching minimalen Gewichts zwischen den (gerade vielen) Knoten ungeraden Grades in T zu T hinzufügt.

Übung 10.4.4 (CHRISTOFIDES 1976) *Die* CHRISTOFIDES*-Heuristik hat Approximationsgüte* $< 3/2$ [*Hinweis:* betrachte zwei durch eine optimale Tour C^* definierte perfekte Matchings zwischen den ungeraden Knoten in T]. □

In [PS82] findet man Beispiele, die zeigen, daß für beide Approximationsalgorithmen die jeweilige Güteabschätzung bestmöglich ist.

Nicht-Approximierbarkeit von TSP. Wieder können wir die Theorie der NP-Vollständigkeit auch dazu einsetzen, um die Existenz von Approximationsalgorithmen auszuschließen. Als Beispiel betrachten wir das allgemeine Traveling Salesperson Problem.

Satz 10.4.5 (SAHNI, GONZALES 1976) *Das Problem* TRAVELING SALESMAN *besitzt für kein reelles* $r \geq 1$ *einen* r*-Approximationsalgorithmus (außer* P = NP*).*

Beweis. Wir zeigen, daß sich aus einem r-Approximationsalgorithmus $\mathcal{A}_r$, $r \geq 1$, für TSP ein polynomieller Algorithmus ableiten läßt, der HAMILTONIAN CIRCUIT entscheidet. Da das Problem HAMILTONIAN CIRCUIT NP-vollständig ist, folgt daraus P = NP.
Sei G eine Instanz von HAMILTONIAN CIRCUIT. Wir konstruieren daraus eine TSP-Instanz (K_n, c), indem wir die Kanten $e \in \binom{V}{2}$ des K_n wie folgt gewichten:

$$c(e) := \begin{cases} 1 & \text{falls } e \in E(G), \\ r \cdot n & \text{falls } e \notin E(G). \end{cases}$$

Wir wenden nun den Algorithmus $\mathcal{A}_r$ auf diese Instanz an. Wenn G Hamiltonsch ist, so gilt $OPT = n$, und es wird $c(\mathcal{A}_r(I)) \leq r \cdot n$. Ist G andernfalls nicht Hamiltonsch, so enthält jede TSP-Tour mindestens eine Kante der Länge $c(e) = r \cdot n$, und es wird $c(A_r(I)) \geq n - 1 + r \cdot n > r \cdot n$. Es ist also G Hamiltonsch genau dann, wenn $c(A_r(I)) \leq r \cdot n$, und man kann auf diese Weise das Hamiltonkreis-Problem entscheiden. Der Algorithmus ist offensichtlich polyomiell. □

10.5 Das Zentrumsproblem

Das Zentrumsproblem ist eines der wenigen Optimierungsprobleme, für das man einen optimalen Approximationsalgorithmus kennt, d.h. wo man weißt, daß jede genauere Approximation NP-schwer ist. Es ist wie folgt definiert: Gegeben sei ein zusammenhängender, kantengewichteter Graph $G = (V, E, c)$ mit $c : E \to \mathbb{Q}_+$ sowie ein $k \in \mathbb{N}$. Der Abstand $dist(u, v)$ zweier Knoten u und v sei wie gewohnt die minimale Länge eines $u-v$-Pfades in G, wobei die Länge eines Pfades als die Summe der Kosten seiner Kanten definiert ist. Ein *k-Zentrum* für G ist eine Knotenmenge $Z \in \binom{V}{k}$. Der Wert $dist(v, Z) := \min_{z \in Z} dist(v, z)$ heißt der Abstand eines Knotens $v \in V$ vom Zentrum Z. Der Wert

$$Rad(Z) := \max_{v \in V} dist(v, Z)$$

heißt der *Radius* des Zentrums Z. Gesucht ist nun ein k-Zentrum für $\langle G, c \rangle$ mit minimalem Radius.

Anwendung. Betrachte eine Kaufhauskette, die in einer Menge V von Städten Filialen betreibt und nun in k Städten Warenlager einrichten will, um die Filialen zu beliefern, oder das Problem, in k von n Orten Feuerwehrwachen einzurichten, so daß jeder Ort möglichst schnell von einer der Feuerwachen aus erreichbar ist.

Da das Zentrumsproblem für $k = n$ trivial ist, sei im folgenden o.B.d.A. stets $k < n$. Offenbar besitzt das Zentrumsproblem für jedes feste k und für jedes feste $n - k$ einen polynomiellen Algorithmus.

Satz 10.5.1 (Hsu, Nemhauser 1979) *Das Zentrumsproblem ist* NP*-vollständig. Die Approximation des Zentrumsproblems mit Güteratio r ist* NP*-schwer für alle $r < 2$.*

Beweis. Wir reduzieren das NP-vollständige Problem Dominating Set (vergleiche Übung 1.4.22) auf die Entscheidungsproblemversion des Zentrumsproblems (die offensichtlich in NP liegt). Sei $\langle G, k\rangle$ eine Instanz des Dominating Set Problems. Gewichte alle Kanten $e \in E(G)$ mit $c(e) \equiv 1$. Dann besitzt offenbar G eine dominierende Knotenmenge der Größe höchstens k genau dann, wenn G ein k-Zentrum mit Radius höchstens 1 besitzt. Also ist das Zentren-Problem NP-schwer.
Angenommen nun, wir hätten einen Approximationsalgorithmus $\mathcal{A}$ für das Zentren-Problem mit Güteratio $r < 2$ zur Hand, d.h. zu einer Eingabe $\langle G, c, k\rangle$ konstruiert $\mathcal{A}$ ein k-Zentrum Z für $\langle G, c\rangle$ mit $Rad(Z) < 2R^*$, wobei R^* den Radius eines optimalen k-Zentrums für $\langle G, c\rangle$ bezeichnet. Für die Eingabe $\langle G, 1, k\rangle$ konstruiert $\mathcal{A}$ also ein k-Zentrum Z mit Radius $Rad(Z) \leq 1$ genau dann, wenn G eine dominierende Menge der Kardinalität k besitzt. Also würde $\mathcal{A}$ das NP-vollständige Dominating Set Problem lösen, und es folgte P = NP. □

Wir betrachten nun einen Approximationsalgorithmus für das Zentrumsproblem mit Güteratio 2. Er funktioniert selbst für das knotengewichtete Zentrumsproblem. Beim knotengewichteten Zentrumsproblem ist für den kantengewichteten Graphen $\langle G, c\rangle$ zudem eine Knotengewichtung $w : V \to \mathbb{Q}_+$ gegeben. Der Abstand eines Knotens $v \in V$ zu einem Zentrum Z ist nach wie vor gegeben durch $dist(v, Z) := \min_{z\in Z} dist(v, z)$, doch gehen diese Abstände nun gewichtet in den Radius eines Zentrums ein:

$$Rad(Z) := \max_{v\in V} w(v) \cdot dist(v, Z).$$

Gesucht ist wieder ein k-Zentrum mit minimalem Radius. Der folgende Algorithmus berechnet zu einer Eingabe (G, c, w, R) ein Zentrum Z für $\langle G, c, w\rangle$ mit Radius $Rad(Z) \leq 2R$. (Z darf allerdings beliebig viele Knoten enthalten.)

```
FUNCTION Greedy_Zentrum (G, c, w, R) : 2^V;

BEGIN
  Z := ∅;  U := V;
  WHILE U ≠ ∅ DO BEGIN
    z := argmax {w(u): u ∈ U};
    Z := Z + z;
    U := U \ {u ∈ U : w(u) · dist(u, z) ≤ 2R};
  END;
  Greedy_Zentrum:= Z;
END;
```

Für $R = 0$ enthält das konstruierte Zentrum offenbar n Knoten, für $R = \max_{v \in V} w(v) \cdot \sum_{e \in E} c(e)$, eine obere Schranke für den Radius eines jeden Zentrums, nur einen einzigen Knoten. Wie man leicht sieht, ist die Funktion

$$f(R) := |\textsc{Greedy\_Zentrum}\ (G, c, w, R)|$$

allerdings nicht notwendig schwach monoton fallend in R. Es gilt jedoch:

Lemma 10.5.2 *Sei G ein Graph, $Z^* \subseteq V$ ein k-Zentrum mit minimalem Radius für G und $R^* = Rad(Z^*)$. Dann gilt $f(R) \leq k$ für alle $R \geq R^*$.*

Beweis. Sei $Z^* = \{z_1^*, \ldots, z_k^*\}$. Für $i = 1, \ldots, k$ definiere das Einzugsgebiet $V_i := \{v \in V : w(v) \cdot dist(v, z_i^*) \leq R^*\}$ des Zentrums z_i^*. Nach Definition des Radius eines Zentrums überdecken diese Einzugsgebiete die gesamte Knotenmenge, d.h. es gilt $V \subseteq \bigcup_{i=1}^{k} V_i$.

Betrachte die Wahl des Knotens $z \in U$ in der `WHILE`-Schleife. Sei V_i ein Einzugsgebiet, das z enthält (es existiert mindestens eines). Für alle Knoten $v \in V_i \cap U$ gilt nach Definition eines Einzugsgebiets also $w(v) \cdot dist(v, z_i^*) \leq R^*$ und somit nach Wahl von z

$$\begin{aligned} w(v)dist(v,z) &\leq w(v)\ \ [dist(v, z_i^*) + dist(z_i^*, z)] \\ &\leq w(v) \cdot dist(v, z_i^*)\ +\ w(z) \cdot dist(z, z_i^*) \\ &\leq 2R^*\ \ \leq\ \ 2R. \end{aligned}$$

Also enthält U am Ende eines Schleifendurchlaufs aus keinem Einzugsgebiet V_i, das das in diesem Durchlauf gewählte Zentrum z enthält, noch Knoten, und $\textsc{Greedy\_Zentrum}$ (G, c, w, R) terminiert nach höchstens $k = |Z^*|$ Schleifendurchläufen mit $U = \emptyset$ und einem Zentrum Z aus $|Z| \leq k$ Knoten (und Radius höchstens $2R$). □

Die entscheidende Beobachtung ist nun, daß für den Radius einer optimalen Lösung des knotengewichteten Zentrum-Problems nur

$$anz := |\{w(u) \cdot dist(u, v) : \{u, v\} \in \binom{V}{2}\}| \ \leq\ n(n-1)$$

viele Werte in Frage kommen; es seien dies die Werte $R_1 < R_2 < \cdots < R_{anz}$.

Satz 10.5.3 (Plesník 1987) *Der folgende Algorithmus ist ein 2-Approximationsalgorithmus für das knotengewichtete Zentrumsproblem mit Laufzeit $\mathcal{O}(n^4)$:*

```
FUNCTION Approx_Zentrum (G, c, w, k)  :  2^V;

BEGIN
  berechne die Distanzmatrix (dist(u,v))_{u,v∈V} für ⟨G, c⟩;
  sortiere die Werte {w(u)·dist(u,v) : {u,v} ∈ (V 2)} zu R_1 < ··· < R_anz;
  i := 0;
  REPEAT
    i := i + 1;
    Z := Greedy_Zentrum (G, c, w, R_i);
  UNTIL |Z| ≤ k;
  Approx_Zentrum := Z;
END;
```

Beweis. Aus obigem Lemma folgt zunächst, daß die `REPEAT`-Schleife terminiert. Sei i_0 gleich der Anzahl der Durchläufe durch die `REPEAT`-Schleife, d.h. derjenige Durchlauf, bei dem ein Zentrum Z mit $|Z| \leq k$ gefunden wird, das sodann ausgegeben wird. Aus obigem Lemma folgt $R^* \geq R_{i_0}$. Das von GREEDY_ZENTRUM konstruierte Zentrum Z erfüllt nach Konstruktion aber $Rad(Z) \leq 2R_{i_0}$, ist also eine 2-Approximation an Z^*.

Zur Laufzeit: Wenn man die Distanzmatrix mit Hilfe des Algorithmus von DIJKSTRA, angewandt auf jeden Knoten $v \in V$, berechnet, hat dieser Schritt $\mathcal{O}(n^3)$ Operationen. Ein Sortieralgorithmus mit Laufzeit $\mathcal{O}(N \log N)$ für N Gegenstände berechnet die Anordnung $R_1 < R_2 < \cdots < R_{anz}$ in Zeit $\mathcal{O}(n^2 \log n)$. Da nun die Distanzen vorausberechnet wurden, kann GREEDY_ZENTRUM offensichtlich mit Laufzeit $\mathcal{O}(n^2)$ implementiert werden. Bei $\mathcal{O}(n^2)$ Durchläufen durch die `REPEAT`-Schleife ergibt sich mithin eine Gesamtlaufzeit von $\mathcal{O}(n^4)$. □

Übung und Anmerkungen

Übung 10.5.4 a) *Man konstruiere eine Instanz des Zentrumsproblems, für die das von* APPROX_ZENTRUM *konstruierte Zentrum tatsächlich den doppelten Radius einer Optimallösung hat.* **b)** *Man zeige, daß der Algorithmus* APPROX_ZENTRUM *mit Hilfe einer Art von binärer Suche mit* $\mathcal{O}(\log n)$ *Aufrufen von* GREEDY_ZENTRUM *auskommt.*

Es gibt unter Verwendung des Kürzeste-Wege-Algorithmus von JOHNSON (siehe [CLR90]) mithin eine $\mathcal{O}(n(m + n \log n))$-Implementierung von APPROX_ZENTRUM.

Einen 2-Approximationsalgorithmus für das Zentrumsproblem ohne Knotengewichte gaben schon HOCHBAUM und SHMOYS [HS85]. Ein 3-Approximationsalgorithmus für den knotengewichteten Fall stammt von DYER und FRIEZE [DF85]. Auch das Zentrumsproblem für n Punkte in der Ebene, die durch k möglichst kleine Quadrate zu überdecken sind, besitzt einen 2-Approximationsalgorithmus, der optimal ist [KLC90].

10.6 Randomisierung II: MAX CUT und MAX SAT

Wir behandeln in diesem Abschnitt randomisierte Approximationsalgorithmen. Die Randomisierung ist jedoch von grundsätzlich anderer Natur als bei dem randomisierten (RP-) Algorithmus aus Abschnitt 9.3. Die hier betrachteten Algorithmen liefern nicht in jedem Lauf mit einer bestimmten Mindestwahrscheinlichkeit ein „gutes“ Ergebnis, sondern sie liefern Ergebnisse, die nur im Erwartungswert (d.h. gemittelt über alle Münzwürfe) „gut“ sind.

11.5.1 Zwei einfache derandomisierte Algorithmen

Wir haben die probabilistische Methode benutzt, um die Existenz von Graphen hoher chromatischer Zahl und hoher Taillenweite nachzuweisen (Abschnitt 5.3). In manchen Fällen läßt sich ein solcher Beweis de-randomisieren, d.h. in einen deterministischen Algorithmus umwandeln, der eine Struktur mit den gewünschten Eigenschaften konstruiert. Wir wollen dies an zwei einfachen Beispielen demonstrieren.

Max Cut. Proposition 1.1.19 besagte, daß jeder Graph G einen bipartiten Subgraphen mit mindestens $m/2$ vielen Kanten enthält. MAX CUT ist das Optimierungsproblem hierzu:

MAX CUT:

INSTANZ: ein Graph G und ein $k \in \mathbb{N}$,
FRAGE: gibt es eine Partition $S \dot\cup V \setminus S$ von V mit $|\langle S, V \setminus S\rangle| \geq k$?

Neben seiner theoretischen Bedeutung hat das Optimierungsproblem MAX CUT Anwendungen beim Schaltkreisentwurf und in der Statistischen Mechanik, siehe z.B. [BGJR88]. Während es in den Spezialfällen $k = |E|$ (vgl. Abschnitt 1.3.1), G bipartit oder G planar (vgl. Übung 10.6.12) (bzw. allgemeiner G nicht auf den K_5 kontrahierbar [Bar83a]) polynomiell lösbar ist, gilt für das allgemeine Problem:

Satz 10.6.1 [GJS76] MAX CUT *ist* NP*-vollständig.*

Man beachte, daß wir das Problem, einen *minimum* Schnitt in einem Graphen zu finden, in polynomieller Zeit lösen konnten, siehe Korollar 2.4.7.

Beweis von Satz 10.6.1 [PT95a]:
Wir reduzieren von INDEPENDENT SET. Zum besseren Verständnis zeigen wir zunächst die NP-Vollständigkeit des kantengewichteten WEIGHTED MAX CUT Problems und zeigen dann, wie man diese Reduktion im ungewichteten Fall simulieren kann. Sei also ein Graph $G = (V, E)$ als Instanz von INDEPENDENT SET gegeben. O.B.d.A. enthalte G keine isolierten Knoten, d.h. es gelte $d(v) - 1 \geq 0$ für alle $v \in V$. Sei $H' = (V', F')$ der Graph, der aus G durch Hinzunahme eines neuen Knotens x ensteht, der vollständig zu G verbunden ist, und definiere Kantengewichte $c : F' \to \mathbb{Q}_0^+$ durch

$$c(u,v) := \begin{cases} 1 & \text{falls } \{u,v\} \in E(G), \\ d_G(u) - 1 & \text{falls } v = x. \end{cases}$$

Ein maximum Schnitt $\langle S + x, V \setminus S\rangle_{H'}$, $S \subseteq V$, in (H', c) ist nun insbesondere lokal maximal, d.h. er kann nicht vergrößert werden, indem man einen Knoten $s \in S$ aus S herausnimmt und der Partitionsklasse $V' \setminus (S + x)$ zuschlägt. Es gilt also:

$$\begin{array}{rrcll} & d_G(s) - d_{G[S]}(s) & \geq & d_{G[S]}(s) + (d_G(s) - 1) & \text{für alle } s \in S \\ \Leftrightarrow & d_{G[S]}(s) & \leq & 1/2 & \text{für alle } s \in S, \end{array}$$

d.h. S ist stabil in G. Da andererseits für jede beliebige stabile Menge S in G

$$c\left(\langle S + x, V \setminus S\rangle_{H'}\right) = \sum_{v \in S} d_G(v) + \sum_{v \in V \setminus S} (d_G(v) - 1) = 2m - n + |S|,$$

ist die Maximierung eines Schnittes in (H', c) äquivalent mit der Maximierung der Kardinalität einer stabilen Menge in G.

Wir simulieren diese Reduktion im ungewichteten Fall wie folgt. Den Graphen einer Instanz $G = (V, E)$ von INDEPENDENT SET erweitern wir wie folgt zu einem Graphen $H = (V \cup X \cup Z, F)$. Jeder Knoten $v \in V$ wird zu einer stabilen Menge X_v von $d_G(v) - 1$ neuen Knoten verbunden, $X = \bigcup_{v \in V} X_v$. Um später zu erzwingen, daß die Knoten von X in einem maximum Schnitt stets in ein und derselben Partitionsklasse liegen, sind alle Knoten aus X zudem mit einer Menge Z von $n = |V(G)|$ weiteren neuen Knoten verbunden. Die Kantenmenge von H ist also formal wie folgt definiert:

$$F := E \cup \{\{x,z\} : x \in X, z \in Z\} \cup \bigcup_{v \in V} \{\{v,x\} : x \in X_v\}.$$

Zwei Knoten u und v eines (beliebigen) Graphen mögen *äquivalent* heißen, falls sie selbst nicht benachbart sind, aber dieselben Nachbarn haben. Eine einfache Beobachtung ist, daß dann ein solcher Graph einen maximum (ungewichteten) Schnitt besitzt, in dem alle Paare äquivalenter Knoten zu derselben Partitionsklasse gehören. Im Graph H sind nun aber für jedes $v \in V$ alle Knoten in X_v äquivalent, und ebenso sind alle Knoten in Z äquivalent, liegen also o.B.d.A. in einem maximum Schnitt von H jeweils ganz in einer Partitionsklasse. Da für jedes $v \in V$ mit $d_G(v)-1 > 0$ darüberhinaus $|\langle X_v, Z\rangle| \geq n > d_G(v)-1 = |\langle X_v, \{v\}\rangle|$ gilt, liegen alle nicht-leeren X_v in einem maximum Schnitt in derjenigen Partitionsklasse, die Z nicht enthält, d.h. insbesondere in ein und derselben. Also entsprechen sich maximum Schnitte in H und G' bijektiv, und es gilt:

$$\textsc{Max Cut}(H) \;=\; 2m - n + |S| + \sum_{v \in V} (d_G(v) - 1) \cdot n \;=\; \alpha(G) + (n+1)(2m-n). \qquad \square$$

Betrachten wir nun zunächst den folgenden probabilistischen Beweis von Proposition 1.1.19. Sei $V = \{v_1, \ldots, v_n\}$ und zunächst $V_1, V_2 := \emptyset$. Wir setzen

$$Prob\,(v_i \in V_1) \;=\; 1/2 \;=\; Prob\,(v_i \in V_2).$$

Unser Wahrscheinlichkeitsraum ist also die Menge der 2^n vielen Partitionen von V in $V_1 \dot\cup V_2$ mit der uniformen Verteilung $Prob(V_1 \dot\cup V_2) \equiv 2^{-n}$. Eine Kante $e = \{u, v\} \in E$ ist dann *kreuzend* (d.h. gehört zum Schnitt $\langle V_1, V_2\rangle$: ein Endpunkt von e liegt in V_1 und der andere in V_2) mit Wahrscheinlichkeit $1/2$, da zwei faire Münzwürfe mit Wahrscheinlichkeit $1/2$ übereinstimmen. Sei X_e die Indikatorzufallsvariable des Ereignisses „e ist kreuzend“ (d.h. es ist $X_e = 1$, falls e kreuzt, und $X_e = 0$ sonst). Dann gilt

$$E(X_e) \;=\; Prob\,(\text{„}e \text{ ist kreuzend“}) \;=\; 1/2,$$

und die Zufallsvariable $X := \sum_{e \in E} X_e$ gibt die Anzahl kreuzender Kanten an. Aufgrund der Linearität des Erwartungswertes wird

$$E(X) \;=\; \sum_{e \in E} E(X_e) \;=\; m \cdot 1/2. \tag{10.4}$$

Also gibt es insbesondere *mindestens eine* Partition (V_1, V_2) von V, die mindestens $m/2$ viele kreuzende Kanten enthält und somit einen bipartiten Subgraphen von G definiert, wie er gesucht war.

Einen solchen bipartiten Subgraphen konstruieren wir nun, indem wir obigen probabilistischen Beweis „derandomisieren“. Sei $V_1, V_2 := \emptyset$. Wir gehen die Knoten $\{v_1, \ldots, v_n\}$ von G nun der Reihe nach durch und fügen im i-ten Schritt den Knoten v_i der Menge V_{j_i}, $j_i \in \{1, 2\}$, hinzu. Wir wissen, wenn wir die Knoten von G zufällig und gleichwahrscheinlich auf V_1 und V_2 verteilen, haben wir $m/2$ viele Kanten im Schnitt $\langle V_1, V_2\rangle$ zu erwarten. Den Knoten v_1 können wir offensichtlich beliebig entweder V_1 oder V_2 zuteilen, es gilt also:

$$E\,(X |\, v_1 \in V_{j_1}) \;=\; E(X) \;=\; m/2.$$

Durch die Festlegung von j_1 haben wir uns also nicht verschlechtert. Wir wählen nun auch die folgenden Indizes j_{i+1}, $i = 1, \ldots, n-1$, sukzessive so, daß

$$\begin{aligned} & E\,(X |\, v_1 \in V_{j_1} \wedge \ldots \wedge v_i \in V_{j_i} \wedge v_{i+1} \in V_{j_{i+1}}) \\ \geq\; & E\,(X |\, v_1 \in V_{j_1} \wedge \ldots \wedge v_i \in V_{j_i})\,. \end{aligned} \tag{10.5}$$

Dann ist nach insgesamt n Schritten eine Partition (V_1, V_2) definiert mit

$$|\langle V_1, V_2\rangle| \;=\; E\,(X|\; v_1 \in V_{j_1} \wedge \ldots \wedge v_n \in V_{j_n}) \;\geq\; E(X) \;=\; m/2.$$

Wieso können wir (10.5) immer erfüllen? Nach dem Satz über die vollständige Wahrscheinlichkeit $(*)$ berechnen wir

$$\begin{aligned}
& E\,(X|\; v_1 \in V_{j_1} \wedge \ldots \wedge v_i \in V_{j_i}) \\
= \;& \sum_{e\in E} E\,(X_e|\; v_1 \in V_{j_1} \wedge \ldots \wedge v_i \in V_{j_i}) \\
= \;& \sum_{e\in E} Prob\,(\text{„}e \text{ kreuzt“}|\; v_1 \in V_{j_1} \wedge \ldots \wedge v_i \in V_{j_i}) \\
\overset{(*)}{=} \;& \sum_{e\in E} Prob\,(v_{i+1} \in V_1) \cdot Prob\,(\text{„}e \text{ kreuzt“}|\; v_1 \in V_{j_1} \wedge \ldots \wedge v_i \in V_{j_i} \wedge v_{i+1} \in V_1) \\
& \quad + Prob\,(v_{i+1} \in V_2) \cdot Prob\,(\text{„}e \text{ kreuzt“}|\; v_1 \in V_{j_1} \wedge \ldots \wedge v_i \in V_{j_i} \wedge v_{i+1} \in V_2) \qquad (10.6) \\
= \;& \frac{1}{2}\left[\sum_{e\in E} E\,(X_e|\; v_1 \in V_{j_1} \wedge \ldots \wedge v_{i+1} \in V_1) + \sum_{e\in E} E\,(X_e|\; v_1 \in V_{j_1} \wedge \ldots \wedge v_{i+1} \in V_2)\right] \\
= \;& \frac{1}{2}\left[E\,(X|\; v_1 \in V_{j_1} \wedge \ldots \wedge v_{i+1} \in V_1) + E\,(X|\; v_1 \in V_{j_1} \wedge \ldots \wedge v_{i+1} \in V_2)\right] \qquad (10.7)
\end{aligned}$$

Daher ist mindestens einer der beiden bedingten Erwartungswerte mindestens so groß wie der Erwartungswert $E(X)$, und entsprechend fügen wir v_{i+1} der Menge V_1 bzw. V_2 hinzu.

Unser Vorgehen läßt sich anhand eines Binärbaums mit Wurzel v_1 veranschaulichen, bei dem auf dem i-ten Niveau jeweils entschieden wird, ob v_i der Menge V_1 oder V_2 zugeteilt wird. Die Blätter des Baumes repräsentieren dann die Elemente des Wahrscheinlichkeitsraumes (in unserem Fall die Partitionen $V_1 \dot\cup V_2$). Eine Lösung haben wir dadurch konstruiert, daß wir den Binärbaum von der Wurzel bis in ein Blatt so abgelaufen sind, daß sich die in den Blättern unterhalb der aktuellen Position zu erwartende Anzahl kreuzender Kanten nicht verkleinert hat. In einem Blatt angelangt, gibt es aber keine „freien Variablen“ mehr und der Erwartungswert ist der Funktionswert des zugehörigen Elements des Wahrscheinlichkeitsraumes, d.h. die Anzahl kreuzender Kanten in der zum Blatt gehörigen Partition. Da sich der Erwartungswert nach Wahl des Weges von der Wurzel bis ins Blatt nie verkleinerte, erfüllt auch der Funktionswert im Blatt die Abschätzung über den Erwartungswert $E(X)$ an der Wurzel.

Essentiell für die Effizienz dieses Konstruktionsalgorithmus ist nun, daß die einzelnen (bedingten) Erwartungswerte effizient berechenbar sind. Tatsächlich ist es nicht einmal nötig, die bedingten Erwartungswerte aus (10.7) explizit zu berechnen. Es genügt, einen von beiden zu bestimmen, der mindestens so groß wie der andere ist. Wie man jedoch in (10.6) abliest, unterscheiden sich die beiden bedingten Wahrscheinlichkeiten dafür, daß e kreuzt, nur, wenn ein Endknoten von e der Knoten v_{i+1} ist und der andere einer der Knoten $v_1, \ldots, v_i$. Also genügt es sogar, in jedem Schritt den Index j_{i+1} so zu wählen, daß der neu zuzuordnende Knoten v_{i+1} höchstens so viele Nachbarn in $V_{j_{i+1}}$ wie in der anderen Menge hat (daß dieser simple lineare Greedy-Algorithmus Güteratio 2 hat, kann man allerdings auch direkt einsehen). Da ein Schnitt in einem Graphen höchstens alle Kanten des Graphen enthalten kann, ist dieser Greedy-Algorithmus automatisch ein 2-approximativer Algorithmus für MAX CUT. Zudem läßt er sich in naheliegender Weise auf das Problem WEIGHTED MAX CUT verallgemeinern, bei dem die Kanten des Graphen

durch eine Funktion $w : E \to \mathbb{Q}^+$ gewichtet sind und ein Schnitt gesucht ist, für den die Summe seiner Kantengewichte maximal ist (Übung).

Satz 10.6.2 (SAHNI, GONZALEZ 1976) *Der obige Greedy-Algorithmus ist ein linearer Approximationsalgorithmus für* WEIGHTED MAX CUT *mit Güteratio höchstens 2.* □

Wie der vollständige Graph K_n, n gerade, mit maximum Schnitt $\frac{n^2}{4} \sim m/2$ zeigt, läßt sich für einen Approximationsalgorithmus für MAX CUT eine Güteratio besser als 2 nur zeigen, wenn es gelingt, die Größe eines maximum Schnitts schärfer abzuschätzen; dies gelingt durch Relaxierung des Problems, siehe unten.

Max Sat. Das Problem SAT, zu entscheiden, ob eine gegebene Boolesche Formel $F = \bigwedge_{j=1}^m C_j$ in konjunktiver Normalform erfüllbar ist, d.h. eine Belegung der Booleschen Variablen $x_1, \ldots, x_n$ besitzt, die alle Klauseln C_j erfüllt, war unser erstes NP-vollständiges Problem. Betrachten wir nun das Optimierungsproblem MAX SAT, eine Belegung der Variablen zu finden, die immerhin die meisten der Klauseln erfüllt. Dieses Problem ist offensichtlich NP-schwer. Es ist von Bedeutung bei Experten- und wissensbasierten Systemen. Wir wollen wieder durch Derandomisierung einen Approximationsalgorithmus für MAX SAT gewinnen.

Zunächst: wieviele wahre Klauseln sind zu erwarten, wenn man die Variablen unabhängig von einander mit Wahrscheinlichkeit 1/2 auf wahr bzw. falsch setzt? Unser Wahrscheinlichkeitsraum ist also die Menge aller 2^n verschiedenen Wahrheitswertebelegungen $w \in \{0,1\}^n$ der Booleschen Variablen $x_1, \ldots, x_n$ mit der uniformen Verteilung $Prob(w) \equiv 2^{-n}$.

Bezeichne ℓ_j die Anzahl Literale in der Klausel C_j, $j = 1, \ldots, m$, und Z_j die Indikatorzufallsvariable für das Ereignis, daß die Klausel C_j von der zufälligen Belegung w erfüllt wird. Es ist also $Prob(Z_j = 0) = \left(\frac{1}{2}\right)^{\ell_j}$. Für die Zufallsvariable $Z := \sum_{j=1}^m Z_j$, die die von w erfüllten Klauseln zählt, gilt daher unter Ausnutzung der Linearität des Erwartungswertes

$$\begin{array}{rcccl} E(Z) & = & \sum_{j=1}^m E(Z_j) & = & \sum_{j=1}^m Prob\,(Z_j = 1) \\ & = & \sum_{j=1}^m 1 - 2^{-\ell_j} & \geq & m(1 - 2^{-\ell_{\min}}), \end{array}$$

wobei $\ell_{\min} := \min_j\{\ell_j\}$. Eine zufällig und unabhängig gewählte Belegung der Booleschen Variablen mit Wahrheitswerten erfüllt also im Erwartungswert mindestens $(1 - 2^{-\ell_{\min}})m$ Klauseln. Also gibt es insbesondere mindestens eine Wertebelegung der Variable, die mindestens $(1 - 2^{-\ell_{\min}})m$ Klauseln erfüllt.

Eine solche Wertebelegung $w : \{x_1, \ldots, x_n\} \to \{0,1\}$, $w_i := w(x_i)$, wollen wir durch Derandomisierung deterministisch konstruieren. Wenn $w_1, \ldots, w_i$ bereits definiert sind, so gilt mit den Bezeichnungen

$$\begin{array}{rcl} e_i & := & E(Z|\ x_1 = w_1 \wedge \ldots \wedge x_i = w_i) \\ e_{i0} & := & E(Z|\ x_1 = w_1 \wedge \ldots \wedge x_i = w_i \wedge x_{i+1} = 0) \\ e_{i1} & := & E(Z|\ x_1 = w_1 \wedge \ldots \wedge x_i = w_i \wedge x_{i+1} = 1) \end{array}$$

($e_0 := E(Z)$) wieder wie oben: $e_i = \frac{e_{i0}+e_{i1}}{2}$. Wir wählen also im $(i+1)$-ten Schritt jeweils einen Wahrheitswert w_{i+1} für x_{i+1}, der die (bedingte) Wahrscheinlichkeit nicht verkleinert: $e_{i+1} = e_{iw_{i+1}} \geq e_i$, und finden nach n Schritten eine Belegung w mit

$$E(Z|\ x_1 = w_1 \wedge \ldots \wedge x_n = w_n) \ \geq \ E(Z) \ \geq \ m(1 - 2^{-\ell_{\min}})$$

vielen erfüllten Klauseln.

Man beachte, daß die bedingten Erwartungswerte e_{i0} und e_{i1} nicht explizit berechnet werden müssen. Wir setzen x_{i+1} auf 1 genau dann, wenn $e_{i1} - e_i \geq 0$. Die Festsetzung von w_{i+1} beeinflußt jedoch nur die Klauseln, die x_{i+1} oder $\overline{x_{i+1}}$ enthalten. Wird $w_{i+1} = 1$ gesetzt, so wird jede Klausel C_j, die x_{i+1} enthält, wahr und vergrößert e_i um einen Summanden $1 - (1 - 2^{-\ell_j}) = 2^{-\ell_j}$. Für jede Klausel C_j, die $\overline{x_{i+1}}$ enthält, verringert sich hingegen die Wahrscheinlichkeit, schließlich erfüllt zu werden, von $1-2^{-\ell_j}$ auf $1-2^{-(\ell_j-1)}$ – also ebenfalls um $2^{-\ell_j}$. Es genügt also, diese beiden Effekte gegeneinander abzuwiegen:

$$e_{i1} - e_i = \sum_{C_j \in \mathcal{C}:\, C_j \ni x_i} 2^{-\ell_j} - \sum_{C_j \in \mathcal{C}:\, C_j \ni \overline{x_i}} 2^{-\ell_j},$$

wobei $\mathcal{C}$ die Menge der bisher noch nicht erfüllten Klauseln und ℓ_j die aktuelle Länge der j-ten Klausel bedeutet (aus der gegebenenfalls bereits Literale mit dem Wahrheitswert 0 entfernt wurden). Anschließend entfernen wir alle diejenigen Klauseln, die durch die Setzung von w_{i+1} erfüllt wurden, aus $\mathcal{C}$ und dekrementieren die Länge ℓ_j all derjenigen Klauseln C_j, die den Literal enthalten, der durch die Setzung von w_{i+1} den Wahrheitswert 0 erhalten hat.

Satz 10.6.3 (JOHNSON 1974) *Der folgende Algorithmus ist ein $\mathcal{O}(nm)$-Approximationsalgorithmus für* MAX SAT *mit Güteratio $\leq \frac{1}{1-2^{-\ell_{\min}}} \leq 2$.*

```
PROCEDURE GREEDY_SAT;

BEGIN
  C := {C_1, ..., C_m};
  FOR i = 0 TO n - 1 DO BEGIN
    IF Σ_{C_j ∈ C: C_j ∋ x_i} 2^{-ℓ_j} - Σ_{C_j ∈ C: C_j ∋ ¬x_i} 2^{-ℓ_j} ≥ 0
    THEN BEGIN
      w_{i+1} := 1;
      C := C \ {C_j ∈ C : C_j ∋ x_{i+1}};
      FOR C_j ∈ C : C_j ∋ ¬x_{i+1} DO ℓ_j := ℓ_j - 1;
    END
    ELSE BEGIN
      w_{i+1} := 0;
      C := C \ {C_j ∈ C : C_j ∋ ¬x_{i+1}};
      FOR C_j ∈ C : C_j ∋ x_{i+1} DO ℓ_j := ℓ_j - 1;
    END;
  END; {for}
END;
```

Beweis. Die Güteratio sahen wir schon weiter oben. Zur Laufzeit: Die Menge $\mathcal{C}$ wird durch einen Inzidenzvektor realisiert. Die Wahrscheinlichkeiten $q_j := Prob\ (Z_j = 0) = 2^{-\ell_j}$ werden eingangs berechnet und dann jeweils ggf. modifiziert. Die Komplexität des Algorithmus ist damit offenbar $\mathcal{O}(nm)$, wenn Anfragen der Form „$C_j \ni x_i$?“ in konstanter Zeit durchführbar sind. Hierzu stellt man eingangs die $n \times m$ Matrix (c_{ij}) über $\{-1, 0, +1\}$ bereit, deren Eintrag c_{ij} angibt, ob die Klausel C_j die Boolesche Variable x_i in positiver oder negierter Form oder gar nicht enthält. □

11.5.2 Relaxierung und randomisiertes Runden

Max Sat. Der randomisierte Algorithmus für MAX SAT aus dem letzten Abschnitt erfüllte im Erwartungswert

$$E(Z) = (1 - 2^{-\ell_{\min}})$$

viele Klauseln einer Booleschen Formel F in CNF, wobei $\ell_{\min} := \min_j\{\ell_j\}$ die minimale Länge (d.h. Anzahl Literale) einer Klausel C_j von F angibt. Er liefert also um so bessere Approximationen, je länger die kürzeste Klausel einer Formel ist. Wir werden nun einen (randomisierten) Approximationsalgorithmus kennenlernen, der um so bessere Approximationen liefert, je kürzer die Klauseln sind. Indem man dann die von den beiden Algorithmen produzierten Lösungen miteinander vergleicht, erhält man einen 4/3-Approximationsalgorithmus für MAX SAT (für alle Instanzen).

Die Idee ist, das MAX SAT-Problem als ein lineares Programm zu formulieren und auf Methoden zur polynomiellen Lösung von linearen Programmen zurückzugreifen, wie sie seit einigen Jahren bekannt sind.[1]

Sei C_j^+ (bzw. C_j^-) die Menge der Indizes von Booleschen Variablen, die in C_j positiv (bzw. negiert) auftreten; o.B.d.A. sei $C_j^+ \cap C_j^- = \emptyset$. Betrachte das folgende lineare Programm (IP) zu einer Instanz von MAX SAT:

$$\begin{array}{lll} & \max \sum_{j=1}^m z_j & \\ s.t. \quad \sum_{i\in C_j^+} w_i \; + \; \sum_{i\in C_j^-}(1-w_i) \; \geq \; z_j & \forall j \in \{1,\ldots,m\} & (10.8) \\ w_i \in \{0,1\} & \forall i \in \{1,\ldots,n\} & (10.9) \\ z_j \in \{0,1\} & \forall j \in \{1,\ldots,m\} & (10.10) \end{array}$$

Die Zielfunktion $\sum_{j=1}^m z_j$ zählt die Anzahl der erfüllten Klauseln C_j, wobei die Bedingung (10.8) dafür sorgt, daß z_j nur einen Beitrag von 1 zur Zielfunktion liefert, wenn die Belegung $w : \{x_1,\ldots,x_n\} \to \{0,1\}$ der Booleschen Variablen die Klausel C_j erfüllt. Aufgrund der Ganzzahligkeitsbedingungen (10.9,10.10) ist dieses lineare Programm ein sogenanntes *ganzzahliges lineares Programm*. Während das Problem INTEGER LINEAR PROGRAMMING NP-schwer ist, gibt es für LINEAR PROGRAMMING polynomielle Algorithmen, vgl. [GLS88, Abschnitt 6.6]. Wir relaxieren daher obige Ganzzahligkeitsbedingungen zu

$$\begin{array}{lll} 0 \; \leq \; w_i \; \leq \; 1 & \forall i \in \{1,\ldots,n\} \\ 0 \; \leq \; z_j \; \leq \; 1 & \forall j \in \{1,\ldots,m\}. \end{array}$$

Sei $\hat{w}, \hat{z}$ eine Optimallösung des so definierten linearen Programms (LP) und w^*, z^* eine Optimallösung des ganzzahligen linearen Programms (IP). Dann gilt offensichtlich

$$OPT \; = \; \sum_{j=1}^m z_j^* \; \leq \; \sum_{j=1}^m \hat{z}_j,$$

d.h. wir haben eine polynomiell berechenbare obere Schranke für den Optimalwert OPT unseres MAX SAT-Problems gefunden (die i.a. kleiner ist als m, die Anzahl aller Klauseln).

[1] Ein lineares Programm ist eine Menge von Variablen $\{x_1,\ldots,x_n\}$ zusammen mit einer Menge von linearen Ungleichungen der Form $\sum_{i=1}^n a_{ij}x_i \geq b_j$, $j = 1,\ldots,m$, und einer Zielfunktion $\sum_{i=1}^n c_i x_i$, die es zu maximieren oder zu minimieren gilt (es sind $a_{ij}, b_j, c_i \in \mathbb{Q}$). Geometrisch formuliert sucht man, falls sie existiert, eine Ecke des durch die Ungleichungen definierten Polyeders im $\mathbb{R}^n$, die am weitesten in Richtung $c = (c_1,\ldots,c_n)$ liegt.

Wir werden aus der Lösung $\hat{w}, \hat{z}$ des relaxierten Problems jedoch noch ein weiteres Mal Nutzen ziehen und mit ihrer Hilfe eine Wertebelegung $\hat{x}$ der Booleschen Variablen x konstruieren. Die Idee ist hier, die rationalen Zahlen $\hat{w}_i$ der LP-Lösung auf 0 oder 1 zu runden, wobei die Werte $\hat{w}_i$ als Wahrscheinlichkeiten dienen. D.h. wir setzen $\hat{x}_i$ mit Wahrscheinlichkeit $\hat{w}_i$ auf 1 (*wahr*) und mit Wahrscheinlichkeit $1 - \hat{w}_i$ auf 0 (*falsch*).

Um die erwartete Anzahl der von $\hat{x}$ erfüllten Klauseln abzuschätzen, benötigen wir das folgende Lemma.

Lemma 10.6.4 *Sei $\beta_k := 1 - (1 - 1/k)^k$. Eine Klausel C_j mit k Literalen wird von der randomisiert gerundeten Wertebelegung $\hat{x}$ mit Wahrscheinlichkeit mindestens $\beta_k \hat{z}_j$ erfüllt.*

Beweis. O.B.d.A. können wir annehmen, daß C_j keine negierten Variablen als Literale enthält und daß C_j die Form $x_1 \vee \ldots \vee x_k$ hat. Nach Bedingung (10.8) des LPs gilt

$$\hat{w}_1 + \cdots + \hat{w}_k \;\geq\; \hat{z}_j. \tag{10.11}$$

Klausel C_j wird von $\hat{x}$ nur dann nicht erfüllt, wenn alle Variablen $x_1, \ldots, x_k$ auf 0 gerundet wurden. Da die Variablen unabhängig voneinander gerundet werden, ist die Wahrscheinlichkeit hierfür $\prod_{i=1}^{k}(1 - \hat{w}_i)$. Es bleibt, zu zeigen, daß

$$1 - \prod_{i=1}^{k}(1 - \hat{w}_i) \;\geq\; \beta_k \hat{z}_j. \tag{10.12}$$

Aufgrund der Beziehung

$$\sqrt[k]{a_1 \cdot \cdots \cdot a_k} \;\leq\; (a_1 + \cdots + a_k)/k$$

zwischen geometrischem und arithmetischem Mittel von k nicht-negativen Zahlen gilt

$$1 - \prod_{i=1}^{k}(1 - \hat{w}_i) \;\geq\; 1 - \left(\frac{\sum_{i=1}^{k}(1 - \hat{w}_i)}{k}\right)^k \;=\; 1 - \left(1 - \frac{\sum_{i=1}^{k} \hat{w}_i)}{k}\right)^k \overset{(10.11)}{\geq} \; 1 - \left(1 - \frac{z_j}{k}\right)^k.$$

Da die Funktion $f(y) = 1 - (1 - y/k)^k$ auf dem Intervall $[0, 1]$ konkav ist, verläuft sie oberhalb ihrer Sekante durch $(0, 0)$ und $(1, \beta_k)$:

$$f(y) \;\geq\; \frac{y}{1 - 0}(f(1) - f(0)) \;=\; \beta_k y \qquad \text{für alle } y \in [0, 1],$$

womit schließlich (10.12) folgt. □

Beachte, daß die Funktion $f(n) = (1 + x/n)^n$ für $x \in \mathbb{R}$ streng monoton steigend ist, $\lim_{n\to\infty}(1 + x/n)^n = e(x)$ und daher die Funktion β_k streng monoton fallend in k ist mit $\lim_{k\to\infty} \beta_k = 1 - 1/e$.

Proposition 10.6.5 *Sei F eine Instanz von* MAX SAT *mit höchstens $k_{\max}$ Literalen pro Klausel. Bezeichne OPT die maximale Anzahl erfüllbarer Klauseln von F. Dann ist die erwartete Anzahl erfüllter Klauseln bei randomisiertem Runden der Lösung $\hat{w}$ der LP-Relaxation auf $\{0, 1\}$ mindestens $\beta_{k_{\max}} \cdot OPT$, d.h. der so definierte Algorithmus ist ein randomisierter r-Approximationsalgorithmus für* MAX SAT *zu $r \leq \beta_{k_{\max}}^{-1} \leq \frac{e}{e-1} \approx 1.58$.*

Beweis. Sei Z_j die Indikatorzufallsvariable für das Ereignis, daß die Klausel C_j von der gerundeten Belegung $\hat{x}$ erfüllt wird. Wegen der Linearität des Erwartungswerts finden wir für die erwartete Anzahl erfüllter Klauseln

$$\begin{array}{rcll} E(\sum_{j=1}^m Z_j) & = & \sum_{j=1}^m E(Z_j) & \\ & = & \sum_{j=1}^m \text{Prob}(\hat{x} \text{ erfüllt } C_j) & \\ & \geq & \sum_{j=1}^m \beta_{k(C_j)} \hat{z}_j & \geq \beta_{k_{\max}} \cdot OPT. \end{array}$$

□

Satz 10.6.6 (Goemans, Williamson 1994a) *Wenn man sowohl den trivialen randomisierten Algorithmus für* Max Sat *(der alle Booleschen Variablen unabhängig mit Wahrscheinlichkeit 1/2 auf wahr bzw. falsch setzt) als auch den obigen mit randomisiertem Runden anwendet und die bessere der beiden Approximationen wählt, so erhält man einen randomisierten* 4/3*-Approximationsalgorithmus für* Max Sat.

Beweis. Sei F eine Instanz von Max Sat und E_1 (bzw. E_2) der Erwartungswert für die Anzahl der erfüllten Klauseln beim ersten, trivialen Algorithmus (bzw. beim zweiten mit randomisiertem Runden). Wegen

$$\max\{E_1, E_2\} \geq (E_1 + E_2)/2 \overset{(*)}{\geq} 3/4 \sum_{j=1}^m \hat{z}_j \geq 3/4 \cdot OPT$$

genügt es, $(*)$ zu zeigen. Bezeichne S_k die Menge der Klauseln von F mit genau k Literalen. Aus

$$E_1 = \sum_k \sum_{C_j \in S_k} (1 - 2^{-k}) \geq \sum_k \sum_{C_j \in S_k} \alpha_k \hat{z}_j$$

mit $\alpha_k := 1 - 2^{-k}$ und

$$E_2 \geq \sum_k \sum_{C_j \in S_k} \beta_k \hat{z}_j$$

folgt

$$\frac{E_1 + E_2}{2} \geq \sum_k \sum_{C_j \in S_k} \frac{\alpha_k + \beta_k}{2} \hat{z}_j.$$

Die ersten Werte von α_k und β_k sind in der folgenden Tabelle aufgelistet.

k	α_k	β_k
1	0.5	1.0
2	0.75	0.75
3	0.875	0.704

Für $k = 1, 2$ gilt offensichtlich $\alpha_k + \beta_k \geq 3/2$. Da α_k als Funktion in k streng monoton steigend ist, während β_k streng monoton fallend ist mit Grenzwert $\lim_{k\to\infty} \beta_k = 1 - 1/e$, folgt für $k \geq 3$: $\alpha_k + \beta_k \geq 0.875 + 1 - 1/e \approx 1.51 > 3/2$. □

Max Cut. Das Max Cut Problem läßt sich wie folgt als ein äquivalentes ganzzahliges quadratisches Programm formulieren:

$$\begin{array}{lll} & \max \sum_{i<j} a_{ij}(1 - x_i x_j)/2 & \\ s.t. & x_i \in \{-1, 1\} & \forall i \in \{1, \ldots, n\}, \end{array} \tag{10.13}$$

wobei $A = (a_{ij})$ die Adjazenzmatrix des betrachteten Graphen G bezeichnet. Die Zahlen $x_1, \ldots, x_n \in \{-1, 1\}$ codieren in naheliegender Weise einen Schnitt, und die Zielfunktion zählt offensichtlich die Anzahl der geschnittenen Kanten. Die Bedingung (10.13) relaxieren wir nun wie folgt. Es bezeichne $\$_n := \{x \in \mathbb{R}^n : ||x||_2 = 1\}$ die Einheitssphäre im $\mathbb{R}^n$, d.h. die Menge der n-dimensionalen Vektoren mit EUKLIDischer Norm 1. Statt nun $x_i \in \$_1$ zu wählen, lassen wir beliebige n-dimensionale Richtungsvektoren $y_i \in \$_n$ zu und ersetzen das Produkt $x_i x_j$ in naheliegender Weise durch das Skalarprodukt der Vektoren y_i und y_j.

$$\begin{array}{ll} \max \sum_{i<j} a_{ij}(1 - y_i \cdot y_j)/2 & \\ s.t. \quad y_i \in \$_n & \forall i \in \{1, \ldots, n\}. \end{array} \tag{10.14}$$

Für jede Kante $\{v_i, v_j\} \in E$ (d.h. es ist $a_{ij} = 1$) versuchen wir in dieser Formulierung also, das Skalarprodukt

$$y_i \cdot y_j \;=\; y_i^T y_j \;=\; \sum_{k=1}^{n} (y_i)_k (y_j)_k \;=\; \cos \alpha(y_i, y_j)$$

zu minimieren, wobei $\alpha(y_i, y_j)$ den Winkel zwischen den beiden Vektoren y_i und y_j bezeichnet (gemessen in Radiant); das heißt aber wiederum, daß zwei zu einer Kante gehörige Vektoren einen möglichst stumpfen Winkel bilden sollen. Der Optimalwert dieses Programms ist offenbar höchstens größer als der Wert eines maximum Schnittes in G. Wie sich herausstellt, ist diese obere Schranke an den Optimalwert $OPT = OPT(G)$ des MAX CUT Problems erheblich besser als die triviale Schranke $m = |E(G)|$.

Es stellen sich nun zwei Probleme. Erstens: wie kann man die Relaxierung (10.14) lösen? Und zweitens: wie erhält man aus einer Lösung des relaxierten Problems eine Lösung des ursprünglichen Problems, d.h. einen Schnitt, zurück und wie gut ist dieser?

Gehen wir zunächst der Frage nach, wie man obiges Programm lösen kann. Dazu einige Begriffe aus der Linearen Algebra. Eine symmetrische Matrix S heißt *positiv semidefinit*, falls $x^T S x \geq 0$ für alle $x \in \mathbb{R}^n$. Dies ist äquivalent damit, daß S eine Darstellung $S = Y^T Y$ besitzt, wobei $rg(Y) = rg(S)$. Sei Y nun die $n \times n$ Matrix mit den Vektoren $y_1, \ldots, y_n$ in den Spalten. Dann ist die sogenannte GRAMsche Matrix $S = (s_{ij}) = Y^T Y$ der Vektoren $y_1, \ldots, y_n$ positiv semidefinit, es gilt $s_{ij} = y_i \cdot y_j$, und S hat lauter Einsen auf der Hauptdiagonale. Wir können obiges Programm damit wie folgt umschreiben:

$$\begin{array}{ll} \max \sum_{i<j} a_{ij}(1 - s_{ij})/2 & \\ s.t. \quad S = (s_{ij}) \text{ positiv semidefinit} & \\ \qquad s_{ii} = 1 & \forall i \in \{1, \ldots, n\}. \end{array} \tag{10.15}$$

Eine Lösung des Programms (10.14) liefert offenbar unmittelbar auch eine Lösung von (10.15). Umgekehrt kann man mit Hilfe der (unvollständigen) CHOLESKY-Zerlegung (siehe z.B. [GL89, Seite 90]) eine Lösung S des Programms (10.15) zerlegen in $S = Y^T Y$ und erhält somit eine Lösung von (10.14) mit demselben Zielfunktionswert. (10.15) heißt ein *semidefinites Programm*. Für jedes vorgegebene $\epsilon > 0$ kann in Zeit polynomiell in der Eingabe und $\log 1/\epsilon$ eine Lösung von (10.15) mit Wert mindestens Optimalwert von (10.15) weniger ϵ berechnet werden [GLS88, NN94, Ali95] (siehe auch [VB95]).

Kommen wir nun zum Rundungsteil. Wir „runden“ die gefundene, kontinuierliche Lösung $y_1, \ldots, y_n$ von (10.14) zu einer diskreten Lösung des MAX CUT Problems, indem

wir gemäß der n-dimensionalen Standardnormalverteilung zufällig einen Vektor $h \in \mathbb{R}^n$ wählen (vgl. hierzu [Knu81, Seite 130]); dieser Vektor ist Normalenvektor einer Hyperebene durch den Nullpunkt, die $\mathbb{R}^n$ in zwei Hälften teilt. Die entsprechende Einteilung der Vektoren $y_1, \ldots, y_n$ in zwei Mengen definiert einen Schnitt $\langle S_h, V \setminus S_h \rangle$ von G gemäß $S_h := \{v_i \in V : y_i \cdot h \geq 0\}$.

Satz 10.6.7 (Goemans, Williamson 1994b) *Obiger Algorithmus ist ein randomisierter Approximationsalgorithmus für* Max Cut *mit*

$$\frac{OPT(G)}{E_h(|\langle S_h, V \setminus S_h \rangle|)} \leq 1.14$$

für alle Graphen G, wobei $OPT(G)$ die Größe eines maximum Schnittes in G bezeichnet.

Beweis. Seien y_i und y_j zwei zu einer Kante $\{v_i, v_j\} \in E$ gehörige Vektoren. y_i und y_j definieren eine Ebene, die wiederum einen Kreis aus $\$_n$ ausschneidet. Die Vektoren h, so daß die durch h definierte Hyperebene y_i und y_j trennt, liegen in zwei Sektoren dieses Kreises, die sich jeweils um den Winkel $\alpha(y_i, y_j)$ öffnen. Da die Projektion der n-dimensionalen Standardnormalverteilung auf die Hyperebene die 2-dimensionale Standardnormalverteilung ergibt, ist die Wahrscheinlichkeit, daß h die Knoten y_i und y_j trennt, $\frac{2\alpha(y_i,y_j)}{2\pi}$. Wegen der Linearität des Erwartungswerts folgt

$$E_h(|\langle S_h, V \setminus S_h \rangle|) = \sum_{i<j} a_{ij} \frac{\alpha(y_i, y_j)}{\pi}. \tag{10.16}$$

Definiere

$$r := \max_{0 \leq x \leq \pi} \frac{\pi}{2} \frac{1 - \cos x}{x},$$

dann gilt also

$$\frac{1 - \cos x}{2} \leq \frac{rx}{\pi}. \tag{10.17}$$

Bezeichne z^*_{SDP} den Wert einer Optimallösung $S^* = (s^*_{ij})$ des semidefiniten Programms (10.15) und $y^*_1, \ldots, y^*_n$ die aus S^* berechenbare Optimallösung des Programms (10.14). Dann gilt

$$\begin{aligned}
OPT(G) &\leq z^*_{SDP} \\
&= \sum_{i<j} a_{ij} \frac{1 - s^*_{ij}}{2} \\
&= \sum_{i<j} a_{ij} \frac{1 - y^*_i \cdot y^*_j}{2} \\
&= \sum_{i<j} a_{ij} \frac{1 - \cos \alpha(y^*_i, y^*_j)}{2} \\
&\overset{(10.17)}{\leq} r \cdot \sum_{i<j} a_{ij} \frac{\alpha(y^*_i, y^*_j)}{\pi} \\
&\overset{(10.16)}{=} r \cdot E_h(|\langle S_h, V \setminus S_h \rangle|)
\end{aligned}$$

Wegen $r < 1.14$ ist nun ein aus einer $\epsilon-$Approximation an S^* berechneter Schnitt $\langle S_h, V \setminus S_h \rangle$ für ϵ klein genug noch eine 1.14-Approximation an MAX CUT. □

Übungen und Anmerkungen

MAX CUT ist selbst für kubische Graphen NP-vollständig [Yan81a]. Die NP-Vollständigkeit des gewichteten MAX CUT Problems zeigte schon KARP [Kar72].

Übung 10.6.8 [PT95a] **a)** MAX CUT *ist* NP*-vollständig auf der Menge der 3-partiten Graphen.*
b) *Sei* $0 \leq r < 1$ *fest. Dann ist das Problem* MAX CUT*, eingeschränkt auf die Menge aller Graphen* G *mit Taillenweite* $g(G) \geq n^r$*,* $n = |V(G)|$*,* NP*-vollständig.* [*Hinweis:* Reduktion von MAX CUT, vergleiche Übung 1.5.10]

Satz 10.6.2 läßt sich wie folgt verbessern.

Übung 10.6.9 a) *Ein bipartiter Graph* $B = (V_1, V_2, E)$ *auf* n *Knoten heißt balanciert, falls* $|V_1| = \lfloor \frac{n+1}{2} \rfloor$ *und* $|V_2| = \lceil \frac{n-1}{2} \rceil$*. Jeder kantengewichtete Graph* $G = (V, E, w)$*,* $w : E \to \mathbb{Q}_+$*, auf* $n = 2k$ *[rsp.* $n = 2k+1$*] Knoten enthält einen balancierten, bipartiten Subgraphen* $B = (V_1, V_2, F)$*,* $F = \langle V_1, V_2 \rangle_G$*, mit* $w(F) \geq \frac{w(E)}{2} \cdot (1 + \frac{1}{n-1})$ *[rsp.* $w(F) \geq \frac{w(E)}{2} \cdot (1 + \frac{1}{n})$*] vielen Kanten* [Erd67]. [*Hinweis*: probabilistisches Argument – wähle einen passenden Wahrscheinlichkeitsraum, o.B.d.A. sei wieder $V_1 \dot\cup V_2 = V$. Der Fall n ungerade läßt sich durch Entfernen eines Knotens minimalen Grades auf den Fall n gerade reduzieren.]
b) *Man gebe einen linearen Algorithmus für dieses Problem an* [HL96b]. [*Hinweis:* konstruiere zunächst ein Matching M in G mit $w(M) \geq w(G)/(n-1)$, vergleiche dazu Übung 4.1.1c]
c) *Man gebe einen linearen Algorithmus an, der in einem Graphen* $G = (V, E, w)$ *einen bipartiten Subgraphen* $B = (V_1, V_2, F)$ *mit* $w(F) \geq \frac{w(E)}{2} \cdot (1 + \frac{1}{\min\{\Delta(G)+1, \lceil \sqrt{m} \rceil\}})$ *konstruiert* [HL96b]. [*Hinweis:* zunächst knotenfärbe G und kontrahiere die Farbklassen]

Während sich in einem zusammenhängenden Graphen noch ein bipartiter Subgraph mit $\frac{m}{2} \cdot (1 + \frac{n-1}{2m})$ [PT82, NT93] bzw. $m/2 + \sqrt{m/8} + \Omega(\sqrt[4]{m})$ [Alo96] vielen Kanten in polynomieller Zeit konstruieren läßt, kann man zeigen, daß ein maximum bipartiter Subgraph in fast allen Graphen die Größe $(1/2 + o(1)) \cdot m$ hat [NT93]. Darüberhinaus gilt:

Übung 10.6.10 [HV91, NT93] *Für jedes* $0 < \epsilon < 1/2$ *ist das Entscheidungsproblem, ob ein Graph* G *einen bipartiten Subgraphen mit mindestens* $(1/2 + \epsilon) \cdot m$ *vielen Kanten enthält,* NP*-vollständig.*

Übung 10.6.11 [JAMS91] MAX CUT *bleibt* NP*-vollständig, wenn man verlangt, daß eine der beiden Partitionsklassen stabil sein soll.*[2]

MAHAJAN und RAMESH [MR95b] gelang es, den Approximationsalgorithmus für MAX CUT von GOEMANS und WILLIAMSON zu derandomisieren. Daß die semidefinite Relaxierung keine substanziell besseren Approximationsalgorithmen liefern kann, zeigten DELORME und POLJAK [DP93a]. Auf dichten Graphen (mit $\Omega(n^2)$ vielen Kanten) gibt es dagegen sogar ein polynomielles Approximationsschema für MAX CUT [AKK95]. Zu oberen Schranken für MAX CUT über Eigenwerte der Adjazenzmatrix siehe [DP93a, DP93b]. Ein sehr schöner Übersichtsartikel zu MAX CUT ist [PT95a].

Übung 10.6.12 [OD72, Had75, AI77, Bar90a] *Das Problem* MAX CUT *ist für planare Graphen in polynomieller Zeit lösbar.* [*Hinweis:* Übung 6.3.11 und Satz 4.3.8]

[2] Dieses Problem trat als Teilproblem bei der Färbungsheuristik RECURSIVE_LARGEST_FIRST auf.

Das Problem MINIMUM k-CUT: zu einem Graphen G und einer natürlichen Zahl $k \geq 3$ eine Kantenmenge F minimalen Gewichts in G zu bestimmen, so daß $G \setminus F$ in (mindestens) k Zusammenhangskomponenten zerfällt, ist ebenfalls NP-vollständig [GH94]. Einen 2-Approximationsalgorithmus findet man in [SV95]. Während für das MINIMUM k-CUT für k fest polynomiell lösbar wird [GH94], ist dieses Problem interessanterweise bei Spezifizierung von k Knoten sogar für festes k NP-schwer: das Problem MULTITERMINAL CUT, zu einem Graphen $G = (V, E)$ und einer k-elementigen Knotenmenge $T \subset V$, $k \geq 3$ fest, eine minimum Kantenmenge zu finden ist, die die Knoten aus T paarweise voneinander trennt, ist NP-vollständig [DJPSY94].

Übung 10.6.13 *Die folgende Heuristik für* MULTITERMINAL CUT *hat Güteratio höchstens* $2\frac{k-1}{k}$ [DJPSY94]*: Man berechnet für alle Knoten* $t_i \in T$, $1 \leq i \leq k$, *einen minimum-Schnitt* $C_i \subseteq E$, *der* t_i *von allen anderen Knoten aus* T *trennt, berechnet den größten Schnitt* $C_{i_{\max}}$ *unter diesen und gibt* $\bigcup_{i \neq i_{\max}} C_i$ *aus. Die angegebene Güteratio ist bestmöglich für diese Heuristik.*

Den Begriff der k-Färbbarkeit relaxierend, kann man sich fragen, wie man einen gegebenen Graphen $G = (V, E)$ mit k Farben (Knoten-) färben kann, so daß die Endknoten möglichst vieler Kanten verschiedene Farben tragen (das sogenannte MAX k-COLORABLE SUBGRAPH Problem).

Übung 10.6.14 (VITÁNYI 1981) *Sei* $k \in \mathbb{N}$, $k \geq 3$ *fest.*
a) MAX k-COLORABLE SUBGRAPH *ist* NP*-vollständig.*
b) *Man gebe einen randomisierten Algorithmus für* MAX k-COLORABLE SUBGRAPH *an, der zu einem* $\mathcal{O}(n+m)$*-Approximationsalgorithmus mit Güteratio* $\frac{k}{k-1}$ *derandomisiert werden kann.*

Den ersten 4/3-Approximationsalgorithmus für MAX SAT gab YANNAKAKIS [Yan94]. In [BGLR93] wird gezeigt, daß die Approximation von MAX 3-SAT mit Ratio $\leq 113/112$ NP-schwer ist. Die Technik des randomisierten Rundens wurde von RAGHAVAN und THOMPSON [RT87, Rag88] eingeführt. Einen 4/3-Approximationsalgorithmus für MAX SAT erhält man auch, wenn man nicht mit Wahrscheinlichkeit $\hat{w}_i$ auf 1 rundet, sondern mit Wahrscheinlichkeit $f(\hat{w}_i)$, wobei f eine reelle Funktion mit

$$1 - 4^{-y} \;\leq\; f(y) \;\leq\; 4^{y-1} \qquad \text{für alle } y \in [0,1]$$

ist (wie z.B. $f(y) = 1/4 + y/2$); man muß also nur noch einen Algorithmus anwenden [GW94a].

Übung 10.6.15 (HOCHBAUM 1982) *Betrachte die folgende LP-Relaxation für eine Instanz* $G = (V, E)$ *von* NODE COVER*:*

$$\begin{array}{rll} & \min \sum_{v \in V} x_v & \\ s.t. & x_u + x_v \;\geq\; 1 & \forall \{u, v\} \in E, \\ & 0 \leq x_v \leq 1 & \forall v \in V. \end{array}$$

Sei $\hat{x}$ *eine Optimallösung dieses LPs und* $LP(G) := \sum_{v \in V} \hat{x}_v$.
a) *Die Menge* $C_{LP} := \{v \in V : \hat{x}_v \geq 0.5\}$ *ist eine Knotenüberdeckung von* G *der Kardinalität höchstens* $2\tau(G)$.
b) *Man gebe eine Familie* $(G_k)_{k \geq 3}$ *von Graphen an, für die* $OPT(G_k)/LP(G_k) \nearrow 2$ *für* $k \to \infty$. *(D.h. der Faktor 2 geht schon bei der Relaxation verloren.)*
c) *Man gebe eine Familie* $(G_k)_{k \geq 1}$ *von Graphen an mit* $OPT(G_k)/LP(G_k) \searrow 1$, *aber* $|C_{LP}| = 2 \cdot LP(G_k)$.
d) *Man verallgemeinere auf das knotengewichtete Knotenüberdeckungsproblem, bei dem eine Knotenüberdeckung* $C \subseteq V$ *minimalen Gewichts* $\sum_{c \in C} w(c)$ *gesucht ist.*

In Korollar 12.3.3 wird ein einfacher, linearer 2-Approximationsalgorithmus für das gewichtete Knotenüberdeckungsproblem vorgestellt, der ohne Lineare Programmierung auskommt.

10.7 Lokale Verbesserung

Wir haben nun einige (primale oder Eröffnungs-) Heuristiken kennengelernt, mittels derer sich Lösungen verschiedener Optimierungsprobleme konstruieren ließen. Meist waren sie recht einfach strukturiert. Der Greedy-Algorithmus für MAX CUT beispielsweise erfordert gerade mal einen einzigen Durchgang durch die Knotenmenge des Graphen. Da wir auf der Suche nach einer möglichst guten Lösung des Optimierungsproblems sind, ist es naheliegend, zu versuchen, die erhaltene Lösung noch weiter zu verbessern. Die Idee von lokalen Verbesserungsheuristiken ist, die Lösung sukzessive durch kleine lokale Veränderungen weiter zu verbessern. Eine Verbesserungsheuristik zu einem kombinatorischen Optimierungsproblem Π besteht demnach aus

- einem Nachbarschaftsgraph $\mathcal{N}(I)$ auf der Menge der zulässigen Lösungen $Sol(I)$ für jede Instanz $I \in D_\Pi$ und
- einem polynomiellen Algorithmus, der zur einer Eingabe $\langle I, \sigma \rangle \in D_\Pi \times Sol(I)$ entscheidet, ob die Lösung σ hinsichtlich der Zielfunktion $c : Sol(I) \to \mathbb{Q}$ mindestens so gut ist wie die Lösungen $\sigma' \in \Gamma_\mathcal{N}(\sigma)$, oder aber eine Lösung $\sigma' \in \Gamma_\mathcal{N}(\sigma)$ zurückliefert mit Zielfunktionswert $c(\sigma')$ besser als $c(\sigma)$.

Anhand des Problems MAX CUT läßt sich das gut illustrieren. Betrachte den Graphen G, der aus einem Dreieck v_2, v_3, v_4 durch Anheften eines Blattes v_1 an v_3 hervorgeht. Wenn der Algorithmus die Knoten von G wie durch die Indizes angedeutet verarbeitet, konstruiert er ggf. die Partition $V_1 = \{v_1, v_3, v_4\}$ und $V_2 = \{v_2\}$ mit $|\langle V_1, V_2 \rangle| = m/2$. Der Knoten $v_3 \in V_1$ hat nun aber mehr Nachbarn in V_1 als in V_2. Es wäre also klüger, ihn aus der Menge V_1 herauszunehmen und V_2 zuzuordnen – wodurch sich der Schnitt um (mindestens) eine Kante vergrößern würde. In diesem Beispiel wäre der gewonnene Schnitt sogar optimal. Da sich die Anzahl der Kanten im Schnitt $\langle V_1, V_2 \rangle$ bei jedem solchen Schritt vergrößert, terminiert dieses sogenannte 1-OPT-Verbesserungsverfahren ausgehend von einem beliebigen Schnitt in G nach höchstens m Schritten in einer sogenannten *lokal optimalen* Lösung, die von diesem Verfahren nicht weiter verbessert werden kann. Wie allerdings der Graph $K_{r,r}$, $r \in \mathbb{N}$ gerade, zeigt, dessen Bipartitionsklassen gleichmäßig auf V_1 und V_2 verteilt seien, muß eine lokal optimale Lösung nicht notwendig global optimal sein. Nach Proposition 1.1.19 hat dieser $\mathcal{O}(m^2)$-Approximationsalgorithmus aber immerhin ebenfalls Güteratio 2. Beachte, daß die Greedy-Heuristik nach SAHNI und GONZALEZ u.U. genau diese Partition des $K_{r,r}$ konstruiert, so daß sogar das Hintereinanderschalten der Greedy-Heuristik und des 1-OPT-Verbesserungsverfahrens für MAX CUT keine bessere Güteratio erreicht.

Proposition 10.7.1 *Die Güteratio der Greedy-Heuristik für* MAX CUT *ist, selbst auf bipartiten Graphen und selbst wenn man die* 1-OPT-*Verbesserungsheuristik aufsetzt,* mindestens *2.* □

In obigem Beispiel des $K_{r,r}$ führen jedoch sukzessive Austausche zweier Knoten aus verschiedenen V_i zum Ziel (die sogenannte SWAP-Heuristik).

Im Fall der kantengewichteten Version von MAX CUT kann man zeigen, daß das 1-OPT-Verbesserungsverfahren i.a. erst nach exponentiell vielen Schritten terminiert [HL88]. Das gleiche gilt für die naheliegende Heuristik FLIP beim Klausel-gewichteten MAX SAT

– Problem, eine erhaltene Lösung sukzessive durch Flippen (Umsetzen) des Wahrheitswertes einer Variable zu verbessern zu versuchen [Kre90]. In beiden Fällen kann man zudem zeigen, daß es NP-schwer ist, eine lokal optimale Lösung anzugeben, die eine dieser Heuristiken konstruieren würde [SY91, Kre90].

Eines der wichtigsten kombinatorischen Optimierungsprobleme überhaupt ist MINIMUM BISECTION (auch GRAPH PARTITIONING genannt). Hierbei möchte man einen Graphen in möglichst „unabhängige“, gleich große Teile zerlegen, wobei die Kanten des Graphen Abhängigkeiten modellieren. Wichtige Anwendungen gibt es u.a. im VLSI-Design, siehe z.B. [Len90].

MINIMUM BISECTION:

INSTANZ: ein Graph G der Ordnung $n = 2\ell$, $\ell \in \mathbb{N}$, und ein $k \in \mathbb{N}$,
FRAGE: gibt es eine Partition $U \dot{\cup} V \setminus U$ von V mit $|U| = |V \setminus U|$ und $|\langle U, V \setminus U \rangle| \leq k$?

Übung 10.7.2 a)[GJS76] *Die Probleme* MINIMUM BISECTION *und* MAXIMUM BISECTION *sind* NP*-vollständig.*
b) *Die* SWAP*-Verbesserungsheuristik für* MINIMUM BISECTION*, die in jedem Schritt zwei Knoten aus den beiden (gleich großen) Partitionsklassen gegeneinander austauscht, bis sich keine Verbesserung mehr einstellt, liefert i.a. beliebig schlechte Lösungen: Man konstruiere eine Folge von Graphen $(G_k)_{k=6,8,\ldots}$, die jeweils eine Bisektion mit null Kanten und eine bezüglich* SWAP *lokal optimale Bisektion mit k Kanten besitzen.* □

Mehr zum GRAPH PARTITIONING in [BCLS87, JPY88, JAMS89, JS93, FJ95, DLMS96].

JOHNSON, PAPADIMITRIOU und YANNAKAKIS [JPY88] studierten die Komplexität des Problems, lokal optimale Lösungen für kombinatorische Optimierungsprobleme zu konstruieren, indem sie die folgende Komplexitätsklasse PLS definierten.

Definition 10.7.3 *Ein kombinatorischen Optimierungsproblem Π zusammen mit einer Nachbarschaft $\mathcal{N}(I)$ auf der Menge der zulässigen Lösungen $Sol(I)$ für jede Instanz $I \in D_\Pi$ gehört zur Klasse* PLS *(für polynomial-time local search), wenn es einen primale Heuristik (die zu einer Instanz $I \in D_\Pi$ in polynomieller Zeit eine zulässige Lösung $\sigma \in Sol(I)$ produziert) und eine Verbesserungsheuristik zum Nachbarschaftsgraph $\mathcal{N}(I)$ gibt.*

Unter dem in diesem Zusammenhang natürlichen Reduzierbarkeitsbegriff konnten bemerkenswert einfache lokale Suchprobleme als PLS-vollständig klassifiziert werden.

Satz 10.7.4
(i) [SY91] *Lokal optimale Lösungen von* MAX CUT *hinsichtlich der 1-*OPT*-Nachbarschaft zu finden, ist* PLS*-vollständig.*
(ii)[SY91] *Lokal optimale Lösungen von* MINIMUM BISECTION *hinsichtlich der* SWAP*-Nachbarschaft zu finden, ist* PLS*-vollständig.*
(iii)[Kre90] *Lokal optimale Lösungen von* SAT *hinsichtlich der* FLIP*-Nachbarschaft zu finden, ist* PLS*-vollständig.* □

Es verwundert nun nicht mehr, daß auch die beiden wohl erfolgreichsten lokalen Verbesserungsheuristiken überhaupt: die sogenannte LIN-KERNIGHAN-Heuristik [LK73] für das TRAVELING SALESMAN PROBLEM, siehe [Pap92], und die KERNIGHAN-LIN-Heuristik [KL70] für MINIMUM BISECTION, siehe [JPY88], PLS-vollständig sind.

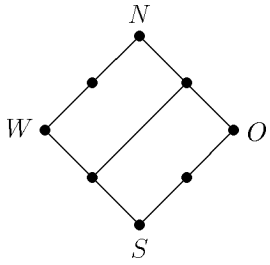
Die Klassifikation eines Problems als PLS-vollständig bedeutet, daß eine lokale Verbesserungsheuristik für dieses Problem durch Wahl entsprechender Gewichte für ihre Instanzen jedes andere Problem in PLS simulieren kann. Dies mutet unwahrscheinlich an, ist doch LINEAR PROGRAMMING $\in$ PLS. D.h. jeder polynomielle Algorithmus für ein PLS-vollständiges Problem muß „mindestens so intelligent“ sein wie die Ellipsoid-Methode, die LIN-KERNIGHAN- oder die KERNIGHAN-LIN-Heuristik. Andererseits gilt:

Satz 10.7.5 (JOHNSON, PAPADIMITRIOU, YANNAKAKIS 1988)
Wenn ein lokales Suchproblem $\Pi \in$ PLS NP*-schwer ist, so ist* NP = co-NP. □

Während man einerseits mit lokalen Verbesserungsalgorithmen wie beispielsweise der LIN-KERNIGHAN-Heuristik für das TRAVELING SALESMAN PROBLEM in der Praxis sehr gute Resultate erzielt, kennt man auf der anderen Seite keine rigorosen Güteschranken für diese Heuristiken. Ein hübsches Ergebnis von PAPADIMITRIOU und STEIGLITZ [PS77] besagt allerdings, daß im Fall des TRAVELING SALESMAN PROBLEMS von *keiner* Verbesserungsheuristik exakte Lösungen zu erwarten sind. Hierzu beweisen wir zunächst die folgende Verschärfung von HAMILTONIAN CIRCUIT: Beim Problem RESTRICTED HAMILTONIAN CIRCUIT ist nach der Existenz eines HAMILTONkreises in einem Graphen G gefragt, wobei dem Graphen noch die Codierung eines HAMILTONpfades in G beiliegt. Wie die nächste Proposition zeigt, hilft die Kenntnis eines HAMILTONpfades in G jedoch nicht bei der Suche nach einem HAMILTONkreis.

Proposition 10.7.6 RESTRICTED HAMILTONIAN CIRCUIT *ist* NP*-vollständig.*

Beweis. Wir transformieren eine Instanz $G = (V, E)$ von HAMILTONIAN CIRCUIT zu einer Instanz $R(G)$ von RESTRICTED HAMILTONIAN CIRCUIT. Sei $V = \{v_1, \ldots, v_n\}$. Der Graph $R(G)$ enthält für jeden Knoten $v_i \in V$ eine Kopie des folgenden Graphen B:



Die Eckknoten in der i-ten Kopie B_i von B seien mit $\{N_i, S_i, W_i, O_i\}$ bezeichnet. Die Graphen B_i sind nun wie folgt untereinander verbunden. Für $i = 1, \ldots, n-1$ enthält $R(G)$ die Kante $\{S_i, N_{i+1}\}$, und für alle $\{v_i, v_j\} \in E$ die Kanten $\{W_i, O_j\}$ und $\{W_j, O_i\}$. Beachte, daß jeder Hamiltonpfad in $R(G)$ die Knoten einer Kopie von B vollständig abläuft, bevor er die nächste betritt. Ferner kann er die Knoten einer Kopie von B nur in West-Ost-, Ost-West-, Nord-Süd- oder Süd-Nord-Richtung durchlaufen. Offenbar enthält $R(G)$ einen HAMILTONpfad (der in N_1 beginnt und S_n endet). Ein HAMILTONkreis in G induziert einen in $R(G)$. Umgekehrt durchläuft aber jeder HAMILTONkreis in $R(G)$ entweder *jede* Kopie von B in Ost-West- oder *jede* Kopie in Nord-Süd-Richtung. Da $R(G)$ die Kante $\{S_n, N_1\}$ nicht enthält, durchläuft also jeder HAMILTONkreis in $R(G)$ jede Kopie von B in Ost-West-Richtung und induziert somit einen HAMILTONkreis in G. □

Sei (K_n, c) eine Instanz des TRAVELING SALESMAN PROBLEMS mit Dreiecksungleichung und $C \subseteq E(G)$ eine TSP-Tour in G. Das Problem, zu entscheiden, ob C eine optimale TSP-Tour in (K_n, c) ist, nennen wir ΔTSP OPTIMALITY.

Satz 10.7.7 (PAPADIMITRIOU, STEIGLITZ 1977)
ΔTSP OPTIMALITY *ist* co-NP-*vollständig.*

Beweis. ΔTSP OPTIMALITY ist offensichtlich in co-NP, denn jede Tour C' in (K_n, c), die (hinsichtlich der Gewichtsfunktion c) kürzer ist als C, beweist, daß $\langle K_n, c, C\rangle$ eine *NEIN*-Instanz von ΔTSP OPTIMALITY ist. Ein solches Zertifikat kann offensichtlich in polynomieller Zeit überprüft werden. Um die co-NP-Vollständigkeit zu zeigen, reduzieren wir von RESTRICTED HAMILTONIAN CIRCUIT. Sei also $\langle G, P\rangle$ eine Instanz von RESTRICTED HAMILTONIAN CIRCUIT, d.h. insbesondere P ist ein HAMILTONpfad in G. Definiere eine Gewichtsfunktion c auf den Kanten des K_n, $n = |V(G)|$, wie folgt:

$$c(e) := \begin{cases} 1, & \text{falls } e \in E(G), \\ 2, & \text{falls } e \notin E(G). \end{cases}$$

Offensichtlich genügt c der Dreiecksungleichung, und der HAMILTONpfad P induziert einen HAMILTONkreis C in (K_n, c) vom Gewicht höchstens $n+1$. Falls $c(C) = n$, so stellt C offenbar einen HAMILTONkreis in G dar, andernfalls besitzt G einen HAMILTONkreis genau dann, wenn $\langle K_n, c, C\rangle$ nicht optimal ist. □

Korollar 10.7.8 *Falls* P $\neq$ NP, *ist keine lokale Verbesserungsheuristik für* ΔTSP, *die polynomielle Laufzeit pro Iterationsschritt hat, exakt.*

Beweis. Insbesondere könnte eine solche Verbesserungsheuristik TSP OPTIMALITY entscheiden. □

10.8 Ein PAS für PLANAR INDEPENDENT SET

Das Problem INDEPENDENT SET bleibt, wie in Abschnitt 8.3 gesehen, auch auf der Klasse der planaren Graphen NP-vollständig. Es zeigt sich nun, daß INDEPENDENT SET hier immerhin approximierbar ist; für allgemeine Graphen ist dagegen selbst das Approximationsproblem für INDEPENDENT SET NP-schwer (vgl. Abschnitt 11.2).

Übung 10.8.1 *Auf (dreiecksfreien) planaren Graphen G hat* GREEDY_MIN_IS *Güteratio höchstens 5 (3).*

Ein $\mathcal{O}(n \log n)$ Algorithmus für INDEPENDENT SET mit Güteratio 2 für alle planaren Graphen findet sich in [CNS82]. Unter Benutzung der Separatoren-Technik wird in diesem Abschnitt sich nun zeigen, daß PLANAR INDEPENDENT SET in polynomieller Zeit *beliebig gut* approximierbar ist.

Definition 10.8.2 *Ein* polynomielles Approximationsschema *(kurz* PAS*) für ein kombinatorisches Optimierungsproblem* II *ist eine Menge von Algorithmen* $\{A_\epsilon : \epsilon > 0\}$, *so daß* A_ϵ *ein* $(1+\epsilon)$*-Approximationsalgorithmus für* II *ist für alle* $\epsilon > 0$. *Ein Approximationsschema* $\{A_\epsilon : \epsilon > 0\}$ *heißt* vollpolynomial *(kurz* FPAS*), falls seine Laufzeit zudem polynomiell in* $1/\epsilon$ *ist.*

Ein FPAS zeigt also im Gegensatz zu einem PAS einen verhaltenen Trade-Off zwischen Approximationsgüte und Laufzeit. Ein FPAS ist hinsichtlich der Laufzeit bezüglich ϵ das beste, was man für ein NP-schweres Optimierungsproblem in polynomieller Zeit erwarten kann:

Proposition 10.8.3 *Sei* Π *ein NP-schweres kombinatorisches Optimierungsproblem mit Optimalwerten aus* $\mathbb{N}$. *Dann gibt es kein Approximationsschema* $\{A_\epsilon : \epsilon > 0\}$ *für* Π, *so daß die Laufzeit von* A_ϵ *polynomiell in der Codierungslänge* $\langle\epsilon\rangle \sim \log 1/\epsilon$ *von* ϵ *ist.*

Beweis. Sei Π o.B.d.A. ein Minimierungsproblem. Der Algorithmus A_ϵ mit

$$OPT(I) \;\leq\; A_\epsilon(I) \;\leq\; (1+\epsilon)\, OPT(I)$$

löst offenbar das Entscheidungsproblem zu Π, falls

$$(1+\epsilon)\, OPT(I) \;<\; (1+\frac{1}{OPT(I)})\, OPT(I) \;=\; OPT(I)+1.$$

Wenn man also zur Eingabe I zunächst $A_2(I)$ berechnet und dann $\epsilon_0 := (A_2(I)+1)^{-1}$ setzt, so wird wegen $OPT(I) \leq A_2(I)$

$$\epsilon_0 \;=\; \frac{1}{A_2(I)+1} \;<\; \frac{1}{A_2(I)} \;\leq\; \frac{1}{OPT(I)},$$

d.h. der dadurch definierte Algorithmus A_{ϵ_0} arbeitet exakt. Zudem ist offenbar die Codierungslänge $\langle\epsilon_0\rangle \sim \log 1/\epsilon_0$ von ϵ_0 polynomiell in $\langle I\rangle$, denn $A_2(I)$ ist die Ausgabe eines in I polynomiellen Algorithmus, und somit A_{ϵ_0} ein polynomieller Algorithmus. □

Leider können wir für das Problem Planar Independent Set wie für viele andere NP-schwere Optimierungsprobleme nicht einmal die Existenz eines FPAS erwarten.

Proposition 10.8.4 (Garey, Johnson 1978) *Sei* Π *ein* NP*-schweres kombinatorisches Optimierungsproblem mit*

- $OPT(I) \in \mathbb{N}$ *für alle* $I \in D_\Pi$;
- *es gibt ein Polynom* q, *so daß* $OPT(I) < q(\langle I\rangle)$ *für alle* $I \in D_\Pi$.

Dann gibt es kein FPAS *für* Π, *außer es ist* P= NP.

Beweis. Sei $\{A_\epsilon : \epsilon > 0\}$ ein FPAS für Π, und Π o.B.d.A. ein Maximierungsproblem. Der Algorithmus A_{ϵ_0} zu $\epsilon_0 := \epsilon_0(I) := q(\langle I\rangle)^{-1}$ ist dann polynomiell in $\langle I\rangle$ und $1/\epsilon_0 = q(\langle I\rangle)$, also ein polynomieller Algorithmus. Darüberhinaus gilt:

$$\begin{array}{rrcll} & OPT(I) & \leq & (1+\epsilon_0)\, A_{\epsilon_0}(I) & \\ \Rightarrow & OPT(I) - A_{\epsilon_0}(I) & \leq & \epsilon_0 \cdot A_{\epsilon_0}(I) & \leq \quad \epsilon_0 \cdot OPT(I) \quad < 1 \\ \Rightarrow & OPT(I) & = & A_{\epsilon_0}(I). & \end{array}$$

□

Das PAS für Planar Independent Set, das wir im folgenden entwickeln, benutzt Satz 8.2.5, um den gegebenen Graphen in kleine Teilgraphen zu separieren – dies ist der „divide"-Teil –, auf denen das Problem exakt gelöst wird – was den „conquer"-Teil darstellt. Die Approximationslösung besteht dann aus der Vereinigung der Teillösungen.

Satz 10.8.5 (Lipton, Tarjan 1980) *Für jede Funktion* $k = k(n) \leq n$ *gibt es einen Algorithmus* A_k, *der zu jedem planaren Graphen* $G = (V, E)$ *auf* n *Knoten mit maximum stabiler Knotenmenge* S^* *in* $\mathcal{O}(n2^k + n\log n)$ *Schritten eine stabile Menge* S *bestimmt mit*

$$\frac{|S^*|}{|S|} \;\leq\; 1 + \frac{40}{\sqrt{k}}.$$

Beweis. Für ein gegebenes k sei $\tilde{k} := \max\{41^2, k\}$ und A_k der folgende Algorithmus.

1. Schritt: Wende den Algorithmus von Satz 8.2.5 mit $\epsilon = \tilde{k}/n$ auf den Graphen G an. liefert dieser einen Separator C, so daß alle Komponenten von $G[V\backslash C]$ höchstens $\tilde{k}$ Knoten enthalten.

2. Schritt: Bestimme durch vollständige Enumeration in jeder Komponente von $G[V\backslash C]$ eine maximum stabile Menge. Sei S die Vereinigung dieser unabhängigen Mengen.

(Korrektheit) Offenbar ist S unabhängig.
(Approximationsgüte) Da G als planarer Graph vierfärbbar ist, gilt $|S^*| \geq n/4$. Nach Satz 8.2.5 und wegen $\epsilon n \geq \tilde{k} \geq 41^2$ hat C also die Größe

$$|C| \;\leq\; \left(2 + \frac{6}{41^2}\right)\sqrt{6}\frac{n}{\sqrt{\tilde{k}}} \;\leq\; 4\left(2 + \frac{6}{41^2}\right)\sqrt{6}\frac{|S^*|}{\sqrt{\tilde{k}}}.$$

Mit der Setzung $d := 4\left(2 + \frac{6}{41^2}\right)\sqrt{6}$ folgt, da S maximum stabil in $G[V\backslash C]$ ist,

$$|S^*| \;\leq\; |S| + |C| \;\leq\; |S| + d\frac{|S^*|}{\sqrt{\tilde{k}}}$$

womit

$$\frac{|S^*|}{|S|} \;\leq\; \frac{1}{1 - \frac{d}{\sqrt{\tilde{k}}}}.$$

Nach Wahl von $\tilde{k} \geq 41^2$ gilt nun aber wie gewünscht

$$\frac{1}{1 - \frac{d}{\sqrt{\tilde{k}}}} \;\leq\; 1 + \frac{40}{\sqrt{k}}.$$

(Laufzeit) Der erste Schritt benötigt nach Satz 8.2.5 $\mathcal{O}(n \log \frac{n}{\tilde{k}}) = \mathcal{O}(n \log n)$ Schritte. In einer Komponente der Größe n_i (bzw. $\leq 41^2$) können wir nach Proposition 1.5.2 beispielsweise in $\mathcal{O}(1.39^{n_i})$ Schritten (bzw. in konstanter Zeit) eine maximum unabhängige Menge bestimmen. Daher ist die gesamte Rechenzeit für den 2. Schritt beschränkt durch

$$\mathcal{O}\left(\max\{\textstyle\sum 2^{n_i} \mid 0 \leq n_i \leq \tilde{k} \text{ und } \sum n_i \leq n\}\right) \;=\; \mathcal{O}(n2^{\tilde{k}}) \;=\; \mathcal{O}(n2^k). \qquad \square$$

Korollar 10.8.6 *Das Problem* Planar Independent Set *besitzt ein* PAS.

Beweis. Sei ein $\epsilon > 0$ (fest) gegeben und $k(\epsilon) := \frac{1600}{\epsilon^2}$. Falls nun der Eingabegraph G weniger als $k(\epsilon)$ viele Knoten hat, so bestimmen wir eine maximum stabile Menge in G durch Enumeration in konstanter Laufzeit. Ansonsten benutzen wir den Approximationsalgorithmus $A_{k(\epsilon)}$ mit Güteratio $1 + \frac{40}{\sqrt{k(\epsilon)}} \leq 1 + \epsilon$. $\square$

Da die Laufzeit des Algorithmus $A_{k(\epsilon)}$ jedoch, wie gesehen, exponentiell in $k(\epsilon)$ ist, ist dieses Resultat lediglich von theoretischer Bedeutung. Wenn man auf der anderen Seite in Satz 10.8.5 $k(n) := \log\log n$ setzt, so erhält man zwar einen Approximationsalgorithmus mit Laufzeit $\mathcal{O}(n \log n)$ und Güteratio $1 + \frac{40}{\sqrt{\log\log n}}$; dieser Algorithmus hat allerdings erst für Graphen mit mehr als $2^{2^{1600}}$ Knoten Gütegarantie 2.

Übung und Anmerkung

Übung 10.8.7 [CNS82] *In einem planaren Graphen kann in $\mathcal{O}(n \log n)$ Schritten*
a) ein induzierter bipartiter Subgraph,
b) ein induzierter Wald,
c) ein induzierter kreisartig planarer Subgraph
bestimmt werden, dessen Größe nur um einen Faktor von $1 - \mathcal{O}(1/\sqrt{\log \log n})$ von der eines maximum induzierten bipartiten (kreisfreien, kreisartig planaren) Subgraphen abweicht.

BAKER [Bak94] gibt ein effizienteres Approximationsschema an, das in $\mathcal{O}(8^k kn)$ Laufzeit eine Approximationsgüte von $1 + \frac{1}{k}$ erreicht. Dabei unterteilt sie einen gegebenen planaren Graphen in sogenannte k-kreisartig planare Graphen und benutzt die Methode der dynamischen Programmierung. Diese Idee ist ebenso auf die Probleme VERTEX COVER, DOMINATING SET und PARTITION INTO TRIANGLES anwendbar.

10.9 GRAPH COLORING und INDEPENDENT SET

Das erste Ziel dieses Abschnitts ist der Approximationsalgorithmus MIS_COLOR von JOHNSON für GRAPH COLORING; dieser war der erste Approximationsalgorithmus für GRAPH COLORING mit sublinearer Güteratio. Sodann werden wir sehen, wie sich durch eine *gleichzeitige* Approximation von INDEPENDENT SET und GRAPH COLORING Approximationsalgorithmen für beide Probleme mit Güteratio $\mathcal{O}(\frac{n}{(\log n)^2})$ ergeben. Darüberhinaus zeigen wir einfache Nicht-Approximierbarkeitsresultate für diese Probleme.

Graph Coloring. Zum Färben allgemeiner Graphen haben wir den Greedy-Färbungsalgorithmus in diversen Variationen kennengelernt. Die Anzahl $\mathcal{A}(G)$ von Farben, die er vergibt, konnten wir a priori durch $\Delta(G) + 1$ abschätzen. 2-färbbare Graphen konnten wir durch Breitensuche in linearer Zeit erkennen und färben, o.B.d.A. haben also die betrachteten Graphen chromatische Zahl $\chi(G) \geq 3$. Wir erhalten somit einen Färbungsalgorithmus mit Güteratio

$$\frac{\mathcal{A}(G)}{\chi(G)} \leq \frac{\Delta(G)+1}{3}.$$

Ändert man die bipartiten Graphenfamilien $\{K_{k,k} - M_k\}_{k \in \mathbb{N}}$ (M_k sei jeweils ein perfektes Matching in $K_{k,k}$) leicht ab, so sieht man, daß diese Schranke auch bestmöglich ist. Die Approximationsgüte des Greedy-Färbungsalgorithmus ist also $\Theta(\Delta(G))$ und dasselbe gilt für verschiedene Varianten, siehe [Joh74b, Mit76].

Durch Enumeration erhält man:

Proposition 10.9.1 *Für jede Konstante $0 < c \leq 1$ gibt es einen linearen Approximationsalgorithmus $\mathcal{A}$ für* GRAPH COLORING *mit Güteratio $R_{\mathcal{A}}(n) \leq \max\{1, c \cdot n\}$.*

Beweis. O.B.d.A. enthalte G mehr als $2/c$ viele Knoten - sonst färbt beispielsweise der Greedy-Algorithmus angewandt auf alle Permutationen von V den Graphen in höchstens $(2/c)!\binom{2/c}{2} = const$ vielen Schritten optimal, d.h. mit Güteratio 1.

Partitioniere $V(G)$ in $\lfloor cn \rfloor$ möglichst gleich große Teile, d.h. Teile der Größe $\lfloor n/\lfloor cn \rfloor \rfloor$ oder $\lceil n/\lfloor cn \rfloor \rceil$. Dann enthält jeder Teil $M \subseteq V$ (wegen $n > 2/c$) höchstens $\frac{n}{cn-1} + 1 \leq \frac{2}{c} + 1 = const$ viele Knoten, und der jeweils induzierte Graph $G[M]$ kann daher wieder in konstanter Zeit optimal, und das heißt mit höchstens $\chi(G)$ vielen Farben, gefärbt werden. Verwendet man dabei für jeden Teil einen neuen Satz Farben, so berechnet dieser

Algorithmus in insgesamt linearer Zeit eine zulässige Färbung von G mit Güteratio

$$\frac{\#\text{ Farben}}{\chi(G)} \leq \frac{\lfloor cn \rfloor \cdot \chi(G)}{\chi(G)} \leq cn.$$

Im Gegensatz zu Proposition 10.9.9 schlagen sich hier also die Kosten für die Enumeration in der Proportionalitätskonstante nieder. □

Es erhebt sich die Frage, ob es überhaupt Approximationsalgorithmen für GRAPH COLORING mit sublinearer Güteratio gibt bzw. geben kann.

Proposition 10.9.2 *Unter der Voraussetzung* P $\neq$ NP *gibt es keinen polynomiellen Approximationsalgorithmus $\mathcal{A}$ für* GRAPH COLORING *mit Güteratio $R_{\mathcal{A}}(n) < 4/3$.*

Beweis. Angenommen, es gäbe einen Approximationsalgorithmus $\mathcal{A}$ für GRAPH COLORING und also für CHROMATIC NUMBER, der für jeden Graphen G eine zulässige Knotenfärbung mit $\mathcal{A}(G)$ Farben konstruiert, so daß

$$\begin{aligned} & \mathcal{A}(G) < (1 + 1/3)\chi(G) \\ \Leftrightarrow \quad & \mathcal{A}(G) - \chi(G) < \chi(G)/3. \end{aligned}$$

Dann ergibt sich daraus wie folgt ein polynomieller Algorithmus für das Entscheidungsproblem 3-COLORABILITY.
Fall 1: Der Eingabegraph ist 3-färbbar.
Aus $\chi(G) \leq 3$ folgt $\mathcal{A}(G) - \chi(G) < 1$, d.h. $\mathcal{A}(G) = \chi(G) \leq 3$.
Fall 2: Der Eingabegraph ist nicht 3-färbbar.
Aus $\chi(G) > 3$ folgt $\mathcal{A}(G) \geq \chi(G) \geq 4$.
Je nach dem, ob der Algorithmus $\mathcal{A}$ höchstens 3 oder mindestens 4 Farben verwendet, ist der Eingabegraph also 3-färbbar oder nicht. Aus der NP-Vollständigkeit von 3-COLORABILITY folgt mithin P = NP. □

Bis vor kurzem war die folgende Verbesserung dieses Ergebnisses der Stand der Dinge:

Satz 10.9.3 (GAREY, JOHNSON 1976) *Für jeden Färbungsalgorithmus $\mathcal{A}$ gilt $R_{\mathcal{A}}^{\infty}(n) \geq 2$.* □

Ein Beweis findet sich auch in [GJ79]. In Abschnitt 11.3 werden wir dagegen den folgenden, viel stärkeren Satz kennenlernen:

Satz 10.9.4 (LUND, YANNAKAKIS 1993) *Unter der Voraussetzung* P $\neq$ NP *gibt es ein $\epsilon > 0$, so daß kein Approximationsalgorithmus für $\chi(G)$ existiert mit Güteratio $\leq n^{\epsilon}$.* □

Insbesondere müssen wir uns also mit Approximationsalgorithmen für GRAPH COLORING zufrieden geben, deren Güteratio gegen Unendlich strebt bei wachsender Eingabelänge. Nach einer Reihe von Arbeiten verschiedener Autoren konnten FEIGE und KILIAN sogar das folgende negative Resultat zeigen.

Satz 10.9.5 (FEIGE, KILIAN 1996) *Falls nicht* NP $\subseteq$ ZPP, *gibt es für kein $\epsilon > 0$ einen Approximationsalgorithmus für $\chi(G)$ mit Güteratio $n^{1-\epsilon}$.*

Es stellt sich die Frage, ob es überhaupt Färbungsalgorithmen mit Güteratio $o(n)$ gibt. Betrachten wir zunächst den Algorithmus MAXIMUM_IS_COLOR, der sukzessive eine maximum stabile Menge aus dem durch die ungefärbten Knoten induzierten Subgraphen entfernt und mit einer neuen Farbe färbt.

```
FUNCTION MAXIMUM_IS_COLOR (G) : INTEGER;

BEGIN
  U := V;                                          {ungefärbte Knoten}
  k := 0;
  WHILE U ≠ ∅ DO BEGIN
    k := k + 1;
    finde eine maximum stabile Knotenmenge S in G[U];
    FOR u ∈ S DO Farbe[u] := k;
    U := U − S;
  END;
  MAXIMUM_IS_COLOR:= k;
END;
```

Da wir das Problem, eine $\chi(G)$-Färbung zu finden, als SET COVERING Problem auffassen können, wenn wir wir die Menge der Hyperkanten $\mathcal{F}$ als die Menge der (i.a. exponentiell vielen) stabilen Knotenmengen in G definieren, ist der Algorithmus MAXIMUM_IS_COLOR, also nichts anderes als der Greedy-Algorithmus für dieses spezielle Mengenüberdeckungsproblem, vgl. Abschnitt 12.4. Nach Satz 12.4.2 hat er Güteratio $\leq 1 + \ln n$. Nur – leider ist schon der Schritt, eine maximum stabile Menge im Restgraphen zu finden, NP-schwer, so daß dieser Algorithmus nicht weiterhilft. Wenn man jedoch auch dieses Teilproblem approximiert und sich mit einer nur jeweils *maximalen* stabilen Menge zufrieden gibt, wie sie beispielsweise der Algorithmus GREEDY_MIN_IS für INDEPENDENT SET liefert, findet man für den so definierten Approximationsalgorithmus MIS_COLOR (vgl. Abschnitt 5.5):

Satz 10.9.6 (JOHNSON 1974b) *Der Algorithmus* MIS_COLOR *ist ein* $\mathcal{O}(nm)$*-Approximationsalgorithmus für* GRAPH COLORING *mit Güteratio* $\mathcal{O}(n/\log n)$.

Beweis. Sei $G = (V, E)$ ein Graph mit chromatischer Zahl $\chi(G) = k$. O.B.d.A. sei

$$\begin{array}{rrcl} & \log n - 2\log k & \geq & \frac{1}{2}\log n \\ \Leftrightarrow & \chi(G) & \leq & n^{1/4}, \end{array} \tag{10.18}$$

da ansonsten für die Güteratio von MIS_COLOR sofort folgt:

$$\frac{\text{MIS\_COLOR}(G)}{\chi(G)} \leq \frac{n}{n^{1/4}} = \mathcal{O}(n/\log n).$$

In einer optimalen Färbung von G hat mindestens eine Farbklasse Kardinalität $\geq n/k$, und somit gilt $\delta(G) \leq n - n/k$. Untersuchen wir, wie MIS_COLOR die erste Farbklasse S_1 konstruiert. Zunächst erhält ein Knoten x_1 vom Grad $\delta(G)$ die Farbe 1, und x_1 sowie alle seine Nachbarn werden aus G entfernt. Es verbleiben mindestens $n/k - 1$ Knoten, und der Restgraph ist natürlich ebenfalls k-färbbar. Sodann wird ein Knoten x_2 minimalen Grades in diesem Restgraphen bestimmt, mit 1 gefärbt und x_2 sowie seine Nachbarn aus dem Restgraphen entfernt. Es verbleiben analog zu oben $\geq (n/k - 1)/k - 1 = n/k^2 - 1/k - 1$

viele Knoten. Es werden mithin solange Knoten mit Farbe 1 gefärbt, wie

$$\frac{n}{k^t} - \frac{1}{k^{t-1}} - \cdots - \frac{1}{k} - 1 \;>\; 0.$$

Insbesondere folgt, daß solange eine weiterer Knoten mit Farbe 1 gefärbt werden kann, wie $n \geq k^{t+1}$ gilt, denn es ist $k \geq 2$ und

$$\sum_{i=0}^{t-1} \frac{1}{k^i} \;=\; \frac{1 - 1/k^t}{1 - 1/k} \;<\; 2.$$

Es folgt $n < k^{|S_1|+1}$ oder $|S_1| \geq \lfloor \log n / \log k \rfloor$. MIS_Color konstruiert auf diese Weise Farbklassen $S_1, S_2, \ldots, S_{\text{MIS\_Color}(G)}$. Seien $n_1 = n = |V(G)|$, n_i, $i \geq 2$, die Anzahl Knoten in $G - S_1 - \cdots - S_{i-1}$, und j, $1 \leq j \leq$ MIS_Color(G) derjenige Index, so daß $n_1 > n_2 > \ldots > n_j \geq \sqrt{n}$ und $n_{j+1} < \sqrt{n}$. Dann gilt für $i = 1, \ldots, j$:

$$|S_i| \;\geq\; \lfloor \log n_i / \log k \rfloor \;\geq\; \left\lfloor \frac{1}{2} \log n / \log k \right\rfloor,$$

woraus

$$n \;\geq\; \sum_{i=1}^{j} |S_i| \;\geq\; j \cdot \left\lfloor \frac{1}{2} \log n / \log k \right\rfloor \;>\; j \cdot \left(\frac{\log n}{2 \log k} - 1\right)$$

folgt, so daß

$$j \;\leq\; \frac{2n \log k}{\log n - 2 \log k} \;\overset{(10.18)}{\leq}\; \frac{4n \log k}{\log n}.$$

MIS_Color verwendet offenbar weniger als $j + \sqrt{n}$ Farben; Für seine Güteratio gilt also:

$$\frac{\text{MIS\_Color}(G)}{\chi(G)} \;\leq\; \frac{j}{k} + \frac{\sqrt{n}}{k} \;=\; \mathcal{O}(n / \log n). \qquad \square$$

Johnson gibt auch eine Graphenfamilie an, für die die Güteratio von MIS_Color tatsächlich $\Omega(n / \log n)$ beträgt.

Independent Set. In Abschnitt 1.5 haben wir den Greedy-Algorithmus für Independent Set untersucht und festgestellt, daß die Güteratio von Greedy_Min_IS $\Omega(n)$ beträgt. Obwohl Independent Set auf Graphen von beschränktem Grad noch immer NP-vollständig ist (vgl. Satz 1.4.13), hat Greedy_Min_IS hier jedoch beschränkte Güteratio: Da (mit den Bezeichnungen aus dem Beweis zu Proposition 1.5.6) die Mengen $u_i + \Gamma_{H_i}(u_i)$, $u_i \in S_{Greedy}$, eine Partition von V bilden und jede der Knotenmengen $u_i + \Gamma_{H_i}(u_i)$, $u_i \in S_{Greedy}$, nur jeweils höchstens $d(u_i) \leq \Delta(G)$ viele, paarweise unabhängige Knoten enthalten kann, gilt für eine maximum stabile Knotenmenge S von G

$$\alpha(G) \;=\; |S| \;=\; \sum_{u_i \in S_{Greedy}} |S \cap (u_i + \Gamma_{H_i}(u_i))| \;\leq\; |S_{Greedy}| \;\cdot\; \Delta(G).$$

Die Güteratio von Greedy_Min ist also höchstens $\Delta(G)$. (Tatsächlich ist dieses Argument für jeden Algorithmus gültig, der eine maximale stabile Knotenmenge konstruiert.) Wie schon die Graphenfamilie aus Abschnitt 1.5 zeigt, ist die Güteratio auch $\Omega(\Delta(G))$.

Mit Algorithmen, die gleichzeitig eine schärfere obere Schranke an $\alpha(G)$ als $\alpha(G) \leq n$ liefern, läßt sich die Güteratio bezüglich $\Delta(G)$ verbessern.

Proposition 10.9.7 (HALLDÓRSSON, LAU 1996a) *Es gibt einen $\mathcal{O}(\Delta(G)m)$-Approximationsalgorithmus $\mathcal{A}$ für* INDEPENDENT SET *mit Güteratio*

$$R_{\mathcal{A}}(G) \leq \left\lfloor \frac{\Delta(G)}{3} + 1 \right\rfloor .$$

Beweis. Sei $G = (V, E)$ ein Graph. Auf Graphen vom Maximalgrad 2 kann INDEPENDENT SET in linearer Zeit optimal gelöst werden, vergleiche Übung 1.4.14. Die Idee des Approximationsalgorithmus ist daher, die Knotenmenge V in $k := \left\lfloor \frac{\Delta(G)}{3} + 1 \right\rfloor$ Mengen $V_1, \ldots, V_k$ zu partitionieren, so daß für die induzierten Subgraphen $G_i := G[V_i]$, $i = 1, \ldots, k$, gilt: $\Delta(G_i) \leq 2$. Dies läßt sich durch die folgende lokale Verbesserungsheuristik erreichen: Sei $v \in V_i$ ein Knoten, der nicht in einer Partitionsklasse liegt, in der er am wenigsten Nachbarn hat. Sei $i' = argmin_{1 \leq j \leq k} |\Gamma(v) \cap V_j|$. Wenn man also v aus V_i entfernt und stattdessen $V_{i'}$ zuschlägt, so hat v offenbar im neuen Subgraphen $G_{i'}$ kleineren Grad als vorher in G_i. Dieser Verbesserungsschritt wird solange iteriert, wie es solche Knoten gibt. Da sich in jedem Schritt die Anzahl der Kanten, die zwischen verschiedenen Partitionsklassen verlaufen, um mindestens 1 erhöht, terminiert das Verfahren nach höchstens m Schritten. Für $v \in V$ sei nun $i(v)$ derjenige Index, so daß $v \in V_{i(v)}$. Wegen

$$d_{G_{i(v)}}(v) \leq \frac{\Delta(G)}{k} < 3$$

gilt $\Delta(G_i) \leq 2$ für $i = 1, \ldots, k$. Der Algorithmus berechnet nun in den Graphen G_i jeweils eine maximum stabile Menge S_i, $i = 1, \ldots, k$, und gibt eine Menge S_{i_0}, $i_0 := argmax\{|S_i| : i = 1, \ldots, k\}$ aus. Sei S^* eine maximum stabile Menge in G. Dann gilt

$$\begin{array}{rrcll} & |S_i| & \geq & |S^* \cap V_i| & \text{für } i = 1, \ldots, k \\ \Rightarrow & \sum_{i=1}^{k} |S_i| & \geq & |S^*| = \alpha(G) & \end{array}$$

und mithin

$$|S_{i_0}| \geq \frac{\sum_{i=1}^{k} |S_i|}{k} \geq \frac{\alpha(G)}{k}.$$

□

Übung 10.9.8 a) *Man gebe einen linearen Algorithmus mit derselben Güte an.* [*Hinweis:* Übung 10.6.14] **b)** *Man verallgemeinere auf das knotengewichtete* INDEPENDENT SET *Problem.* [*Hinweis:* Kapitel 9]

Tatsächlich hat schon GREEDY_MIN_IS eine Güteratio von $\sim \frac{\Delta+2}{3}$ [HR94]; die Analyse hierzu ist jedoch erheblich aufwendiger.

Wenn man einem Approximationsalgorithmus mehr Laufzeit spendiert, so kann man schon durch einen trivialen Enumerationsalgorithmus jede lineare Güteratio erreichen:

Proposition 10.9.9 *Für jede Konstante $0 < c \leq 1$ gibt es einen Approximationsalgorithmus $\mathcal{A}$ für* INDEPENDENT SET *mit Güteratio $R_{\mathcal{A}}(n) \leq \max\{1, c \cdot n\}$.*

Beweis. Sei $k = \lceil 1/c \rceil$. Dann läßt sich in jedem Graphen auf höchstens k Knoten durch vollständige Enumeration aller Teilmengen von V eine maximum stabile Menge in höchstens $2^k \cdot \binom{k}{2} = const$ vielen Schritten berechnen. Bei einem Graphen G mit $n > k$ vielen Knoten berechnen wir für sämtliche k-elementige Knotenmengen $M \in \binom{V}{k}$

eine maximum stabile Menge in $G[M]$ und merken uns die größte aufgetretene stabile Menge S. Ist nun $\alpha(G) \leq k$, so ist S sogar eine maximum stabile Menge: $|S| = \alpha(G)$. Im Fall $\alpha(G) > k$ ist immerhin $|S| = k$, woraus noch $\frac{\alpha(G)}{|S|} \leq \frac{n}{k} \leq cn$ folgt. Die Laufzeit dieses Algorithmus ist offensichtlich $\mathcal{O}(n^k)$, also polynomiell. □

Es stellt sich die Frage nach der Existenz von Approximationsalgorithmen für INDEPENDENT SET mit sublinearer Gütefunktion. Das folgende frühe Ergebnis schließt die Existenz eines Approximationsalgorithmus mit endlicher Güteratio zwar nicht aus, es macht sie aber immerhin unwahrscheinlich.

Satz 10.9.10 (GAREY, JOHNSON 1979) *Das Problem* INDEPENDENT SET *hat entweder für jedes konstante $r > 1$ einen r-Approximationsalgorithmus oder für keines.*

Beweis. Der Beweis benutzt wesentlich die Operation „Graphenprodukt“ $G_1[G_2] = (V, E)$ für zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$. Der Graph $G_1[G_2] = (V, E)$ ist wie folgt definiert. Ausgehend vom Graphen G_1, dessen Kanten rot gefärbt seien, wird jeder Knoten $v \in V_1$ durch einen Graphen G_2 mit blau gefärbten Kanten ersetzt. Anschließend werden je zwei Kopien von G_2, zwischen denen eine rote Kante verläuft, (d.h. zwischen ursprünglich in G_1 benachbarten Knoten) vollständig verbunden. Die formale Definition lautet wie folgt. $G_1[G_2] = (V, E)$ hat Knotenmenge $V = V_1 \times V_2$ und zwei Knoten (v_1, v_2) und (w_1, w_2) in V sind durch eine Kante verbunden genau dann, wenn entweder $\{v_1, w_1\} \in E_1$ gilt oder wenn $v_1 = w_1$ und $\{v_2, w_2\} \in E_2$.

Sei G ein Graph. Angenommen nun, $\mathcal{A}$ sei ein (polynomieller) Approximationsalgorithmus für INDEPENDENT SET mit Güteratio $R_{\mathcal{A}}(n) \leq R < \infty$. Zu zeigen ist, daß dann auch ein Approximationsalgorithmus $\mathcal{A}_r$ für INDEPENDENT SET mit Güteratio r für jedes $1 < r < R$ existiert. Setze $G^1 := G$ und $G^k := G^{k-1}[G]$ für $k \geq 2$. Diese Operation potenziert die Größe einer größten stabilen Menge, denn offenbar gilt für jedes $k \in \mathbb{N}$

$$\alpha(G^k) \;=\; \alpha(G)^k.$$

Der Algorithmus $\mathcal{A}$ wird daher auf G^k angewandt für ein noch zu bestimmendes (konstantes) $k := k(r) \in \mathbb{N}$. Der Graph G^k hat wegen $|G^k| = |G^{k-1}| \cdot |G|$ gerade $|G|^k$ Knoten und kann daher sicherlich in (polynomieller) Zeit $\mathcal{O}(n^{2k})$ konstruiert werden. Nach Voraussetzung liefert $\mathcal{A}$ eine stabile Menge $A^{(k)}$ in G^k mit

$$\frac{\alpha(G^k)}{|A^{(k)}|} \;\leq\; R.$$

Nun soll sozusagen die k-te Wurzel gezogen werden, d.h. es wird eine stabile Menge S in G konstruiert mit $|S|^k \;\geq\; |A^{(k)}|$. Indem wir die Konstruktion von G^k zurückverfolgen, können wir für $i = k-1, \ldots, 1$ stabile Mengen $A^{(i)}$ in G^i konstruieren, die wie folgt definiert sind: $A^{(i)}$ besteht aus all den Knoten v in G^i, so daß die Kopie von G, durch die v bei der Konstruktion von G^{i+1} ersetzt wurde, mindestens einen Knoten aus $A^{(i+1)}$ enthält; $A^{(i)}$ ist also sozusagen die Projektion von $A^{(i+1)}$ auf G^i. Nun wird für $i = k, \ldots, 1$ die Größe $\sigma^{(i)}$ einer größten stabilen Menge in G^i berechnet, die aus $A^{(i)}$ durch Restriktion auf eine der ($|G^{(i-1)}|$ vielen zur Konstruktion von G^i verwendeten) Kopien von G in G^i entsteht. Sei S eine stabile Menge in G mit $|S| = \max_{i=1,\ldots,k} \sigma^{(i)}$. Dann enthält G^k (man vollziehe die Konstruktion von G^k nach) ein stabile Menge S^k der Größe $|S|^k = |S^k| \geq |A^{(k)}|$. Beachte: die stabilen Mengen $A^{(k-1)}, \ldots, A^{(1)}$ und S können in (polynomieller) Zeit

$\mathcal{O}(\sum_{i=k}^{1} |G^i|) = \mathcal{O}(|G|^{k+1})$ konstruiert werden.
Wir erhalten somit:

$$\frac{\alpha(G)^k}{|S|^k} \;\leq\; \frac{\alpha(G^k)}{|A^{(k)}|} \;\leq\; R,$$

woraus sich die Forderung

$$\frac{\alpha(G)}{|S|} \;\leq\; R^{1/k} \;\stackrel{!}{\leq}\; r,$$

an k ergibt. Mithin stellt der obige Algorithmus für

$$k(r) := \left\lceil \frac{\log R}{\log r} \right\rceil \;=\; const$$

einen r-Approximationsalgorithmus dar. □

Im nächsten Kapitel werden wir dagegen das folgende, viel stärker Ergebnis kennenlernen:

Satz 10.9.11 (Arora, Lund, Motwani, Sudan, Szegedy 1992) *Unter der Voraussetzung* P $\neq$ NP *gibt es ein* $\epsilon > 0$*, so daß kein Approximationsalgorithmus für* $\omega(G)$ *existiert mit Güteratio* $\leq n^\epsilon$*.* □

Insbesondere muß man sich also mit Approximationsalgorithmen für Clique zufrieden geben, deren Güteratio gegen Unendlich strebt bei wachsender Eingabelänge. Nach einer langen Kette von Arbeiten verschiedener Forscher gelang es kürzlich sogar, das folgende Schwere-Resulat zu zeigen.

Satz 10.9.12 (Hastad 1996) *Falls nicht* NP = co-RP*, gibt es für kein* $\epsilon > 0$ *einen Approximationsalgorithmus für* $\omega(G)$ *mit Güteratio* $n^{1-\epsilon}$*.*

Der erste Approximationsalgorithmus überhaupt für Clique mit sublinearer Güteratio ist noch recht jung. Er wird im folgenden entwickelt.

Ein Problem mit der Greedy-Heuristik für Independent Set ist, daß sie, wenn sie einen beliebigen Knoten $v \in V$ für die stabile Menge S_{Greedy} ausgewählt hat, die gesamte Nachbarschaft $\Gamma(v)$ in G unberücksichtigt läßt, obwohl diese eine sehr große stabile Menge enthalten könnte. Der folgende rekursive Algorithmus behebt diesen Mangel und berechnet in einem Graphen H ein Tupel (C, I), so daß C eine Clique und I eine stabile Menge in H ist.

```
FUNCTION Ramsey (H);

BEGIN
  IF H = ∅ THEN
    Ramsey:= (∅, ∅)
  ELSE BEGIN
    wähle einen Knoten v ∈ V(H);
    (C1, I1) := Ramsey (H[Γ(v)]);
    (C2, I2) := Ramsey (H[V(H) \ (Γ(v) + v)]);
    Ramsey:= (max{C1 + v, C2}, max{I1, I2 + v});
  END:
END;
```

(Das Maximum über Mengen soll hier jeweils die größere der beiden Mengen bezeichnen.)

Übung 10.9.13 RAMSEY (G) *kann mit Laufzeit* $\mathcal{O}(n+m)$ *implementiert werden.*

Jede Rechnung von RAMSEY(G) für einen Graphen G läßt sich an einem Binärbaum $T = T(G)$ veranschaulichen, dessen Knoten Aufrufen der Funktion RAMSEY(H) entsprechen. In jedem dieser Aufrufe, in dem H nicht der leere Graph war, wurde ein Knoten $v \in V(G)$ ausgewählt und zu Aufrufen RAMSEY$(H[\Gamma(v)])$ bzw. RAMSEY$(H[V(H)\setminus(\Gamma(v)+v)])$ verzweigt, die dem linken bzw. rechten Nachbarn in $T(G)$ (wir stellen uns vor: unterhalb) des zu RAMSEY(H) gehörigen Knotens entsprechen. Die Kanten von T unterteilen sich entsprechend in linke und rechte Kanten. Der zu RAMSEY(G) gehörige Knoten von T heißt die Wurzel von T. Die Blätter von T entsprechen Aufrufen von RAMSEY(H) mit leerem Graphen H; wir nennen diese Knoten auch die *äußeren*, alle anderen die *innernen* Knoten von T. Die n Knoten $v \in V(H)$, die in Aufrufen RAMSEY(H), $H \neq \emptyset$, gewählt werden, definieren eine Bijektion zwischen der Menge der inneren Knoten von T und $V(G)$. Wie man leicht einsieht (Übung), ist die Anzahl der äußeren Knoten von T gerade $n+1$.

Darüberhinaus ersieht man, daß der Clique C (bzw. der stabilen Menge I), die RAMSEY(G) zurückgibt, ein Pfad in T mit genau $|C|$ linken (bzw. $|I|$ rechten) Kanten entspricht, von denen RAMSEY jeweils den oberen Endknoten (d.h. denjenigen, der näher an der Wurzel liegt) in C (bzw. I) aufnahm; und es gibt keinen Pfad mit mehr als $|C|$ linken (bzw. mehr als $|I|$ rechten) Kanten in T. Mit der Definition

$$r(s,t) := \min\{n : \text{alle gewurzelten Binärbäume mit } n \text{ inneren Knoten enthalten einen Pfad mit mindestens } s \text{ linken Kanten oder einen Pfad mit mindestens } t \text{ rechten Kanten (oder beides)}\}$$

haben wir also:

Lemma 10.9.14 *Sei G ein Graph der Ordnung n und* RAMSEY $(G) = (C, I)$. *Dann gilt* $r(|C|+1, |I|+1) \geq n+1$. □

Weiterhin ist uns nach Definition von $r(s,t)$ garantiert, daß der Algorithmus RAMSEY in Graphen der Ordnung $n \geq r(s,t)$ eine Clique der Größe mindestens s oder eine stabile Menge der Größe mindestens t (oder beides) liefert. Die kleinste Zahl k, so daß jeder Graph auf mindestens k Knoten eine Clique der Größe s oder eine stabile Menge der Größe t enthält, heißt die RAMSEYzahl $R(s,t)$. Somit gilt:

$$R(s,t) \leq r(s,t).$$

Wir bestimmen nun $r(s,t)$. $r(s,t)$ ist offenbar um genau 1 größer als das maximale $n \in \mathbb{N}$, so daß es einen Binärbaum T mit n inneren Knoten gibt, der weder einen Pfad mit s linken noch einen Pfad mit t rechten Kanten enthält. Nach obigen Überlegungen ist $r(s,t)$ somit gleich dem maximalen $a \in \mathbb{N}$, so daß es einen Binärbaum T mit a äußeren Knoten gibt, der weder einen Pfad mit s linken noch einen Pfad mit t rechten Kanten enthält. Zu jedem äußeren Knoten von T gehört ein eindeutiger Pfad von der Wurzel von T zu diesem äußeren Knoten, der sich durch die Folge der benutzten Kanten (z.B. über dem Alphabet $\{L, R\}$) codieren läßt. Nach dem Schubfachprinzip enthält in einem Binärbaum, der weder einen Pfad mit s linken noch einen Pfad mit t rechten Kanten enthält, ein solcher Pfad höchstens $s-1+t-1$ Kanten. Also gibt es höchstens $\binom{(s-1)+(t-1)}{t-1}$ viele solcher Pfade zu äußeren Knoten und damit äußere Knoten. Offensichtlich gibt es zudem auch einen Binärbaum, der genau so viele äußere Knoten besitzt, womit folgt

Lemma 10.9.15 (ERDÖS, SZEKERES 1935)

$$R(s,t) \leq r(s,t) = \binom{(s-1)+(t-1)}{t-1}.$$

Mit diesem Satz und Lemma 10.9.14 wollen wir nun eine untere Schranke für das Produkt der Kardinalitäten der von RAMSEY gefundenen Clique und unabhängigen Menge beweisen.

Lemma 10.9.16 *Sei G ein Graph der Ordnung n und* RAMSEY$(G) = (C, I)$. *Dann gilt*

$$|C| \cdot |I| \geq \frac{1}{4}(\log n)^2.$$

Beweis. Es gilt nach Lemma 10.9.14 und Lemma 10.9.15

$$\begin{aligned} |C| \cdot |I| &\geq \min\{st : s,t \in \mathbb{N} \text{ und } r(s+1,t+1) \geq n+1\} \\ &= \min\{st : s,t \in \mathbb{N} \text{ und } \binom{s+t}{t} \geq n+1\} \end{aligned} \tag{10.19}$$

Der Binomialkoeffizient $\binom{s+t}{t}$ läßt sich mit Hilfe der sogenannten Γ-Funktion für alle $(s,t) \in \mathbb{R}_+^2$ definieren. Sei also $f : \mathbb{R}_+^2 \to \mathbb{R}$ mit $f(s,t) = \binom{s+t}{t}$ für $s,t \in \mathbb{N}$ eine Fortsetzung des Binomialkoeffizienten. Dann ist (10.19) größer oder gleich

$$\min\{st : s,t \in \mathbb{R}_+ \text{ und } \binom{s+t}{t} = n+1\}. \tag{10.20}$$

Dies ist eine Extremwertaufgabe mit einer Nebenbedingung. Eine Lösung (s_0, t_0) von (10.20), die also $s_0 t_0$ minimiert, muß nach einem Satz aus der Analysis (vgl. z.B. [For84, Satz 4, Seite 78])

$$\frac{\frac{\partial f}{\partial s}(s_0,t_0)}{t_0} = \frac{\frac{\partial f}{\partial t}(s_0,t_0)}{s_0}$$

erfüllen. Eine etwas längere Rechnung zeigt, daß dies $s_0 = t_0$ impliziert. Wegen $\binom{\log n}{\log n/2} \leq 2^{\log n} = n$ kann daher (10.20) durch

$$\min\{s^2 : s \in \mathbb{R}_+ \text{ und } \binom{2s}{s} = n+1\} \quad \geq \quad \frac{1}{4}(\log n)^2$$

abgeschätzt werden. □

Falls G keine großen unabhängigen Mengen hat, falls also beispielsweise $\alpha(G) = \mathcal{O}(\log n)$ wie in zufälligen Graphen (vgl. Satz 13.1.5) gilt, dann findet RAMSEY eine Clique der Größe $\Omega(\log n)$. Der folgende Algorithmus IND_SET_REMOVAL zeigt, wie wir in allgemeinen Graphen eine Approximationsgüte von $\mathcal{O}(\frac{n}{(\log n)^2})$ für CLIQUE erreichen können. IND_SET_REMOVAL entfernt sukzessive die von RAMSEY gefundenen unabhängigen Mengen aus dem Graphen. Bei jedem Schritt entfernen wir nach Lemma 10.9.16 entweder eine große unabhängige Menge I_i, oder wir finden eine große Clique. Falls wir also in keinem Schritt eine große Clique finden konnten, so müssen alle stabilen Mengen groß gewesen sein und damit die Partition $\mathcal{I} = \{I_1, \ldots, I_k\}$ der Knotenmenge in unabhängigen Mengen

klein. Wegen $\omega(G) \leq \chi(G)$ ist daher in diesem Fall auch die Cliquenzahl des Graphen klein.

```
FUNCTION IND_SET_REMOVAL (G);
BEGIN
  k := 0;
  G_1 := G;
  REPEAT
    k := k + 1;
    (C_k, I_k) := RAMSEY (G_k);
    G_{k+1} := G_k - I_k;
  UNTIL G_{k+1} = ∅;
  IND_SET_REMOVAL:= (max_{i=1}^{k} C_i, {I_1, ..., I_k});
END;
```

Der Algorithmus hat die folgende Güteratio.

Satz 10.9.17 (BOPPANA, HALLDÓRSSON 1992) IND_SET_REMOVAL *ist ein Approximationsalgorithmus für* CLIQUE *und* GRAPH COLORING *mit Güteratio* $R_{\mathcal{A}}(n) = \mathcal{O}(\frac{n}{(\log n)^2})$.

Beweis. Seien also $C = \max\{C_i : i = 1, \ldots, k\}$ bzw. $\mathcal{I} = \{I_1, \ldots, I_k\}$ die von IND_SET_REMOVAL(G) konstruierte Clique bzw. Färbung von G. Nach Lemma 10.9.16 gilt für $i = 1, \ldots, k$:

$$|C| \cdot |I_i| \;\geq\; |C_i| \cdot |I_i| \;\geq\; (\frac{1}{2}\log|V(G_i)|)^2. \tag{10.21}$$

Sei G_i wie im Algorithmus und $V(G_i) = \{v_1, \ldots, v_{j_i}\}$ mit $1 \leq j_k < \ldots < j_1 = n$ und $j_{k+1} = 0$. Es ist also $I_i = \{v_{j_{i+1}+1}, \ldots, v_{j_i}\}$. Der Einfachheit halber sei in der folgenden Rechnung $\log 1 := 1$ gesetzt. Damit gilt

$$\begin{aligned}
k \;=\; \sum_{i=1}^{k} \frac{|I_i|}{|I_i|} \;&\overset{(10.21)}{\leq}\; 4|C| \sum_{i=1}^{k} \frac{|I_i|}{(\log j_i)^2} \\
&\leq\; 4|C| \sum_{j=1}^{n} \frac{1}{(\log j)^2} \qquad (\text{beachte } \textstyle\sum_{i=1}^{k} |I_i| = n) \\
&=\; 4|C| \left(\sum_{j=1}^{\sqrt{n}} \frac{1}{(\log j)^2} + \sum_{j=\sqrt{n}+1}^{n} \frac{1}{(\log j)^2} \right) \\
&\leq\; 4|C| \left(\sqrt{n} + \frac{n}{(\log \sqrt{n})^2} \right) \;=\; \mathcal{O}\left(\frac{|C|n}{(\log n)^2} \right).
\end{aligned}$$

Also ist

$$\frac{\omega(G)}{|C|} \frac{|\mathcal{I}|}{\chi(G)} \;\leq\; \mathcal{O}\left(\frac{n}{(\log n)^2} \right) \frac{\omega(G)}{\chi(G)} \;=\; \mathcal{O}\left(\frac{n}{(\log n)^2} \right), \tag{10.22}$$

woraus man die gewünschten Abschätzungen für die beiden Güteratios abliest. □

Wenn wir IND_SET_REMOVAL auf $\overline{G}$ anwenden, erhalten wir mit Satz 10.9.17 auch eine Approximation von INDEPENDENT SET und CLIQUE PARTITION mit Güteratio $\mathcal{O}(\frac{n}{(\log n)^2})$.

Anmerkungen und Übungen

Einen anderen Approximationsalgorithmus für GRAPH COLORING mit Güteratio $\mathcal{O}(n/\log n)$ fand HALLDÓRSSON [Hal96a]; dieser ist zudem parallelisierbar und führt zu einem „on-line" Approximationsalgorithmus mit derselben Gütegarantie. Der bisher beste Approximationsalgorithmus für GRAPH COLORING hat Güteratio $R_{\mathcal{A}} = \mathcal{O}(\frac{(\log\log n)^2}{(\log n)^3} n)$ [Hal93a]. Beinahe trivial ist dagegen die Approximation der Anzahl der „nicht benutzten Farben":

Übung 10.9.18 [DGP94, HL94] **a)** *Es gibt einen linearen (!) Färbungsalgorithmus $\mathcal{A}$, so daß $n - \chi(G) \leq 2(n - \mathcal{A}(G))$ für alle Graphen G.* [*Hinweis:* betrachte ein maximales Matching in $\overline{G}$]
b) *Es gibt einen $\mathcal{O}(n \min\{m+n^{1.5}, \overline{m}\})$-Färbungsalgorithmus $\mathcal{A}$, so daß $n-\chi(G) \leq 3/2(n-\mathcal{A}(G))$ für alle Graphen G, wobei $\overline{m} := m(\overline{G})$.* [*Hinweis:* packe Dreiecke in $\overline{G}$ und färbe den Restgraphen optimal]

Einen Approximationsalgorithmus mit Güteratio 4/3 gibt [Hal96b].

Übung 10.9.19 *Man zeige Proposition 10.9.2 auch für die asymptotische Güteratio.*

Auf zufälligen Graphen hat IND_SET_REMOVAL wegen $\frac{\omega(G)}{\chi(G)} \sim \frac{4(\log n)^2}{n}$ sogar konstante Güteratio. Doch haben hier schon die simplen Greedy-Algorithmen eine bessere Güteratio, vergleiche Kapitel 13.

HALLDÓRSSON und RADHAKRISHNAN [HR95] konnten sogar einen Approximationsalgorithmus für INDEPENDENT SET mit Güteratio $o(\Delta(G))$ angeben.

Für alle $\epsilon > 0$ ist auch die Approximation einer (kardinalitäts-) minimalen (inklusions-) maximalen unabhängigen Knotenmenge mit Faktor $n^{1-\epsilon}$ NP-schwer [Hal93b].

Übung 10.9.20 (TURÁN 1941) *Umso weniger Kanten ein Graph hat, umso größer wird die Größe einer maximum stabilen Menge. Seien n, a natürliche Zahlen. Offensichtlich existiert ein Minimum $e(n,a)$ von Kanten, die ein Graph auf n Knoten mit Stabilitätszahl $\leq a$ haben muß. Aus $m(G) < e(n,a)$ folgt dann also $\alpha(G) > a$. Der Graph $S(n,a)$ bestehe aus a möglichst gleich großen, disjunkten Cliquen. Man bestimme $e(n,a)$ und zeige, daß die Graphen $S(n,a)$ die eindeutigen extremalen Graphen sind.* [*Hinweis zur Eindeutigkeit:* Im Fall, daß G dieselbe Gradsequenz hat wie $S(n,a)$, zeige, daß G keinen $K_{1,2}$ enthält.] □

Übung 10.9.21 *Die Sätze 10.9.10 und 11.2.1 gelten auch für asymptotische Approximationsalgorithmen: Sei $\mathcal{A}$ ein Approximationsalgorithmus für* INDEPENDENT SET *mit asymptotischer Güteratio r, d.h. es gebe eine Konstante $c \in \mathbb{R}^+$, so daß $\alpha(G) \leq r\mathcal{A}(G) + c$ für alle Graphen G. Dann ist der Algorithmus, der* max $\{1, \mathcal{A}(G)\}$ *ausgibt, ein Approximationsalgorithmus mit Güteratio $r' := r + c$.*

Übung 10.9.22 [BK77, Cat78] *Sei $r \geq 3$ fest. Dann gibt es einen linearen Algorithmus, der alle Graphen G mit $\omega(G) \leq r$ (insbesondere also alle r-färbbaren Graphen) mit höchstens $\frac{r}{r+1}\Delta(G) + r$ vielen Farben färbt.* [*Hinweis:* Proposition 10.9.7]

Kapitel 11

Nicht-Approximierbarkeit

Dieser Abschnitt geht auf neueste Resultate aus der Komplexitätstheorie ein, die die Nichtapproximierbarkeit in z.T. sehr strengen Sinne von kombinatorischen Optimierungsproblemen zur Folge haben. Grundlage ist eine neue Charakterisierung der Klasse NP.

Die Ergebnisse des letzten Kapitels suggerieren, daß sich kombinatorische Optimierungsprobleme hinsichtlich des Schwierigkeitsgrades ihrer Approximationsprobleme in drei Klassen einordnen lassen.

I. **Beliebig gut approximierbare Optimierungsprobleme.** Diese Klasse enthält diejenigen Optimierungsprobleme, die ein polynomielles Approximationsschema (PAS) besitzen. In diese Klasse fällt beispielsweise, wie in Abschnitt 10.8 gesehen, das Problem Planar Independent Set. Andere graphentheoretische Beispiele sind Max Cut auf dichten Graphen [AKK95] und zwei Spezialfälle von Metric TSP, nämlich Planar Graph TSP, wo die Gewichte durch die Distanzmetrik eines zusammenhängenden, planaren Graphen gegeben sind [GKP95], und Euclidean TSP, wo eine Tour für n Punkte in der Ebene gesucht ist [Aro96].

   Die berühmtesten Vertreter dieser Klasse entstammen jedoch nicht der Graphentheorie. Das sogenannte Knapsack Problem beispielsweise ist nur aufgrund der Größe der involvierten Zahlen NP-vollständig und besitzt sogar ein FPAS [IK75]. Für das Problem Multi-Processor-Scheduling führt „vollständige Enumeration auf Anfangsstücken“ zu einem PAS [Gra66]. Für das Problem Bin Packing gibt es immerhin ein sogenanntes asymptotisches FPAS [KK82], dem sozusagen auf kleinen Instanzen eine schlechtere Approximationsgüte gestattet ist.

II. **Mit konstantem Faktor approximierbare Optimierungsprobleme.** Zu dieser Klasse zählen Optimierungsprobleme wie Node Cover, Δ-TSP, Steiner Tree, Max Cut und Max Sat, die zwar immerhin einen Approximationsalgorithmus mit endlicher Gütegarantie besitzen, aber kein Approximationsschema, d.h. es gibt ein $\epsilon > 0$, so daß es keinen Approximationsalgorithmus für ein solches Problem mit Güteratio kleiner als $1 + \epsilon$ gibt (außer P = NP).

III. **Nicht approximierbare Optimierungsprobleme.** Die dritte Klasse schließlich enthält Optimierungsprobleme, deren r-Approximationsproblem für jede endliche Güteratio $r \in \mathbb{N}$ NP-schwer ist. In diese Klasse fallen beispielsweise das allgemeine

TSP und, wie noch sehen werden, CLIQUE, DOMINATING SET, CHROMATIC NUMBER und CLIQUE COVER. Solche Probleme können aber immerhin Approximationsalgorithmen besitzen, deren Güteratio als Funktion der Eingabelänge wächst, und es gibt Probleme (wie z.B. DOMINATING SET, vgl. Abschnitt 12.4), für die es einen Approximationsalgorithmus gibt, dessen Güteratio substantiell langsamer wächst, als es bei anderen Problemen (wie z.B. CLIQUE und CHROMATIC NUMBER) möglich ist.

Die Ergebnisse dieses Kapitels besagen, daß die Probleme aus Klasse II tatsächlich nicht in Klasse I liegen und die aus Klasse III tatsächlich nicht in Klasse II (jeweils modulo der Vermutung P≠NP).

11.1 MAX-SNP und PCP

Definition 11.1.1 [PY91] *SNP, MAX-SNP.*

Satz 11.1.2 *Die folgenden Probleme sind* MAX SNP*-schwer:*

- NODE COVER *($\Delta \leq 4$) [PY91]*
- Δ-TSP *[PY93]*
- STEINER TREE *[BP89]*
- MAX CUT *[PY91]*
- MAX SAT, MAX 2SAT *[PY91]*

Die Bedeutung der Klasse MAX SNP liegt darin, daß sich für MAX-SNP-schwere Probleme untere Schranken für die Approximierbarkeit zeigen lassen.

Satz 11.1.3 (ARORA, LUND, MOTWANI, SUDAN, SZEGEDY 1992)
Unter der Voraussetzung P $\neq$ NP *besitzt kein* MAX-SNP*-schweres Problem ein* PAS.

Satz 11.1.4 (ARORA, LUND, MOTWANI, SUDAN, SZEGEDY 1992)

$$\text{NP} = \text{PCP}(\log n, 1).$$

11.2 CLIQUE

Da sich Cliquen in G und stabile Knotenmengen im Komplement $\bar{G}$ entsprechen, sind die Probleme CLIQUE und INDEPENDENT SET auf der Menge aller Graphen hinsichtlich ihrer Approximierbarkeit äquivalent. Wir untersuchen hier daher nur das Problem CLIQUE.

Satz 11.2.1 (ARORA, LUND, MOTWANI, SUDAN, SZEGEDY 1992) *Unter der Voraussetzung $P \neq NP$ gibt es ein $\epsilon > 0$, so daß kein Approximationsalgorithmus für* CLIQUE *existiert mit Güteratio $\leq n^{\epsilon}$.*

11.3 GRAPH COLORING

Satz 11.3.1 (LUND, YANNAKAKIS 1993) *Unter der Voraussetzung $P \neq NP$ gibt es ein $\epsilon > 0$, so daß kein Approximationsalgorithmus für $\chi(G)$ existiert mit Güteratio $\leq n^{\epsilon}$.*

Kapitel 12

Überdeckungsprobleme

12.1 Cliquenüberdeckung und Schnittgraph-Darstellung

Definition 12.1.1 *Sei $G = (\{1,\ldots,n\}, E)$ ein Graph. Eine Familie $S_1,\ldots,S_n$ von (nicht notwendig verschiedenen) endlichen Mengen heißt* Schnittgraph-Darstellung *von G über $S := \bigcup_{i=1}^{n} S_i$ bzw. G der Schnittgraph von $S_1,\ldots,S_n$ genau dann, wenn*

$$\{i,j\} \in E \;\Leftrightarrow\; S_i \cap S_j \neq \emptyset \qquad \text{für alle } 1 \leq i < j \leq n. \tag{12.1}$$

Graphen treten ganz natürlich als Schnittgraphen auf, wie schon das Beispiel in der Einleitung von Abschnitt 5.1 zeigte.

Lemma 12.1.2 (Szpilrajn-Marczewski 1945)
Jeder Graph besitzt eine Schnittgraph-Darstellung.

Beweis. Für einen Graphen $G = (V,E)$ setze $S := E$ und definieren für jeden Knoten $v \in V$ die Menge $S_v \subseteq S$ als die Menge der mit v inzidierenden Kanten $e \in E$. Dann ist die Bedingung (12.1) trivialerweise erfüllt. Leicht läßt sich auch erreichen, daß die Mengen S_v alle verschieden sind: man fügt jeder Menge S_v noch den Knoten v als Element hinzu. □

Da also jeder Graph ein Schnittgraph ist, ist der folgende Graphenparameter wohldefiniert:

Definition 12.1.3 *Die* Schnittzahl *$\iota(G)$ eines Graphen G ist die minimale Kardinalität einer Menge S, so daß G eine Schnittgraph-Darstellung über S besitzt, d.h.*

$$\iota(G) \;=\; \min\{|\bigcup_{v\in V} S_v| : (S_v)_{v\in V} \text{ Schnittgraph-Darstellung von } G\}.$$

Proposition 12.1.4 *Für jeden Graphen G gilt $\iota(G) \leq m$ mit Gleichheit genau dann, wenn G keine Dreiecke enthält.*

Beweis. Aus dem Beweis des letzten Lemmas folgt $\iota(G) \leq m$. Zur Gleichheit:
„$\Rightarrow$“: Enthalte $G = (V,E)$ ein Dreieck x,y,z, und sei $(S_v)_{v\in V}$ eine Schnittgraph-Darstellung von $(V, E - \{x,z\} - \{y,z\})$ über $S := \bigcup_{v\in V} S_v$ wie im Beweis von Lemma 12.1.2 mit $|S| = m(G) - 2$. Erweitert man nun die Menge S_z um das Element $e = \{x,y\}$, so erhält man eine Schnittgraph-Darstellung von G mit $m-2$ Elementen.
„$\Leftarrow$“: Sei G ein dreiecksfreier Graph und $(S_v)_{v\in V}$ eine Schnittgraph-Darstellung von G.

Dann gibt es für jede Kante $e = \{u, v\} \in E$ ein $s_e \in S_u \cap S_v$, und es gilt $s_e \notin S_w$ für alle $w \in V - u - v$ (sonst bildeten u, v, w ein Dreieck). Es folgt $|S| \geq m$. □

Beispiele. Bäume haben folglich die Schnittzahl $n - 1$, vollständig bipartite Graphen $K_{r,s}$ Schnittzahl $r \cdot s$. Der K_n hingegen hat offenbar Schnittzahl 1. □

Schnittgraph-Darstellungen gestatten also u.U. eine äußerst Speicherplatz-effiziente Codierung von Graphen. Für Anwendungen der Schnittzahl siehe [Rob85].

Übung 12.1.5 (HARARY 1974)
a) *Sei $\iota'(G)$ die minimale Kardinalität einer Menge S, so daß der Graph G eine Schnittgraph-Darstellung über S besitzt, in der alle Mengen S_v paarweise verschieden und nicht leer sind. Dann gilt auch $\iota'(G) \leq m$, falls nur jede Komponente von G mindestens drei Knoten hat.*
b) *Man bestimme $\iota(K_4 - e)$ und $\iota'(K_4 - e)$ und vergleiche mit den Schnittgraphdarstellungen aus dem Beweis von Lemma 12.1.2.*

Das Konzept der Schnittgraph-Darstellung ist eng verwandt mit dem folgenden Begriff.

Definition 12.1.6 *Sei $G = (V, E)$ ein Graph und $(C_i)_{i=1}^k$, $k \in \mathbb{N}$, eine Familie von (nicht notwendig verschiedenen) Cliquen $C_i \subseteq V$ in G. Dann heißt $(C_1, \ldots, C_k)$ eine* (Kanten-) Cliquenüberdeckung *von G, falls jede Kante $e \in E$ in mindestens einer Clique $G[C_i]$, $1 \leq i \leq k$, enthalten ist. Die* Cliquenüberdeckungszahl *$\theta_e(G)$ ist das kleinste k, so daß G eine Cliquenüberdeckung aus k Cliquen enthält.*

Eine entsprechend definierte Knoten-Cliquenüberdeckungszahl ist offenbar gleich der Cliquenpartitionszahl $\theta(G)$. Die Kanten-Cliquenpartitionszahl ist i.a. jedoch größer als die Kanten-Cliquenüberdeckungszahl, wie schon der $K_4 - e$ zeigt.
Der Zusammenhang mit Schnittgraph-Darstellungen wird im folgenden Satz sichtbar.

Satz 12.1.7 (ERDŐS, GOODMAN, PÓSA 1966)
Sei G ein Graph und $\mathcal{S}$ (bzw. $\mathcal{C}$) die Menge der Schnittgraph-Darstellungen (Cliquenüberdeckungen) von G. Dann sind die Abbildungen $f : \mathcal{S} \to \mathcal{C}$ mit

$$(S_v)_{v \in V} \mapsto (C_i)_{i \in S}, \qquad \text{wobei } C_i := \{v \in V | S_v \ni i\},\ S := \bigcup_{v \in V} S_v,$$

und $g : \mathcal{C} \to \mathcal{S}$ mit

$$(C_i)_{i \in S} \mapsto (S_v)_{v \in V}, \qquad \text{wobei } S_v := \{i \in S | C_i \ni v\},$$

bijektiv.

Beweis. Wohldefiniertheit von f. Da die Mengen S_v nach Definition für alle Knoten $v \in C_i$ den Punkt $i \in S$ enthalten, bilden die C_i's aufgrund der Schnittgraphbedingung Cliquen. Daß die C_i's auch E überdecken, sieht man wie folgt ein. Sei $e = \{u, v\} \in E$ eine Kante von G. Dann gibt es ein $i \in S_u \cap S_v$. Folglich sind die Knoten u und v und damit auch e in der Clique C_i enthalten.
Wohldefiniertheit von g. Sei $e = \{u, v\} \in E$. Dann ist nach Definition einer Cliquenüberdeckung $e \in C_i$ für ein $i \in S$. Die Mengen S_u und S_v enthalten also nach Definition beide das Element i, d.h. sie schneiden sich. Gilt andererseits $S_u \cap S_v \ni i$ für zwei Mengen S_u und S_v, so enthält C_i folglich sowohl u als auch v, d.h. es ist $e \in E$. Somit ist $(S_v)_{v \in V}$ eine Schnittgraph-Darstellung von G.
Bijektivität folgt aus $f \circ g = id = g \circ f$. □

Korollar 12.1.8 *Für jeden Graphen G gilt* $\iota(G) = \theta_e(G)$.

Wie die nächste Übung zeigt, ist (je-)der Greedy-Algorithmus für CLIQUE COVER auf der Klasse der Intervallgraphen optimal.

Übung 12.1.9 (OPSUT, ROBERTS 1981) *Für einen Intervallgraphen G gilt:*

$$\iota(G) \;=\; \theta_e(G) \;=\; \#\ \textit{maximaler Cliquen in } G.$$

Übung 12.1.10 *Seien $(S_v)_{v\in V}$ und $(C_i)_{i\in S}$ eine Schnittgraph-Darstellung bzw. eine Cliquenüberdeckung eines Graphen G, die sich gemäß der Bijektion aus Satz 12.1.7 entsprechen. Dann gilt:*

(i) Für ein $k \in \mathbb{N}$ gilt $\forall v \in V : |S_v| \leq k$ genau dann, wenn jeder Knoten $v \in V$ von höchstens k Cliquen überdeckt wird.

(ii) Alle S_v der Schnittgraph-Darstellung sind verschieden genau dann, wenn $(C_i)_{i\in S}$ knotentrennend ist: $\forall u \neq v\ \exists i \in S : \quad |C_i \cap \{u, v\}| = 1$.

(iii) Für je zwei Knoten $u \neq v$ gilt $|S_u \cap S_v| \leq 1$ genau dann, wenn die Cliquen paarweise disjunkt sind, d.h. $(C_i)_{i\in S}$ eine Cliquenpartition von E darstellt.

Wie der nächste Satz (bzw. sein Beweis) zeigt, ist das Problem, eine minimum (Kanten-) Cliquenüberdeckung (oder äquivalent eine Schnittgraph-Darstellung mit minimalem $|S|$) in einem Graphen zu konstruieren, nicht nur NP-schwer, sondern sogar genauso schwer zu approximieren wie GRAPH COLORING.

Satz 12.1.11 (KOU, STOCKMEYER, WONG 1978) *Sei $c \in \mathbb{R}^+$. Genau dann gibt es einen Approximationsalgorithmus $\mathcal{A}$ für* CLIQUE COVER *mit*

$$|\mathcal{A}(G)| \;\leq\; c\theta_e(G) + d$$

für eine Konstante $d \in \mathbb{R}^+$, wenn es einen Approximationsalgorithmus $\mathcal{A}'$ für CLIQUE PARTITION *gibt mit*

$$|\mathcal{A}'(G)| \;\leq\; c\theta(G) + d'$$

für eine Konstante $d' \in \mathbb{R}^+$.

Beweis. „$\Leftarrow$“: Sei der Graph $G = (V, E)$ eine Instanz von CLIQUE COVER. Konstruiere daraus wie folgt einen Graphen G' auf der Knotenmenge $E = \{e_1, \ldots, e_m\}$. Zwei Kanten $e_i = \{u_i, v_i\}$ und $e_j = \{u_j, v_j\}$ von G seien in G' als Knoten adjazent genau dann, wenn $G[\{u_i, v_i, u_j, v_j\}]$ eine Clique ist. Man sieht nun leicht ein, daß $\theta_e(G) = \theta(G')$, da einerseits jede (Kanten-) Cliquenüberdeckung von G eine Knoten-Cliquenüberdeckung von G' induziert und andererseits jede Cliquenpartition von G' eine (Kanten-) Cliquenüberdeckung von G.
„$\Rightarrow$“: Gegeben einen Approximationsalgorithmus $\mathcal{A}$ für CLIQUE COVER mit $|\mathcal{A}(\cdot)| \leq c\theta_e(\cdot) + d$, konstruiert der Algorithmus $\mathcal{A}'$ zu einer Instanz $G = (V, E)$ von CLIQUE PARTITION zunächst den Graphen $H := G * S_t$, der durch vollständiges Verbinden der Knotenmenge V zu einer disjunkten stabilen Menge S_t der Kardinalität $t \in \mathbb{N}$ entsteht, und läßt dann die Cliquenüberdeckung $\mathcal{A}(H)$ berechnen. Für jeden Knoten $s \in S_t$ definieren die k_s Cliquen aus $\mathcal{A}(H)$, die s enthalten, durch Restriktion auf G eine Cliquenpartition von G. Also findet sich eine Cliquenpartition $\mathcal{C}$ von G unter diesen mit

$$|\mathcal{C}| \;=\; \min_{s\in S_t} k_s \;\leq\; \frac{\sum_{s\in S_t} k_s}{|S_t|} \;\leq\; \frac{|\mathcal{A}(H)|}{t} \;\leq\; \frac{c\theta_e(H) + d}{t}.$$

Da sicher $t \cdot \theta(G)$ Cliquen genügen, um in H die Kanten des Schnitts $\langle V, S_t \rangle$ zu überdecken, und m weitere, um alle Kanten in $E(G)$ zu überdecken, gilt $\theta_e(H) \leq t \cdot \theta(G) + m$. Folglich erfüllt dieses $\mathcal{C}$:

$$|\mathcal{C}| \;\leq\; \frac{c\,(t\theta(G)+m)\;+d}{t} \;=\; c\,\theta(G) \;+\; \frac{cm+d}{t}.$$

Für $t := \lceil cm + d \rceil$ ergibt sich daher insgesamt $|\mathcal{C}| \leq c\theta(G) + 1$. Da für ein so gewähltes t die Konstruktion von H und das Finden der Cliquenpartition $\mathcal{C}$ sicherlich in polynomieller Zeit bewerkstelligt werden kann, definiert der Algorithmus $\mathcal{A}'$, der dieses $\mathcal{C}$ ausgibt, einen Approximationsalgorithmus für CLIQUE PARTITION wie er im Satz behauptet wird. □

Da die Cliquenüberdeckungszahl $\theta_e(G)$ mithin nur schwer zugänglich ist, gewinnen Abschätzungen derselben an Bedeutung.

$\theta_e(G)$ ändert sich nicht, wenn wir isolierte Knoten aus G entfernen. Ferner, nenne zwei Knoten u und v in V *äquivalent*, falls $\{u, v\} \in E$ und für alle $z \in V - u - v$ gilt $\{u, z\} \in E \Leftrightarrow \{v, z\} \in E$. Wenn also zwei Knoten u und v äquivalent in einem Graphen G sind, so erhält man offenbar aus einer Cliquenüberdeckung von G/e, $e := \{u, v\}$, eine Cliquenüberdeckung gleicher Kardinalität für G, indem man den Knoten v_e, zu dem die Kante e kontrahiert wurde, in jeder Clique, die ihn enthält, wieder durch die beiden Knoten u und v ersetzt; es gilt somit $\theta_e(G) = \theta_e(G/e)$. Durch rekursives Kontrahieren äquivalenter Knoten erhält man aus einem Graphen einen Graphen ohne äquivalente Knoten. Vom Standpunkt der Berechenbarkeit aus genügt es also, $\theta_e(G)$ für Graphen ohne isolierte und ohne äquivalente Knoten zu berechnen.

Satz 12.1.12 (GYÁRFÁS 1990) *Für einen Graphen G der Ordnung $n \geq 2$ ohne isolierte und ohne äquivalente Knoten gilt:*

$$\theta_e(G) \;\geq\; \log(n+1). \tag{12.2}$$

Beweis. Wir zeigen $n \leq 2^{\iota(G)} - 1$. Sei $(S_v)_{v \in V}$ eine Schnittgraph-Darstellung von G mit $|S| = \iota(G)$ für $S := \bigcup_{v \in V} S_v$. Da G keine isolierten Knoten enthält, gilt $S_v \neq \emptyset$ für alle $v \in V$. Außerdem, gälte $S_u = S_v \neq \emptyset$ für zwei Knoten $u \neq v$, so wären sie offenbar äquivalent. Mithin sind die S_v, $v \in V$, paarweise verschiedene, nicht-leere Teilmengen von S, und davon gibt es höchstens $2^{|S|} - 1$ viele. □

Diese beinahe triviale Schranke ist beispielsweise für die Cocktailparty-Graphen $\overline{M_{2k}}$ (auch n-dimensionale Oktaeder genannt), die aus vollständigen Graphen K_{2k} durch Herausnahme eines perfekten Matchings M_{2k} entstehen, zumindest asymptotisch scharf [GP82]:

$$\theta_e(\overline{M_{2k}}) \;=\; \log n + o(\log n).$$

Für die Komplemente von Kreisen gilt immerhin $\theta_e(\overline{C_n}) < 1.459 \log n$ [CGP85, Koh91].

GYÁRFÁS gibt die folgenden Graphen an, die (12.2) sogar mit Gleichheit erfüllen. Der Graph G_k, $k \in \mathbb{N}$, hat $n = 2^k + k$ viele Knoten und besteht aus einer Clique auf den 2^k verschiedenen 0-1-Wörtern der Länge k und einer dazu disjunkten, stabilen Menge $\{s_1, \ldots, s_k\}$, die wie folgt verbunden sind. Ein Knoten s_i, $1 \leq i \leq k$, hat genau die 0-1-Wörter als Nachbarn, die an der i-ten Stelle eine 1 haben. Der Graph G_k hat damit weder isolierte noch äquivalenten Knoten und erfüllt offenbar

$$\theta_e(G_k) \;\leq\; k+1 \;=\; (\log 2^k) + 1 \;\leq\; \left\lceil \log(2^k + k + 1) \right\rceil.$$

Die folgende Proposition verschärft wegen $\iota(G) = \theta_e(G)$ das Lemma 12.1.4 für dichte Graphen.

Satz 12.1.13 (ERDŐS, GOODMAN, PÓSA 1966) *Die Kantenmenge eines jeden Graphen G der Ordnung $n \geq 2$ ohne isolierte Knoten kann in höchstens $n^2/4$ viele Kanten und Dreiecke partitioniert werden.*[1] *Insbesondere gilt also*

$$\theta_e(G) \;\leq\; \left\lfloor \frac{n^2}{4} \right\rfloor . \tag{12.3}$$

Beweis. Wir induzieren nach n. Der Induktionsanfang ($n = 2$) ist trivial. Beachte

$$\left\lfloor \frac{n^2}{4} \right\rfloor \;=\; \left\lfloor \frac{(n-1)^2}{4} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor .$$

Wir unterscheiden daher zwei Fälle.

Fall 1: $\exists x \in V : \quad d(x) \leq \lfloor \frac{n}{2} \rfloor$

Dann können die Kanten in $G - x$ nach Induktionsvoraussetzung mit $\left\lfloor \frac{(n-1)^2}{4} \right\rfloor$ vielen disjunkten Kanten und Dreiecken überdeckt werden. Es genügt also, die mit x inzidierenden Kanten schlicht in die Überdeckung von $G - x$ mit aufzunehmen.

Fall 2: $\forall v \in V : \quad d(v) > \lfloor \frac{n}{2} \rfloor$

Sei $x \in V$ ein Knoten minimalen Grades in G und

$$d(x) = \delta(G) = \left\lfloor \frac{n}{2} \right\rfloor + r$$

für ein $r \in \mathbb{N}$. Wir wollen analog zu Fall 1 vorgehen, außer daß nun auch Dreiecke verwendet werden müssen, um die mit x inzidierenden Kanten zu überdecken. Mit x inzidierende Dreiecke sind genau dann disjunkt, wenn die jeweils x nicht enthaltenden Kanten dieser Dreiecke ein Matching im Graphen $H := G[\Gamma(x)]$ bilden. Da $V \setminus \Gamma(x)$ aber höchstens

$$|V \setminus \Gamma(x)| \;\leq\; n \;-\; \left(\left\lfloor \frac{n}{2} \right\rfloor + r \right) \;\leq\; \left\lfloor \frac{n}{2} \right\rfloor - r + 1$$

viele Knoten enthält, hat jeder Knoten $y \in \Gamma(x)$ im Graphen H mindestens

$$d_H(y) \;\geq\; \delta(G) - |V \setminus \Gamma(x)| \;\geq\; \left\lfloor \frac{n}{2} \right\rfloor + r - \left(\left\lfloor \frac{n}{2} \right\rfloor - r + 1 \right) \;=\; 2r - 1$$

viele Nachbarn. Also gilt $|V(H)| \geq 2r$, und schon der Greedy-Algorithmus (vgl. Proposition 4.1.3) findet ein Matching M in H der Kardinalität mindestens r. Mithin kann man mindestens $2r$ der $\lfloor \frac{n}{2} \rfloor + r$ mit x inzidierenden Kanten durch r Dreiecke überdecken, die jeweils x und eine Matchingkante aus M enthalten. Die restlichen $\lfloor \frac{n}{2} \rfloor - r$ Kanten fügt man als solche zur Überdeckung hinzu. Dadurch sind die mit x inzidierenden Kanten insgesamt durch höchstens $\lfloor \frac{n}{2} \rfloor$ viele Kanten und Dreiecke überdeckt. Schließlich kann der verbleibende Graph $G[V - x] \setminus M$ nach Induktionsvoraussetzung wieder durch höchstens $\left\lfloor \frac{(n-1)^2}{4} \right\rfloor$ viele disjunkte Kanten und Dreiecke überdeckt werden. □

[1] für eine Verallgemeinerung siehe [Bol78a, Theorem VI.1.14]

Da die vollständig bipartiten Graphen $K_{r,r}$ bzw. $K_{r,r+1}$ gerade $\lfloor \frac{n^2}{4} \rfloor$ viele Kanten haben, ist die Schranke (12.3) wegen Lemma 12.1.4 scharf.[2] Andererseits ist natürlich jede Kante in einer maximalen Clique enthalten, so daß die Menge der maximalen Cliquen eines Graphen trivialerweise eine Kanten-Cliquenüberdeckung bildet. Folglich wird die Abschätzung (12.3) für Graphen mit wenigen maximalen Cliquen, wie z.B. den K_n oder die Klasse der chordalen Graphen (vgl. Proposition 7.2.15), beliebig schlecht. Es liegt daher nahe, zu untersuchen, wieviele Cliquen der Greedy-Algorithmus auswählt, der schlichtweg in jedem Schritt eine maximale Clique im Restgraphen konstruiert und deren Kanten aus dem Graphen entfernt, bis der Graph leer ist. Offenbar sind die derart konstruierten Cliquen disjunkt, d.h. es wird sogar eine Cliquenpartition von $E(G)$ gebildet. Jede Cliquenpartition der Kantenmenge eines Graphen G, die auf diese Weise zustande kommen kann, nennen wir eine Greedy-Cliquenzerlegung von $E(G)$.

Auch dieser Algorithmus partitioniert die Kantenmenge eines Graphen in höchstens $\lfloor \frac{n^2}{4} \rfloor$ viele Cliquen (was wie eben auch bestmöglich ist).

Satz 12.1.14 (McGuinness 1994) *Jede Greedy-Cliquenzerlegung der Kantenmenge E eines Graphen G enthält höchstens $n^2/4$ viele Cliquen.*[3]

Beweis durch Induktion nach n. Für $n = 1, 2$ ist die Aussage trivial. Sei also $\mathcal{C}$ eine Greedy-Cliquenzerlegung von $E(G)$, $n \geq 3$. Wenn jede Clique aus $\mathcal{C}$ mindestens 3 Knoten besitzt und also auch mindestens drei Kanten enthält, folgt sofort $|\mathcal{C}| \leq \frac{m}{3} \leq \frac{n(n-1)}{6} < n^2/4$. Wir nehmen also o.B.d.A. an, daß es eine Clique $Q_{uv} = \{u, v\} \in \mathcal{C}$ gibt, die aus nur einer einzigen Kante besteht. Aufgrund der Beziehung

$$\left\lfloor \frac{n^2}{4} \right\rfloor \;=\; \left\lfloor \frac{(n-2)^2}{4} \right\rfloor + (n-1) \tag{12.4}$$

ist hier die Idee, die Knoten u und v aus G zu entfernen. Da die Cliquen aus $\mathcal{C}$ die Kanten von G partitionieren, enthält keine Clique außer Q_{uv} beide Knoten u und v. Bezeichne $\mathcal{C}_{uv} \subseteq \mathcal{C} - Q_{uv}$ die Menge der Cliquen, die genau einen der Knoten u und v enthalten (und mindestens einen weiteren aus $V - u - v$). Wir behaupten, daß

$$|\mathcal{C}_{uv}| \;\leq\; n - 2. \tag{12.5}$$

Damit geht der Induktionsschluß wie folgt zu Ende. Seien F die Kanten in E, die von den Cliquen $\mathcal{C}_{uv} + Q$ überdeckt werden. Nach Definition von $\mathcal{C}_{uv}$ enthält F alle mit u oder v inzidierenden Kanten von G. $\mathcal{C} - \mathcal{C}_{uv} - Q_{uv}$ stellt aber eine Greedy-Cliquenzerlegung des Graphen $(V - u - v, E \setminus F)$ dar und enthält mithin nach Induktionsannahme höchstens $\left\lfloor \frac{(n-2)^2}{4} \right\rfloor$ viele Cliquen. Aus (12.5) und (12.4) folgt somit wie gewünscht $|\mathcal{C}| \leq \lfloor n^2/4 \rfloor$.

Beweis von (12.5). Hierzu geben wir eine injektive Abbildung

$$\phi : \quad \mathcal{C}_{uv} \;\to\; (\Gamma(u) \cup \Gamma(v)) - u - v$$

an. Sei $Q \in \mathcal{C}_{uv}$ eine Clique. Dann enthält Q also mindestens eine Kante aus dem Schnitt $\langle V - u - v, \{u, v\} \rangle$. Sei w ihr von u und v verschiedener Endknoten.

[2] Frieze und Reed [FR95] bewiesen jedoch, daß die Cliquenüberdeckungszahl eines zufälligen Graphen auf n Knoten (Kantenwahrscheinlichkeit $0 < p < 1$ fest) von der Größenordnung $\Theta(n^2/(\ln n)^2)$ ist, so daß für einen Graphen G „in der Regel“ $\theta_e(G) = o(n^2)$ gilt.

[3] Es gilt Gleichheit nur für die Graphen $K_{\lfloor n/2 \rfloor, \lceil n/2 \rceil}$.

Fall 1: $w \in \Gamma(u)\Delta\Gamma(v)$. Setze $\phi(Q) := w$.

Fall 2: $w \in \Gamma(u) \cap \Gamma(v)$. Q kann nicht beide Kanten $\{w,u\}$ und $\{w,v\}$ enthalten. Sei $Q_{wu} \in \mathcal{C}_{uv}$ diejenige Clique, die die Kante $\{w,u\}$ enthält und $Q_{wv} \in \mathcal{C}_{uv}$ diejenige, die $\{w,v\}$ enthält. O.B.d.A. trete Q_{wu} in der Greedy-Cliquenzerlegung vor Q_{wv} auf. Dann tritt Q_{wu} auch vor $Q_{uv} = \{u,v\}$ in der Greedy-Cliquenzerlegung auf, denn sonst wäre Q_{uv} nicht maximal gewählt. Demnach muß Q_{wu} einen Knoten w' enthalten, der nicht zu v benachbart ist, denn sonst wäre Q_{wu} nicht maximal gewählt. Setze in diesem Fall also

$$\phi(Q) := \begin{cases} w & \text{falls } Q = Q_{wv}, \\ w' & \text{falls } Q = Q_{wu}. \end{cases}$$

Die Injektivität von ϕ ist damit offensichtlich. □

Die letzten beiden Ergebnisse kontrastieren zum folgenden NP-Vollständigkeitsresultat.

Satz 12.1.15 (HOLYER 1981a) *Die folgenden Probleme sind* NP*-vollständig für einen Graphen $G = (V,E)$. Gibt es eine Partition von E in (i) Dreiecke (ii) maximale Cliquen? (iii) Partitioniere E in minimal viele Cliquen.* □

Anmerkungen und Übung

Es gibt einen polynomiellen Algorithmus, der die Kantenmenge eines Graphen so in Cliquen partitioniert, daß die Summe der Kardinalitäten der Cliquen höchstens $\lfloor n^2/2 \rfloor$ beträgt [Su95].

Wegen $\iota(G) \geq \theta(G) \geq \alpha(G)$ bilden die Graphenparameter $\theta(G)$ und $\alpha(G)$ untere Schranken für die Schnittzahl $\iota(G)$ eines Graphen G. Die folgende Übung zeigt, daß die beiden unteren Schranken gleichzeitig angenommen werden und daß für solche Graphen jede minimum Cliquen-Kantenüberdeckungen ausschließlich aus maximalen Cliquen besteht (trivialerweise gilt zudem: $\iota(G) = \mathcal{O}(n)$).

Übung 12.1.16 (CHOUDUM, PARTHASARATHY, RAVINDRA 1975) *Sei G ein Graph ohne isolierte Knoten. Dann sind äquivalent:*

(i) $\iota(G) = \alpha(G)$;

(ii) $\iota(G) = \theta(G)$;

(iii) Jede minimum Cliquen-Kantenüberdeckung $\mathcal{C}$ von G hat die Gestalt
$\mathcal{C} = \{\Gamma(v) + v | \ (v \in S \subseteq V) \wedge (|S| = \iota(G))\}$.

12.2 Erkennung von Linegraphen

Linegraphen sind nach Definition genau die Graphen G, die eine Schnittgraph-Darstellung $(S_v)_{v \in V}$ besitzen, in der alle S_v verschieden und zweielementig sind: man wähle $S_v = \{x,y\}$, falls v der Kante $\{x,y\}$ in dem Graphen entspricht, dessen Linegraph G ist.

Aus Satz 12.1.7 erhalten wir daher die folgende, äquivalente Charakterisierung derjenigen Graphen, welche Linegraphen anderer Graphen sind.

Satz 12.2.1 (KRAUSZ 1943) *Ein Graph ist Linegraph (eines Graphen) genau dann, wenn sich seine Kantenmenge so in Cliquen zerlegen läßt, daß jeder Knoten in höchstens zweien dieser Cliquen vorkommt.* □

Übung 12.2.2 (POLJAK, RÖDL, TURZÍK 1981) *Das Problem, zu entscheiden, ob für einen Graphen G und ein $k \in \mathbb{N}$ eine Cliquenüberdeckung $\{C_i\}_{i \in I}$ von G mit $|C_i| \leq k$ existiert, ist* NP*-vollständig.* [*Hinweis:* durch Reduktion von VERTEX CLIQUE PARTITION. Zu einer Eingabe G konstruiere einen Graphen H mit $\theta(G) = ccw(H) - n$, wobei $ccw(H)$ das minimale k, so daß H eine Cliquenüberdeckung $\{C_i\}_{i \in I}$ besitzt mit $|C_i| \leq k$.] □

Die Autoren zeigten ebenso, daß für einen Graphen G bereits die Frage nach der Existenz einer Schnittgraph-Darstellung mit verschiedenen dreielementigen Mengen S_v, $v \in V$, NP-vollständig ist. Wir wollen hier im Gegensatz dazu zeigen, daß Linegraphen in polynomieller Zeit erkannt werden können. Derartige Algorithmen wurden von LEHOT [Leh74] und VAN ROOIJ und WILF [RW65] angegeben. ROUSSOPOULOS [Rou73] gab den ersten linearen Algorithmus an.

12.3 Primal-dual Approximationsalgorithmen

In einer Reihe von herausragenden Arbeiten gelang es in den Jahren 1991 bis 1995, Approximationsalgorithmen für Probleme des SURVIVABLE NETWORK DESIGNS zu entwerfen. Hierbei geht es um die Konstruktion von ausfallsicheren Verbindungsnetzwerken, mit Anwendungen z.B. in der Telekommunikation. All diese Algorithmen verwenden die sogenannte primal-dual Methode, ein sehr mächtiges Paradigma für den Entwurf von Approximationalgorithmen für eine Vielzahl von Optimierungsproblemen – nachdem Approximationsalgorithmen bisher in der Regel speziell auf ein einziges Problem zugeschnitten waren und dessen problemspezifische Eigenschaften ausnutzten.

Wir entwickeln die wichtigsten Design-Regeln dieser Methode an einfachen Beispielen bis hin zur Approximation von proper 0-1-Funktionen und der Approximation von $\mathbb{N}$-wertigen proper Funktionen, wenn in der Lösung Mehrfachkanten erlaubt sind. Unsere Darstellung orientiert sich in starkem Maße an [GW96].

12.3.1 Der primal-dual Algorithmus der Linearen Optimierung

Bei einem *linearen Programm* ist zu einer $m \times n$-Matrix A, einem Zielfunktionsvektor $c \in \mathbb{Q}_+^n$ und einer sogenannten rechten Seite $b \in \mathbb{Q}^m$ ein Vektor $x \in \mathbb{Q}^n$ gesucht, der das lineare Funktional $c^T x$ unter folgenden linearen Nebenbedingungen minimiert:

$$\begin{aligned} & \min\ c^T x \\ s.t. \quad & Ax \ \geq\ b \\ & x \ \geq\ 0 \end{aligned} \tag{P}$$

(die Ungleichungen sind komponentenweise zu verstehen). Dieses Problem ist eng verknüpft mit dem folgenden:

$$\begin{aligned} & \max\ y^T b \\ s.t. \quad & y^T A \ \leq\ c \\ & y \ \geq\ 0. \end{aligned} \tag{D}$$

Das Minimierungsproblem heißt auch das *primale*, das Maximierungsproblem das *duale* Programm (bzw. umgekehrt). Wenn x eine zulässige Lösung von (P) (d.h. $x \in \mathbb{Q}^n$ erfüllt die Nebenbedingungen des primalen Programms) und y eine zulässige Lösung von (D) (d.h. $y \in \mathbb{Q}^m$ erfüllt die Nebenbedingungen des dualen Programms) ist, so gilt für die entsprechenden Zielfunktionswerte:

$$y^T b \ \leq\ y^T A x \ \leq\ c^T x. \tag{12.6}$$

Diese Beziehung nennt man auch die sogenannte *schwache Dualität*. Insbesondere ist also das Maximum von (D) höchstens so groß wie das Minimum von (P), falls beide endlich sind. Der starke Dualitätssatz besagt hingegen, daß die beiden Optima übereinstimmen, wenn (P) und (D) nur jeweils überhaupt zulässige Lösungen besitzen (siehe z.B. [Chv83]).

Ein Paar zulässiger Lösungen x und y von (P) bzw. (D) läßt sich anhand der sogenannten *primalen* und *dualen komplementären Schlupfbedingungen* als Optimallösungen von (P) bzw. (D) erkennen:

Satz 12.3.1 (komplementärer Schlupf) *Seien x und y eine primal bzw. dual zulässige Lösung von (P) bzw. (D). Dann sind x und y optimal für (P) bzw. (D) genau dann, wenn*

$$\begin{aligned} & (y^T A_{\cdot j} - c_j) \quad x_j \ = \ 0 \qquad \text{für } j = 1, \ldots, n, \\ \wedge\ & (A_{i\cdot}^T x - b_i) \quad y_i \ = \ 0 \qquad \text{für } i = 1, \ldots, m. \end{aligned}$$

Beweis. „$\Rightarrow$“ folgt aus dem starken Dualitätssatz, „$\Leftarrow$“ wegen Gleichheit in (12.6). □

Der primal-dual Algorithmus der linearen Programmierung nutzt diese Bedingungen, um Optimallösungen (gleichzeitig) von (P) und (D) zu berechnen. Er startet mit $x = 0$ und $y = 0$. Reduktion des gewichteten Problems auf ein ungewichtetes, das sich dann mit kombinatorischen Methoden lösen läst. Beispiel: maximum Matching in bipartiten Graphen.

12.3.2 Hitting Set, Node Cover und Dominating Set

Ein *Hypergraph* $\mathcal{H} = (U, \mathcal{F})$ besteht aus einer Menge $U = \{u_1, \ldots, u_n\}$, dem sogenannten Universum, und einer Menge $\mathcal{F} = \{F_1, \ldots, F_m\}$ von Teilmengen von U, den sogenannten Hyperkanten. Ein Graph ist also ein spezieller Hypergraph mit $\mathcal{F} \subseteq \binom{U}{2}$. Dem Node Cover-Problem auf Graphen entspricht auf Hypergraphen das Problem

Hitting Set:

INSTANZ: ein knotengewichteter Hypergraph $(U, \mathcal{F}, c)$, $c : U \to \mathbb{Q}_+$,
LÖSUNGEN: $A \subset U$, so daß $A \cap F \neq \emptyset$ für alle $F \in \mathcal{F}$,
ZIELFUNKTION: minimiere $c(A) := \sum_{u \in A} c_u$.

Es läßt sich wie folgt als ein ganzzahliges lineares Programm (IP) schreiben:

$$\begin{array}{lll} & \min\ c^T x & \\ s.t. & x(F)\ \geq\ 1 & F \in \mathcal{F} \\ & x_u \in \{0,1\} & u \in U, \end{array} \tag{IP}$$

wobei $x(F) := \sum_{u \in F} x_u$. Durch Relaxation der Ganzzahligkeitsbedingungen erhalten wir das folgende lineare Programm (LP):

$$\begin{array}{lll} & \min\ c^T x & \\ s.t. & x(F)\ \geq\ 1 & F \in \mathcal{F} \\ & x_u \geq 0 & u \in U. \end{array} \tag{LP}$$

Sein Dual (D) lautet:

$$\begin{array}{lll} & \max\ y^T 1 = \sum_{F \in \mathcal{F}} y_F & \\ s.t. & \sum_{F \ni u} y_F\ \leq\ c_u & u \in U \\ & y_F\ \geq\ 0 & F \in \mathcal{F}. \end{array} \tag{D}$$

Wir konstruieren nun mit der primal-dual Methode simultan eine (ganzzahlige) Lösung x für (LP) und eine Lösung y für (D). Aufgrund der NP-Vollständigkeit von HITTING SET werden wir allerdings nicht beide komplementären Schlupfbedingungen erfüllen können. Eine Optimallösung von (LP) wird schließlich i.a. nicht einmal ganzzahlig sein. Wir relaxieren daher die dualen komplementären Schlupfbedingungen, wir erzwingen aber die primalen komplementären Schupfbedingungen.

Regel 1. *Man starte mit der i.a. unzulässigen primalen Lösung $x \equiv 0$ und der dual zulässigen Lösung $y \equiv 0$ und bleibe stets dual zulässig.*

Regel 2. *Die primale Lösung x erfülle zu allen Zeiten die primalen komplementären Schlupfbedingungen. Erhöhe daher eine Koordinate x_u nur dann auf 1, wenn in der zugehörigen dualen Packungsbedingung $\sum_{F \ni u} y_F\ \leq\ c_u$ Gleichheit gilt.*

Dies führt uns zu folgendem Algorithmus für HITTING SET. Zu Beginn ist die Lösung $A := \{u \in U :\ x_u > 0\}$ die leere Menge. Solange nun A noch nicht zulässig ist, gibt es eine „verletzte Bedingung“ $A \cap F' = \emptyset$. Wir erhöhen die zu F' gehörige duale Variable $y_{F'}$ so weit wie möglich, ohne eine der Packungsbedingungen $\sum_{F \ni u} y_F \leq c_u$, $u \in F'$, zu verletzen. Dies ist genau dann gewährleistet, wenn der Zuwachs $\Delta y_{F'}$ von $y_{F'}$ die folgenden Bedingungen erfüllt:

$$\sum_{F \ni u} y_F + \Delta y_{F'}\ \leq\ c_u \qquad \forall u \in F'.$$

Mithin berechnet sich der maximale Zuwachs $\Delta y_{F'}$ nach

$$\Delta y_{F'} \;=\; \min_{u \in F'} \{c_u - \sum_{F \ni u} y_F\}.$$

Weiterhin gibt es nun (mindestens) ein $u \in F'$, wo das Minimum auf der rechten Seite angenommen wird. Wegen $A \cap F' = \emptyset$ ist $u \notin A$, und wir können ein solches u unter Beachtung von Regel 2 zu A hinzufügen, und damit die Zulässigkeit von A schrittweise verbessern. Unser Algorithmus sieht damit wie folgt aus:

```
FUNCTION HITTING_SET (F, c) : 2^U;

BEGIN
  A := ∅;
  FOR F ∈ F DO y_F := 0;
  WHILE ∃F' ∈ F : A ∩ F' = ∅ DO BEGIN
    {erhöhe y_F' bis Σ_{F∋u} y_F = c_u für ein u ∈ F':}
    y_F' := y_F' + min {c_u − Σ_{F∋u} y_F : u ∈ F'};
    A := A + u;
  END;
  HITTING_SET:= A;
END;
```

Beachte, daß für ein $u \in A$ die komplementäre Schlupfbedingung $\sum_{F \ni u} y_F = c_u$ ab dem Augenblick, in dem wir u in A aufnehmen, für den gesamten Rest des Algorithmus gültig bleibt, weil für jedes F, das u enthält, $F \cap A \neq \emptyset$ und daher die zugehörige duale Variable y_F nicht mehr erhöht wird.

Nach Konstruktion ist die zurückgegebene Lösung A zulässig. Da höchstens n Elemente in A aufgenommen werden, wird die `WHILE`-Schleife $\mathcal{O}(n)$-mal durchlaufen. Beachte jedoch, daß $\mathcal{F}$ exponentiell in n viele Hyperkanten enthalten kann. Wie wir später sehen werden, kann der Test auf Zulässigkeit von A und ggf. das Finden einer von A noch nicht getroffenen Hyperkante $F' \in \mathcal{F}$ im Kopf der `WHILE`-Schleife in vielen interessanten Anwendungen in (in n) polynomieller Zeit ausgeführt werden. Überlegen wir uns daher, wie auch die Minimumbildung in (in n) polynomieller Zeit vonstatten gehen kann. Man ersieht, daß die dualen Variablen y_F stets nur in der Form $\sum_{F \ni u} y_F$, also summiert an einem bestimmten $u \in U$ benötigt werden. Es liegt also nahe, die Summen

$$d(u) := \sum_{F \ni u} y_F, \qquad u \in U,$$

fortzuschreiben.

Regel 3. *Verwalte die nicht-negativen Komponenten von y implizit.*

Dies führt zu folgendem modifizierten Algorithmus.

```
FUNCTION EFFICIENT_HITTING_SET (F, c) : 2^U;

BEGIN
  A := ∅;
  {setze implizit y_F = 0 für alle F ∈ F:}
  FOR u ∈ U DO d(u) := 0;
  WHILE ∃F' ∈ F : A ∩ F' = ∅ DO BEGIN
```

```
    ε := min {c_u − d(u) : u ∈ F'};
    A := A + argmin {c_u − d(u) : u ∈ F'};
    {erhöhe implizit y_F' um ε:}
    FOR u ∈ F' DO  d(u) := d(u) + ε;
  END;
  EFFICIENT_HITTING_SET:= A;
END;
```

Satz 12.3.2 (BAR-YEHUDA, EVEN 1981) EFFICIENT_HITTING_SET *ist ein* $\mathcal{O}(nm)$*-Approximationsalgorithmus für* HITTING SET *mit Güteratio* $\max\{|F| : F \in \mathcal{F}\}$.

Beweis. Da wir die primalen komplementären Schlupfbedingungen erzwangen, können wir die Kosten der Lösung A wie folgt schreiben:

$$c(A) \;=\; \sum_{u \in A} c_u \;=\; \sum_{u \in A} \sum_{F \ni u} y_F \;=\; \sum_{F \in \mathcal{F}} |A \cap F| y_F.$$

Da andererseits y eine dual zulässige Lösung von (D) ist, nach dem schwachen Dualitätssatz der Optimalwert Z_D^* des dualen Programms höchstens so groß wie der Optimalwert Z_{LP}^* des primalen Programms (LP) ist, und (LP) schließlich eine Relaxierung von (IP) ist, gilt:

$$\sum_{F \in \mathcal{F}} y_F \;\leq\; Z_D^* \;\leq\; Z_{LP}^* \;\leq\; Z_{IP}^*.$$

Es folgt also

$$c(A) \;\leq\; \max\{|F \cap A| : F \in \mathcal{F}\} \cdot Z_{IP}^*. \qquad \square$$

Anwendungen. Das knotengewichtete Knotenüberdeckungsproblem WEIGHTED NODE COVER läßt sich als ein HITTING SET Problem auffassen, bei dem zu einem Graph $G = (V, E, c)$ mit Knotengewichten $c : V \to \mathbb{Q}_+$ als Eingabe eine Knotenmenge $C \subseteq V$ minimalen Gewichts $c(C) := \sum_{v \in C} c(v)$ gesucht ist, die alle Kanten $\{u, v\} \in E$ trifft. Hier gilt offenbar $\max\{|F| : F \in \mathcal{F}\} = 2$, womit obiger Satz das folgende Korollar nach sich zieht.

Korollar 12.3.3 EFFICIENT_HITTING_SET *ist ein 2-Approximationsalgorithmus für* WEIGHTED NODE COVER *und kann mit linearer Laufzeit implementiert werden.*

Es ist sehr instruktiv, den Algorithmus GREEDY_POINT_COVER wie auch den Beweis seiner Güteratio noch einmal speziell für das Problem WEIGHTED NODE COVER aufzuschreiben (Übung). Man vergleiche diesen Approximationsalgorithmus für WEIGHTED NODE COVER mit den Formulierungen in [GP86, Gon95], siehe auch [Cla83].

Das knotengewichtete DOMINATING SET Problem läßt sich als ein HITTING SET Problem modellieren, bei dem eine Knotenmenge $D \subseteq V$ minimalen Gewichts gesucht ist, die alle Mengen der Form $\Gamma(v) + v$, $v \in V$, trifft. Aus Satz 12.3.2 folgt unmittelbar das

Korollar 12.3.4 *Sei* $\Delta \in \mathbb{N}$ *eine natürliche Zahl und* $\mathcal{G}$ *eine Graphenklasse mit* $\Delta(G) \leq \Delta$ *für alle* $G \in \mathcal{G}$*. Dann ist* EFFICIENT_HITTING_SET *ein* $(\Delta+1)$*-Approximationsalgorithmus für* WEIGHTED DOMINATING SET *auf* $\mathcal{G}$ *und kann mit linearer Laufzeit implementiert werden.*

Nicht-Approximierbarkeit. Auf der negativen Seite wissen wir, daß NODE COVER MAX-SNP-vollständig ist. PAPADIMITRIOU und YANNAKAKIS [PY91] konnten zeigen, daß auch DOMINATING SET bei beschränkten Knotengraden MAX-SNP-vollständig ist. Aus Satz 11.1.3 schließen wir, daß, während es für beide Probleme einen Approximationsalgorithmus mit konstanter Güteratio gibt, keines von beiden ein PAS besitzt (außer P=NP).

Übungen

Sei $F = \bigwedge_{j=1}^{m} C_j$ eine Instanz von 2-SAT, d.h. eine Boolesche Formel in CNF über den Variablen $\{x_1, \ldots, x_n\}$, wobei jede Klausel C_j die Konjunktion von jeweils nur höchstens zwei Literalen ist. $c \in \mathbb{Q}_+^n$ sei eine Gewichtung der Booleschen Variablen. Beim Optimierungsproblem MINIMUM (bzw. MAXIMUM) COST 2-SAT ist für eine erfüllbare 2-SAT-Formel F eine erfüllende Wahrheitswertebelegung $x \in \{0,1\}^n$ gesucht, die $\sum_{i=1}^{n} c_i x_i$ minimiert (bzw. maximiert).

Übung 12.3.5 a) MINIMUM COST 2-SAT *und* MAXIMUM COST 2-SAT *sind* NP*-vollständig.*
[*Hinweis:* formuliere die Probleme NODE COVER und INDEPENDENT SET entsprechend]
b) *Es gibt einen 2-Approximationsalgorithmus für* MINIMUM COST 2-SAT [GP92].
[*Hinweis:* schreibe das Problem als ganzzahliges lineares Programm und runde eine fraktionale Lösung der LP-Relaxation. Nur ein Typ von Klauseln bereitet Schwierigkeiten – verwende Satz 1.4.8]

Beim Problem MIN SAT ist eine Wahrheiswertebelegung der Booleschen Variablen einer Booleschen Formel in CNF gesucht, die die Anzahl der erfüllten Klauseln minimiert. Wie die nächste Übung zeigt, kann MIN SAT genauso gut approximiert werden wie NODE COVER.

Übung 12.3.6 [MR96] **a)** *Eine* MIN SAT *Formel besitzt genau dann eine Wertebelegung, die keine ihrer Klauseln erfüllt, wenn jede Variable x_i, $1 \leq i \leq n$, ausschließlich in positiver oder ausschließlich in negierter Form auftritt.*
b) *Zu einer* MIN SAT *Formel F konstruiere einen Hilfgraphen $H(F)$ auf der Menge $\mathcal{C}$ der Klauseln von F wie folgt: zwei Klauseln C_1 und C_2 sind in $H(F)$ durch eine Kante verbunden genau dann, wenn es eine Variable gibt, die in einer der beiden Klauseln positiv und in der anderen negiert auftritt. Es bezeichne $OPT(F)$ die minimale Anzahl von Klauseln, die von einer Wertebelegung für F erfüllt werden. Dann gilt $OPT(F) = \tau(H(F))$.*
c) *Wenn* APPROX_NC *ein Approximationsalgorithmus für* NODE COVER *mit Güteratio r ist, so ist*

```
PROCEDURE MR_MIN_SAT (F);

BEGIN
  konstruiere den Hilfgraphen H(F);
  U := APPROX_NC(H(F));
  konstruiere eine Wertebelegung für F,
        die keine der Klauseln aus C \ U erfüllt;
END;
```

ein Approximationsalgorithmus für MIN SAT *mit Güteratio r.*
d) *Wenn es einen Approximationsalgorithmus mit Güteratio r für* MIN SAT *gibt, dann gibt es auch einen Approximationsalgorithmus mit Güteratio r für* NODE COVER. [*Hinweis:* zu einem gegebenen Graphen G konstruiere eine Boolesche Formel F, so daß $H(F) = G$.]

12.3.3 SHORTEST PATH

Wir kommen nun zu Problemen, wo das Universum U die Menge der Kanten eines Graphen G ist (es ist also insbesondere $|U| = m$) und die Hyperkanten Schnitte in G sind. Lösungen sind nun also Kantenmengen, die alle vorgegebenen Schnitte treffen. Da jeder Schnitt in

$G = (V, E)$ als $\langle S, V \setminus S\rangle$ dargestellt werden kann für eine Knotenmenge $\emptyset \neq S \subset V$, enthalten solche HITTING SET Probleme also nur höchstens $2^{n-1} - 1$ (statt 2^m) viele Schnittbedingungen.

Beispielsweise läßt sich das Problem SHORTEST PATH, zu gegebenen Knoten s und t in $G = (V, E, c), c : E \to \mathbb{Q}^+$, einen (bezüglich c) kürzesten $s-t-$Pfad in G zu finden, als derartiges HITTING SET-Problem auffassen. Mit den Bezeichnungen

$$\delta(S) := \{\{u, v\} \in E |\ u \in S \wedge v \notin S\} \ = \ \langle S, V \setminus S\rangle$$

für den Co-Rand einer Knotenmenge $\emptyset \neq S \subset V$ und $x(F) := \sum_{e \in F} x_e$ für eine Kantenmenge $F \subseteq E$ ist SHORTEST PATH äquivalent zu dem ganzzahligen Optimierungsproblem:

$$\begin{array}{lll} & \min\ c^T x & \\ s.t. & x(\delta(S)) \ \geq \ f(S) & \emptyset \neq S \subset V \\ & x_e \in \{0, 1\} & e \in E, \end{array} \qquad \text{(IP)}$$

wobei die Funktion $f : 2^V \to \{0, 1\}$ definiert ist durch:

$$f(S) \ := \quad \begin{cases} 1 & \text{falls } |S \cap \{s, t\}| = 1, \\ 0 & \text{sonst.} \end{cases}$$

f ist also gerade die Inzidenzfunktion aller Schnitte $\delta(S) = \langle S, V \setminus S\rangle$, die von einer Lösung $A := \{e \in E :\ x_e = 1\}$ von (IP) zu treffen sind. Offenbar ist jede inklusionsweise minimale Lösung A von (IP) ein Wald in G, weil ein Kreis jeden Schnitt in einer geraden Anzahl von Kanten schneidet, so daß bedenkenlos eine beliebige Kante aus dem Kreis entfernt werden könnte. Wir indentifizieren im folgenden wechselseitig eine (nicht notwendig zulässige) Lösung $x : E \to \{0, 1\}$ von (IP) mit der Menge $A \subseteq E$, deren Inzidenzfunktion x ist. Wir identifizieren weiter Mengen $\emptyset \neq S \subset V$ mit den entsprechenden Schnitten in G und sagen, eine Menge S sei verletzt, falls die aktuelle Lösung $A \subseteq E$ den Schnitt (die Hyperkante) $\delta(S) = \langle S, V \setminus S\rangle$ nicht trifft.

Man beachte, daß es in der Formulierung (IP) für jeden Schnitt $\delta(S) = \langle S, V \setminus S\rangle$ in G mit $f(S) > 0$ zwei identische Ungleichungen gibt – eine für S und eine für $V \setminus S$. Die zugehörigen dualen Variablen seien y_S und $y_{V \setminus S}$. Der Wert $y_{\delta(S)}$ ergibt sich damit aus $y_{\delta(S)} = y_S + y_{V \setminus S}$. Das Dual der LP-Relaxation von (IP) lautet mithin

$$\begin{array}{lll} & \max \sum_{\emptyset \neq S \subset V} f(S)\, y_S & \\ s.t. & \sum_{\delta(S) \ni e} y_S \ \leq \ c_e & e \in E \\ & y_S \ \geq \ 0 & \emptyset \neq S \subset V. \end{array} \qquad \text{(D)}$$

Die primal-dual Methode konstruiert nun, beginnend mit der leeren Menge $A := \emptyset$, einen Wald A in G, der einen $s-t-$Pfad enthält. Doch statt einer beliebigen, wird nun in jedem Schritt eine inklusionsweise minimale verletzende Menge S gewählt und eine Kante

aus dem Schnitt $\langle S, V \setminus S\rangle$ zu A hinzugefügt. Mit der Setzung $C_s := \{s\}$ und $C_t := \{t\}$ gibt es zu Beginn also nur die beiden verletzten Mengen C_s und C_t. Der primal-dual Approximationsalgorithmus wähle nun beispielsweise die verletzte Menge C_s aus. Nun wird y_{C_s} erhöht, bis $y_{C_s} = c_{e_0}$ für eine Kante $e_0 \in \delta(C_s)$, d.h. eine mit s inzidierende Kante. Eine solche Kante $e_0 = \{s, v\}$ wird sodann in A aufgenommen; wir setzen $C_s := C_s + v$ und $d(e) := d(e) + c_{e_0}$ für alle mit s inzidierenden Kanten. Wenn man die Kosten c_e als Entfernung zwischen den Endknoten der Kante e interpretiert, so kann man sich vorstellen, daß wir nun auf den Kanten $e \in \delta(C_s)$ bereits einen Abschnitt der Länge $d(e)$ in Richtung t gegangen sind. In den nächstfolgenden Schritten wird entsprechend jeweils eine Kante $e \in \delta(C_s)$ (bzw. $e \in \delta(C_t)$) zu A hinzugefügt, für die die Länge $c_e - d(e)$ des noch abzuschreitenden Weges auf e kürzestmöglich unter allen Kanten in $\delta(C_s)$ (bzw. $\delta(C_t)$) ist. Es gibt somit zu allen Zeiten genau zwei minimale verletzte Schnitte (nämlich C_s und C_t), und das Verfahren endet, sobald eine Kante C_s und C_t vereinigt. Der Leser wird den Algorithmus von DIJKSTRA wiedererkennen – mit der Modifikation, daß sich der Algorithmus hier gewissermaßen symmetrisch von beiden Seiten (s und t) gleichzeitig auf einander zu bewegt, statt sich von s nach t vorzuarbeiten. Diese Variante, auch BIDIREKTIONALE SUCHE genannt, geht auf NICHOLSON [Nic66] zurück.

Zu beachten ist, daß am Schluß, um wirklich einen $s-t$–Pfad auszugeben, noch redundante Kanten aus A entfernt werden müssen. In diesem Fall ist die inklusionsweise minimale zulässige Teillösung A' von A eindeutig bestimmt. Man beachte ferner, daß für jede Kante $e \in A$ unabhängig von den restlichen Kanten aus A entschieden werden kann, ob $e \in A'$: es ist $e \in A'$ genau dann, wenn $A - e$ noch zulässig ist (d.h. den $s-t$–Pfad enthält).

Da y_S nur positiv ist für Mengen $\emptyset \neq S \subset V$, die im Verlauf des Algorithmus als C_s oder C_t auftraten, und der schließlich ausgegebene $s-t$–Pfad A' diese Mengen genau einmal verläßt, erfüllt A' auch die dualen komplementären Schlupfbedingungen. Aus dem Satz über den komplementären Schlupf 12.3.1 folgt, daß A' eine Optimallösung von (IP) darstellt. Wir erhalten also einen alternativen Beweis der Korrektheit des Algorithmus von DIJKSTRA.

Halten wir schließlich die gefundenen zusätzlichen Regeln für den Entwurf von primal-dual Approximationsalgorithmen fest:

Regel 4. *Erhöhe die Zulässigkeit einer Lösung nur an (inklusionsweise) minimal verletzten Mengen.*

Regel 5. *Berechne am Schluß aus der konstruierten zulässigen Lösung $A \subseteq E$ eine inklusionsweise minimale zulässige Teillösung $A' \subseteq A$.*

12.3.4 Das GENERALIZED STEINER TREE Problem

Wir betrachten in diesem Abschnitt das folgende ganzzahlige Optimierungsproblem.

$$\begin{aligned} & \min\ c^T x & & \\ s.t. \quad & x(\delta(S)) \geq f(S) & \emptyset \neq S \subset V & \qquad \text{(IP)} \\ & x_e \in \{0,1\} & e \in E & \end{aligned}$$

für eine Funktion $f : 2^V \to \{0,1\}$. f ist also gerade die Inzidenzfunktion aller Schnitte $\delta(S) = \langle S, V \setminus S\rangle$, die von einer Lösung $A := \{e \in E : x_e = 1\}$ von (IP) zu treffen sind. Man beachte, daß es in der Formulierung (IP) für jeden Schnitt $\delta(S) = \langle S, V \setminus S\rangle$ zwei identische Ungleichungen gibt – eine für S und eine für $V \setminus S$. Die zugehörigen dualen Variablen sind y_S und $y_{V\setminus S}$. Der Wert $y_{\delta(S)}$ ergibt sich aus $y_{\delta(S)} = y_S + y_{V\setminus S}$. Das Dual der LP-Relaxation von (IP) lautet damit

$$\begin{array}{lll} & \max \sum_{\emptyset \neq S \subset V} f(S)\, y_S & \\ s.t. & \sum_{\delta(S) \ni e} y_S \;\leq\; c_e & e \in E \\ & y_S \;\geq\; 0 & \emptyset \neq S \subset V. \end{array} \tag{D}$$

Eine Knotenmenge $S \subset V$, $f(S) > 0$, heißt *verletzt*, falls $A \cap \delta(S) = \emptyset$ (die Lösung A verletzt also die Bedingung $x(\delta(S)) \geq f(S)$ und ist daher nicht zulässig).

Da es für allgemeine 0-1-Funktionen f schwer sein kann, die Zulässigkeit einer Lösung $A \subseteq E$ zu testen bzw. einen verletzten Schnitt $\delta(S)$ zu finden (vgl. z.B. die Funktion f, die nur auf einem einzigen Schnitt $(S_0, V \setminus S_0)$ ungleich Null ist), beschränken wir uns auf die Untersuchung von sogenannten proper Funktionen.

Definition 12.3.7 *Eine Funktion $f : 2^V \to \mathbb{N}$ heißt* proper, *falls*

(P1) $f(V) = 0$;

(P2) [Symmetrie] $f(S) = f(V \setminus S)$ *für alle* $\emptyset \neq S \subset V$;

(P3) [Maximalität] $\max\{f(A), f(B)\} \geq f(A \cup B)$ *für alle* $A, B \subset V$ *mit* $A \cap B = \emptyset$.

Bedingung (P3) ist für eine 0-1-Funktion f offenbar äquivalent zu

$$f(A \cup B) = 1 \;\Rightarrow\; (f(A) = 1) \vee (f(B) = 1) \quad \forall\, A, B \subseteq V :\; A \cap B = \emptyset. \tag{12.7}$$

Die Bezeichnung Generalized Steiner Tree für das Problem (IP) mit proper 0-1-Funktion f rührt daher, daß es insbesondere das Problem Steiner Tree (und mithin auch Minimum Spanning Tree oder Shortest Path) verallgemeinert. Zu einer Terminalmenge $\emptyset \neq T \subseteq V$ sind beim Problem Steiner Tree die Schnitte $\{\delta(S) : \emptyset \neq S \cap T \neq T\}$ von einer Lösung $A \subseteq E$ zu treffen. Die zugehörige Inzidenzfunktion f ist offensichtlich proper. Goemans und Williamson [GW95] bezeichnen Probleme der Form (IP) mit proper 0-1-Funktion f auch als „constrained forest problems“, da alle (inklusionsweise) minimalen Lösungen von (IP) offenbar Wälder sind. Beispielsweise kann man beim Generalized Steiner Tree Problem nach einem Wald $A \subseteq E$ minimalen Gewichts fragen, so daß die Knoten von k Terminalmengen $S_1, \ldots, S_k$ in (V, A) jeweils untereinander verbunden sind.

Das folgende Lemma zeigt, daß die Bedingung der Maximalität (P3) eine effiziente Charakterisierung der (inklusionsweise) minimal verletzten Mengen ermöglicht. Inklusionsweise minimal verletzte Mengen nennen wir im folgenden kurz *aktive Mengen*, da wir die aktuelle Lösung A noch um mindestens eine Kante aus ihrem Schnitt erweitern müssen.

Lemma 12.3.8 *Es erfülle $f : 2^V \to \{0,1\}$ die Maximalitätsbedingung (P3), und sei $A \subseteq E$ eine Lösung von (IP). Dann gilt:*

(i) Die aktiven Mengen bezüglich A sind Zusammenhangskomponenten von (V, A);

(ii) A ist zulässig genau dann, wenn $f(N) = 0$ für alle Zusammenhangskomponenten N von (V, A).

Beweis. Sei $S \subset V$ eine verletzte Menge, d.h. es gelte $f(S) = 1$, aber $A \cap \delta(S) = \emptyset$. Dann gilt für jede Zusammenhangskomponente $C \subseteq V$ von (V, A) offenbar: entweder enthält S die Zusammenhangskomponente C ganz oder gar nicht. S ist also die Vereinigung von Zusammenhangskomponenten von (V, A). Nach Bedingung (P3) gilt $f(C) = 1$ für mindestens eine Zusammenhangskomponente $C \subseteq S$ von (V, A). Wegen $\delta(C) \cap A = \emptyset$ ist also auch die Menge C verletzt. □

Die Menge der minimalen verletzten Mengen läßt sich daher effizient berechnen: man hält sich hierzu lediglich stets die Menge $\mathcal{C}$ der Komponenten von (V, A) zur Verfügung bereit. Zu Beginn ist dies, da wir mit $y \equiv 0$ starten, die Menge $\mathcal{C} := \{\{v\} : v \in V\}$, bei Aufnahme einer Kante $\{u, v\}$ in die Lösung werden die beiden (verschiedenen) Komponenten, in denen u bzw. v liegen, verschmolzen.

Da es in jedem Schritt i.a. mehrere aktiven Mengen gibt, muß der primal-dual Approximationsalgorithmus nun die Lösung y des dualen Problems in jedem Schritt auf allen aktiven Komponenten gleichzeitig und gleichmäßig erhöhen (bis eine Packungsbedingung scharf wird).

Regel 6. *Erhöhe die dualen Variablen in jedem Schritt gleichmäßig auf allen aktiven Mengen.*

Die dualen Variablen werden implizit gehalten durch

$$\sum_{\delta(S) \ni e} y_{\delta(S)} \;=\; \sum_{\delta(S) \ni e} y_S \;=\; d(i) + d(j) \tag{12.8}$$

für eine Kante $e = \{i, j\}$, wobei

$$d(v) := \sum_{S \ni v} y_S.$$

Zum besseren Verständnis berechnen wir den Wert $LB := \sum_{\emptyset \neq S \subset V} f(S)\; y_S$ der dual zulässigen Lösung y mit. Der primal-dual Approximationsalgorithmus für GENERALIZED STEINER TREE hat damit folgende Gestalt:

```
FUNCTION APPROX_PROPER_0-1 (G, c, f) : 2^E;

BEGIN
  A := ∅;                                          {primale Startlösung}
  {setze implizit y_S = 0 für alle ∅ ≠ S ⊂ V :}
  FOR v ∈ V DO  d(v) := 0;
  LB := 0;
  C := {{v} : v ∈ V};                {Zusammenhangskomponenten von (V, A)}
  WHILE ∃C ∈ C : f(C) = 1 DO BEGIN
    finde eine Kante e = {i, j}, i ∈ C_i ≠ C_j ∋ j, die
        ε := (c_e − d(i) − d(j)) / (f(C_i) + f(C_j)) minimiert;
    A := A + e;
    {erhöhe y gleichmäßig auf allen aktiven Komponenten:}
```

```
    FOR C ∈ 𝒞 : f(C) = 1 DO
      {setze implizit y_C := y_C + ε}
      FOR v ∈ C DO  d(v) := d(v) + ε;
    LB := LB + ε · Σ_{C∈𝒞} f(C);
    𝒞 := 𝒞 − C_i − C_j + (C_i ∪ C_j);
  END;
  A' := {e ∈ A : f(N) = 1 für eine Komponente N von (V, A − e)};
  Approx_Proper_0-1:= A';
END;
```

Wir untersuchen im folgenden nacheinander Korrektheit, Approximationsgüte und Laufzeit von Approx_Proper_0-1.

Korrektheit. Es ist unmittelbar einsichtig, daß wir Regel 6 einhalten, d.h. daß wir die dualen Variablen in jedem Schritt gleichmäßig auf allen aktiven Mengen erhöhen. Da die WHILE-Schleife nur betreten wird, wenn es noch aktive Komponenten gibt, in das Minimum ϵ endlich. Insbesondere verbindet e stets eine aktive Komponente mit einer anderen (nicht notwendig aktiven), so daß die Anzahl aktiver Komponenten schwach monoton fällt. Ferner folgt, daß die Menge A stets einen Wald auf der Knotenmenge V darstellt.

Weiter gilt offensichtlich in der Tat zu jedem Zeitpunkt $d(v) := \sum_{S \ni v} y_S$. Mithin gilt für jede Kante $e = \{i, j\}$, solange i und j in verschiedenen Komponenten von (V, A) liegen, Gleichung (12.8). Damit kann die duale Lösung y genau dann gleichmäßig auf allen aktiven Komponenten um einen Wert $\epsilon > 0$ erhöhen werden, ohne die duale Zulässigkeit, d.h. die Packungsbedingungen auf Kanten zwischen betroffenen Komponenten, zu verletzen, wenn gilt:

$$d(i) + d(j) + \epsilon f(C_i) + \epsilon f(C_j) \;\leq\; c_e \qquad \forall e = \{i, j\} : \; i \in C_i \neq C_j \ni j.$$

Somit wird ϵ vom Algorithmus Approx_Proper_0-1 korrekt berechnet und Regel 1 (duale Zulässigkeit) eingehalten. Durch die Wahl von e wird auch Regel 2 (primale komplementäre Schlupfbedingung) eingehalten: immer, wenn eine Kante e zu A hinzugefügt wird, ist die duale Packungsbedingung zu e mit Gleichheit erfüllt. Da ihre Endknoten i und j damit in dieselbe Komponente gelangen, wird $\sum_{\delta(S) \ni e} y_S$ nie mehr erhöht, d.h. die Gleichheit in der Packungsbedingung bleibt erhalten.

Da ein Wald auf V höchstens $n - 1$ Kanten besitzen kann, wird die WHILE-Schleife $\mathcal{O}(n)$-mal durchlaufen und, wenn sie verlassen wird, ist A eine primal zulässige Lösung.

Es bleibt, zu zeigen, daß der abschließende Verbesserungsschritt eine (inklusionsweise) minimale, primal zulässige Lösung A' liefert. Dies ergibt sich wie folgt aus den Eigenschaften einer proper Funktion.

Lemma 12.3.9 (Komplementarität) *Sei $f : 2^V \to \{0, 1\}$ eine proper Funktion. Seien $N \subseteq S \subseteq V$ Knotenmengen mit $f(S) = 0$ und $f(S \setminus N) = 0$. Dann gilt auch $f(N) = 0$.*

Beweis. Angenommen, es gälte $f(N) = 1$. Dann ist aufgrund der Symmetrie von f ebenso $1 = f(N) = f(V \setminus N) = f((V \setminus S) \dot\cup (S \setminus N))$. Aus der Maximalität von f folgte $f(V \setminus S) = f(S) = 1$ oder $f(S \setminus N) = 1$, Widerspruch. □

Somit gilt für proper 0-1-Funktionen ähnlich wie beim Shortest Path Problem:

Lemma 12.3.10 *Sei $f : 2^V \to \{0, 1\}$ eine proper Funktion und $A \subseteq E$ eine zulässige Lösung zu (IP). Dann gibt es eine eindeutige inklusionsminimale zulässige Lösung $A' \subseteq A$,*

und es gilt

$$A' \;=\; \{e \in A :\; f(N) = 1 \textit{ für eine Zusammenhangskomponente } N \textit{ von } (V, A-e)\}.$$

Beweis. Angenommen $B \subseteq A$ ist primal zulässig und für eine Kante $e \in B$ ist $B - e$ noch immer zulässig. Dann ist offenbar auch $A - e$ zulässig. Mithin ist jede zulässige Lösung $B \subseteq A$ Obermenge von A', und es genügt, die Zulässigkeit von A' zu zeigen. Sei $N \subset V$ eine Zusammenhangskomponente von (V, A'). Nach Lemma 12.3.8 ist $f(N) = 0$ zu zeigen. Sei $C \subseteq V$ die Komponente von (V, A), so daß $N \subseteq C$. Sei $\delta(N) \cap N = \{e_1, \ldots, e_k\}$ und seien N_i und $C \setminus N_i$ die Komponenten von $C - e_i$, so daß $N \subset C \setminus N_i$. Da A einen Baum auf C aufspannt, gilt $C = N \cup N_1 \cup \ldots \cup N_k$. Wegen $e_i \in A \setminus A'$ gilt $f(N_i) = 0$ für alle $1 \leq i \leq k$, und aus der Maximalität von f folgt $f(N_1 \cup \ldots \cup N_k) = 0$. Da aufgrund der Zulässigkeit von A zudem $f(C) = 0$, liefert obiges Lemma schließlich $f(N) = 0$. □

Satz 12.3.11 (Agrawal, Klein, Ravi 1995; Goemans, Williamson 1995)
Approx_Proper_0-1 *ist ein Approximationsalgorithmus für* Generalized Steiner Tree *mit Gütegarantie* $2 - 2/\ell$, *wobei* $\ell := |\{v \in V :\; f(\{v\}) = 1\}|$.

Beweis. Da das von Approx_Proper_0-1 konstruierte y dual zulässig ist (Regel 1) und $y_S > 0$ nur, wenn $f(S) > 0$, gilt

$$LB \;=\; \sum_{\emptyset \neq S \subset V} y_S \;=\; \sum_{\emptyset \neq S \subset V} f(S)\, y_S \;\leq\; Z_D^* \;=\; Z_{LP}^* \;\leq\; Z_{IP}^*, \tag{12.9}$$

wobei Z_{LP}^* bzw. Z_D^* den Optimalwert der LP-Relaxation von (IP) bzw. dessen Duals bezeichnen. Da x die primalen komplementären Schlupfbedingungen erfüllt (Regel 2), berechnen sich die Kosten der konstruierten primalen Lösung zu

$$c(A') \;=\; \sum_{e \in A'} c_e \;=\; \sum_{e \in A'} \sum_{\delta(S) \ni e} y_S \;=\; \sum_{\emptyset \neq S \subset V} y_S \cdot |A' \cap \delta(S)|. \tag{12.10}$$

Wir zeigen nun durch Induktion nach der Anzahl Iterationen der `WHILE`-Schleife, daß

$$\sum_{\emptyset \neq S \subset V} y_S \cdot |A' \cap \delta(S)| \;\leq\; (2 - 2/\ell) \cdot \sum_{\emptyset \neq S \subset V} y_S, \tag{12.11}$$

woraus zusammen mit (12.9) und (12.10) die Behauptung des Satzes folgt. Gleichung (12.11) ist nach der Initialisierungsphase zu Beginn des Algorithmus mit Gleichheit erfüllt. Es genügt daher, zu zeigen, daß in jedem Durchlauf der `WHILE`-Schleife die Zuwächse auf beiden Seiten von (12.11)

$$\begin{array}{rrcl} & \sum_{C \in \mathcal{C}_a} \epsilon |A' \cap \delta(C)| & \leq & (2 - 2/\ell)|\mathcal{C}_a|\epsilon \\ \Leftrightarrow & \sum_{C \in \mathcal{C}_a} |A' \cap \delta(C)| & \leq & (2 - 2/\ell)|\mathcal{C}_a| \end{array} \tag{12.12}$$

erfüllen, wobei $\mathcal{C}_a := \{C \in \mathcal{C} : f(C) = 1\}$ die Menge der aktiven Komponenten von $\mathcal{C}$ bezeichnet ($\mathcal{C}$ ist natürlich wie im Algorithmus die Menge der Komponenten der zu Beginn des betrachteten Durchlaufs der `WHILE`-Schleife vorliegenden, noch unzulässigen Lösung). Um nun (12.12) zu zeigen, betrachte den Graphen H, der von den Kanten $\bigcup_{C \in \mathcal{C}} A' \cap \delta(C)$ auf der Knotenmenge $\mathcal{C}$ induziert wird. Da (V, A) ein Wald ist, ist H ein einfacher Graph und ebenfalls ein Wald. Die Komponenten $C \in \mathcal{C}$ haben in H gerade die Knotengrade

$$d_H(C) \;=\; |A' \cap \delta(C)|.$$

Aufgrund der Zulässigkeit von A' entspricht kein isolierter Knoten von H einer aktiven Komponente $C \in \mathcal{C}_a$. Entferne alle isolierten Knoten aus H (die also zu inaktiven Komponenten gehören). Sei nun $V(H) = V_a \dot\cup V_i$ die Zerlegung der Knotenmenge von H in die Menge der aktiven bzw. der inaktiven Komponenten. Nehmen wir für einen Moment an, alle Blätter von H seien aktive Komponenten. Dann gilt $d_H(C) \geq 2$ für alle inaktiven Komponenten $C \in V_i$, und wir können wie folgt abschätzen:

$$\begin{aligned}
\sum_{C \in \mathcal{C}_a} |A' \cap \delta(C)| &= \sum_{C \in V_a} d_H(C) \\
&= \sum_{C \in V_a \cup V_i} d_H(C) - \sum_{C \in V_i} d_H(C) \\
&\leq 2(|V_a| + |V_i| - 1) - 2|V_i| \\
&= 2(|\mathcal{C}_a| - 1) \\
&\leq 2(1 - 1/\ell)|\mathcal{C}_a|,
\end{aligned}$$

wobei die letzte Ungleichung gilt, weil die Anzahl der aktiven Komponenten, wie oben gesehen, schwach monoton fällt. Damit ist der Satz gezeigt.

Es bleibt also, zu zeigen, daß alle Blätter von H zu aktiven Komponenten gehören. Angenommen, ein Blatt C_b von H korrespondiert zu einer inaktiven Komponente, d.h. es gilt $f(C_b) = 0$. Wir zeigen, daß dann der Algorithmus APPROX_PROPER_0-1 im abschließenden Verbesserungsschritt die mit C_b in H inzidierende Kante e aus A entfernt hätte – im Widerspruch zu $e \in A'$. Sei $C \subseteq V$ die Komponente von (V, A) (!), die C_b enthält; C ist also die Vereinigung von Komponenten aus $\mathcal{C}$. Da A eine primal zulässige Lösung ist, gilt $f(C) = 0$. Sei $N \subset V$ die Zusammenhangskomponente von $C - e$, die C_b enthält, und seien $C_1, \ldots, C_k$ – neben C_b – die anderen Komponenten von $\mathcal{C}$ in N. Seien $e_1, \ldots, e_h$ die Kanten in A, die C_b mit C_i's verbinden, und sei N_i, $1 \leq i \leq h$, die Komponente von $C - e_i$, so daß $C_b \subset C \setminus N_i$. Jedes N_i ist i.a. die Vereinigung von C_j's, und es gilt

$$C_1 \dot\cup \ldots \dot\cup C_k \;=\; N_1 \dot\cup \ldots \dot\cup N_h \;=\; N \setminus C_b$$

Da C_b ein Blatt von H ist und H nur Kanten aus A' enthält, sind die Kanten $e_1, \ldots, e_h$ offenbar vom letzten Schritt des Algorithmus aus A entfernt worden, d.h. es ist $e_i \in A \setminus A'$ für $1 \leq i \leq h$. Nach Definition von A' gilt somit $f(N_i) = 0$, $1 \leq i \leq h$. Aus der Maximalität von f folgt hieraus $f(N) = f(C_b \dot\cup N_1 \dot\cup \ldots \dot\cup N_h) = 0$, und wegen $f(C) = 0$ mittels Komplementarität (Lemma 12.3.9) schließlich auch $f(C \setminus N) = 0$. Also hätte e im letzten Schritt aus A entfernt werden müssen, Widerspruch. □

Proposition 12.3.12 *Der Algorithmus* APPROX_PROPER_0-1 *kann (i.a.) mit Laufzeit* $\mathcal{O}(nm)$ *implementiert werden.*

Beweis. Die Komponenten $\mathcal{C}$ lassen sich mit Hilfe der Datenstruktur, wie sie für den Algorithmus von KRUSKAL verwendet wurde, mit Gesamtaufwand $\mathcal{O}(n \log n)$ verwalten.

Die `WHILE`-Schleife wird höchstens $(n-1)$-mal durchlaufen, der Aufwand zur Bestimmung von ϵ und e beträgt jedesmal $\mathcal{O}(m)$. Die Aktualisierung von d kostet jedesmal $\mathcal{O}(n)$ Zeit. Die mit der `WHILE`-Schleife verbundenen Operationen kosten also $\mathcal{O}(nm)$ Zeit. Insgesamt sind für die Verwaltung der aktiven Komponenten von $\mathcal{C}$ höchstens $2n - 1$ Aufrufe von f nötig.

Zur Berechnung von A' aus A: eine betrachtete Kante $e \in A$ liege in einer Komponente C von (V, A). Wegen der Komplementarität (beachte $f(C) = 0$) genügt der Aufruf von f für nur *eine* Komponente von $C - e$. Da A ein Wald ist, sind also für den abschließenden Verbesserungsschritt von APPROX_PROPER_0-1 höchstens $n-1$ Auswertungen von f nötig sowie $\mathcal{O}(n^2)$ weitere Operationen. Mit Hilfe von Dynamischer Programmierung genügt für diese Operationen sogar $\mathcal{O}(n)$ Zeit (Übung).

Für alle interessanten Anwendungen scheint ein Aufruf von f höchstens $\mathcal{O}(n)$ Zeit zu kosten. Die mit Auswertungen von f verbundenen Operationen kosten also (i.a.) $\mathcal{O}(n^2)$ Zeit. □

GOEMANS und WILLIAMSON [GW95] geben auch eine $\mathcal{O}(n^2 \log n)$-Implementierung von APPROX_PROPER_0-1 für dünne Graphen an.

Übung 12.3.13 *Das Problem* MINIMUM SPANNING TREE *läßt sich als ein ganzzahliges Programm (IP) formulieren mit der Funktion f definiert durch $f(S) = 1$ für alle $\emptyset \neq S \subset V$. Man zeige, daß* APPROX_PROPER_0-1 *in diesem Fall den Algorithmus von* KRUSKAL *simuliert.*

Weil schnelle exakte Algorithmen für das MIN-COST PERFECT MATCHING Problem relativ aufwendig sind (z.B. der $\mathcal{O}(n(m + n \log n))$-Algorithmus von GABOW [Gab90]), sucht man schon seit 15 Jahren nach effizient(er)en und einfachen Approximationsalgorithmen für dieses Problem. Eine interessante Anwendung des Algorithmus APPROX_PROPER_0-1 ist:

Korollar 12.3.14 (GOEMANS, WILLIAMSON 1995) *Es gibt einen $\mathcal{O}(nm)$-Approximationsalgorithmus für* MIN-COST PERFECT MATCHING *mit Güteratio 2 in Graphen $G = (V, E, c)$, deren Gewichtsfunktion $c : E \to \mathbb{Q}^+$ die Dreiecksungleichung erfüllt.*

Beweis. Sei $G = (V, E, c)$ ein Graph gerader Ordnung. c genüge der Dreiecksungleichung. Wir wenden den Algorithmus APPROX_PROPER_0-1 an mit der Funktion

$$f(S) := \begin{cases} 1 & \text{falls } |S| \text{ ungerade,} \\ 0 & \text{sonst.} \end{cases}$$

Der Algorithmus liefert dann eine Kantenmenge $A' \subseteq E$ zurück, so daß in (V, A') jede Komponente ein Baum auf gerade vielen Knoten ist. Um daraus ein perfektes Matching zu erhalten, beachte, daß jeder Knoten in (V, A') ungeraden Grad hat: Ein $u \in V$ liege in einer Komponente T von (V, A') und inzidiere mit Kanten $\{u, v_i\}$, $i = 1, \ldots, k$. Da APPROX_PROPER_0-1 die Kante $\{u, v_i\}$ im letzten Schritt (d.h. bei der Berechnung von A' aus A) nicht aus A entfernt hat, ist die Funktion f wegen Lemma 12.3.9 auf beiden Komponenten von $T - \{u, v_i\}$ gleich 1. Wenn wir u als Wurzel von T auffassen, hat also für jedes $i = 1, \ldots, k$ der Teilbaum von T, der v_i enthält, ungerade viele Knoten. Da T selbst gerade viele Knoten enthält, folgt, daß k ungerade ist.

Für jeden Knoten u in (V, A'), der (ungeraden) Grad größer als 1 hat, entferne nun sukzessive zwei anliegende Kanten und verbinde stattdessen die entsprechenden Nachbarn von u durch eine Kante. Aufgrund der Dreiecksungleichung erhöht sich dadurch das Gewicht der Lösung nicht, sie enthält aber eine Kante weniger. Da alle Knoten nach wie vor ungeraden Grad haben, verbleibt nach $\mathcal{O}(n)$ derartigen Schritten ein perfektes Matching. □

12.3.5 SURVIVABLE NETWORK DESIGN mit Mehrfachkanten

Das SURVIVABLE NETWORK DESIGN Problem ist die Verallgemeinerung des GENERALIZED STEINER TREE Problems auf $\mathbb{N}$-wertige Funktionen f. Es ist wie folgt definiert. Gegeben ein Graph $G = (V, E, c)$ mit positiven Kantengewichten $c : E \to \mathbb{Q}^+$ und Zahlen $r_{ij} \in \mathbb{N}_0$ für alle $\{i, j\} \in \binom{V}{2}$, finde einen Subgraphen (V, A) von G von minimalem Gewicht $c(A) := \sum_{e \in A} c_e$, so daß A für jedes Paar $\{i, j\} \in \binom{V}{2}$ mindestens r_{ij} kantendisjunkte $i-j$-Pfade enthält. Dieses Problem läßt sich dank des Satzes 2.2.1 von MENGER wie folgt als ein ganzzahliges lineares Programm schreiben:

$$\begin{aligned} & \min\ c^T x & & \\ s.t. \quad & x(\delta(S)) \geq f(S) & & \emptyset \neq S \subset V \\ & x_e \in \{0, 1\} & & e \in E, \end{aligned} \tag{IP}$$

wobei $f(S) := \max\{r_{ij} : i \in S, j \notin S\}$ (beachte, daß diese Funktion f proper ist). Ein interessanter Spezialfall des SURVIVABLE NETWORK DESIGN Problems ist das Problem MIN COST k-EDGE-CONNECTED SUBGRAPH, wobei $r_{ij} \equiv k$ für alle $\{i, j\} \in \binom{V}{2}$. Man beachte, daß es selbst im ungewichteten Fall und für $k = 2$ NP-schwer ist (Reduktion von HAMILTONIAN CIRCUIT).

Wir behandeln hier die folgende (erheblich einfachere) Relaxation des SURVIVABLE NETWORK DESIGN Problems: Wir lassen in der Lösung Mehrfachkanten zu, d.h. wir approximieren das folgende Problem:

$$\begin{aligned} & \min\ c^T x & & \\ s.t. \quad & x(\delta(S)) \geq f(S) & & \emptyset \neq S \subset V \\ & x_e \in \mathbb{N}_0 & & e \in E, \end{aligned} \tag{IP'}$$

wobei f proper. Diese Formulierung modelliert allerdings weniger Ausfallsicherheit (bei mehreren, parallel verlegten Leitungen wird die Wahrscheinlichkeit eines Verbindungsausfalls beispielsweise aufgrund von Bauarbeiten annähernd gleich bleiben) sondern eher ein Netzwerk Design hinsichtlich der Auslastung des Netzwerkes (man interpretiere die r_{ij} als Kommunikationsanforderungen an das Netz).

Die Idee ist, dieses Problem auf eine Reihe von GENERALIZED STEINER TREE Problemen zurückzuführen, die jeweils mit dem Algorithmus APPROX_PROPER_0-1 gelöst werden. Definiere $f_{\max} := \max\{f(S) : \emptyset \neq S \subset V\}$. Für eine proper Funktion ist dann $f_{\max} = \max\{f(\{v\}) : v \in V\}$.

```
FUNCTION Approx_Proper_With_Duplicates (G, c, f);

BEGIN
  A := ∅;                                   {Multimenge von Kanten}
  f_max := max {f({v}) | v ∈ V};
  FOR p := ⌊log f_max⌋ DOWNTO 0 DO BEGIN
```

```
    {Phase p}
    h_p(S) := { 1  falls f(S) ≥ 2^p,
              { 0  sonst;
    F_p := Approx_Proper_0-1 (G, c, h_p);
    A := A + 2^p ⋆ F_p;
  END;
  Approx_Proper_With_Duplicates := A;
END;
```

Hierbei bedeutet die Anweisung $A := A + 2^p \star F_p$, daß wir von jeder Kante $e \in F_p$ genau 2^p Kopien zu A hinzufügen.

Satz 12.3.15 Approx_Proper_With_Duplicates *konstruiert für eine proper Funktion f eine Lösung A von (IP'), die jede Kante von G weniger als $2f_{\max}$-mal benutzt und für deren Gewicht gilt:*

$$c(A) \;\leq\; 2\,(\lfloor f_{\max} \rfloor + 1) \cdot Z^*_{IP'}.$$

Beweis. Zunächst ist leicht zu sehen, daß die Funktion

$$h_p(S) := \begin{cases} 1 & \text{falls } f(S) \geq 2^p, \\ 0 & \text{sonst} \end{cases}$$

proper ist (Übung). Außerdem benutzt der Algorithmus offensichtlich höchstens

$$\sum_{p=0}^{\lfloor \log f_{\max} \rfloor} 2^p \;=\; 2^{\lfloor \log f_{\max} \rfloor + 1} - 1 \;<\; 2f_{\max}$$

Kopien einer Kante $e \in E$. Auch daß A eine zulässige Lösung von (IP') darstellt, ist unmittelbar einsichtig: Sei $\emptyset \neq S \subset V$ und $f(S) = a \geq 1$. Der Algorithmus setzt $h_p(S) = 1$ in den Phasen $\lfloor \log a \rfloor, \ldots, 0$; also gilt

$$|A \cap \delta(S)| \;\geq\; \sum_{p=\lfloor \log a \rfloor}^{0} 2^p \;=\; 2^{\lfloor \log a \rfloor + 1} - 1 \;>\; a - 1,$$

d.h. A enthält mindestens a viele Kanten aus dem Schnitt $\delta(S)$.

Zur Güteratio: bezeichne (LP_p) die LP-Relaxation des ganzzahligen linearen Programms

$$\begin{array}{lll} & \min\; c^T x & \\ s.t. & x(\delta(S)) \;\geq\; h_p(S) & \emptyset \neq S \subset V \\ & x_e \in \{0,1\} & e \in E, \end{array}$$

das Approx_Proper_With_Duplicates in der p-ten Phase löst, und bezeichne Z^*_p den Wert einer Optimallösung von (LP_p). Dann gilt nach Satz 12.3.11 $c(F_p) \leq 2Z^*_p$. Sei $\hat{x}$ eine Optimallösung von (IP'). Dann ist $\frac{1}{2^p}\hat{x}$ eine zulässige Lösung von (LP_p), und daher $Z^*_p \leq \frac{1}{2^p} Z^*_{IP'}$. Es folgt

$$c(A) \;=\; \sum_{p=0}^{\lfloor \log f_{\max} \rfloor} 2^p c(F_p)$$

$$\begin{aligned} &\leq \sum_{p=0}^{\lfloor \log f_{\max} \rfloor} 2^p \cdot 2 \cdot \frac{1}{2^p} Z^*_{IP'} \\ &= 2\,(\lfloor \log f_{\max} \rfloor + 1)\, Z^*_{IP'} \end{aligned}$$

□

Anmerkungen

Die Ergebnisse dieses Abschnitts sind im wesentlichen der Arbeit [GW95] entnommen, die sich ihrerseits auf [GB93, AKR95] stützt. Der Durchbruch für die primal-dual Methode war vielleicht die Approximation des SURVIVABLE NETWORK DESIGN Problems (ohne Mehrfachkanten) bis auf einen Faktor von $2H(f_{\max})$, $H(k) := \sum_{i=1}^{k} 1/i$ die k-te Harmonische Zahl. Auch hierbei wird das $\mathbb{N}$-wertige Problem auf eine Folge von 0-1-Problemen reduziert, wobei die Funktion h nun allerdings nicht mehr proper ist, siehe [WGMV95, GGP+94]. Für den Spezialfall MIN COST k-EDGE-CONNECTED SUBGRAPH hatten KHULLER und VISHKIN [KV94] schon zuvor einen einfachen 2-Approximationsalgorithmus gefunden. Für das analoge MIN COST k-VERTEX-CONNECTED SUBGRAPH hingegen hat der beste bekannte Algorithmus Güteratio $2H(f_{\max})$ [RW96] und benutzt ebenfalls die primal-dual Methode. Approximationsalgorithmen mit konstanter Güteratio sind nur für $k = 2, 3$ bekannt. Für den Fall allerdings, daß die Kantengewichte die Dreiecksungleichung erfüllen, gaben KHULLER und RAGHAVACHARI [KR96] einen $2 + \frac{2(k-1)}{n}$-Approximationsalgorithmus. Eine Übersicht zur primal-dual Methode und Anwendungen bei Netzwerk-Design Problemen findet man in [GW96, Wil93].

Eine weitere interessante Anwendung der primal-dual Methode ist die Approximation des ganzzahligen Mehrgüterflußproblems auf Bäumen [GVV96].

12.4 Approximation von SET COVER und DOMINATING SET

Viele graphentheoretische Optimierungsprobleme lassen sich als ein SET COVER – Problem formulieren. Hierbei ist ein Hypergraph $\mathcal{H} = (U, \mathcal{F})$ mit $\mathcal{F} = \{F_1, \ldots, F_m\}$, $U := \bigcup_{F \in \mathcal{F}} F$, $|U| = n$, gegeben. Gesucht ist ein Subsystem $\mathcal{C} \subseteq \mathcal{F}$ des Mengensystems $\mathcal{F}$, das minimal viele Mengen $F \in \mathcal{F}$ enthält, das aber noch immer ganz U überdeckt: $\bigcup_{F \in \mathcal{C}} F = U$. Das SET COVER Problem ist also die natürliche Verallgemeinerung des EDGE COVER Problems von Graphen auf Hypergraphen. Während allerdings das Problem EDGE COVER auf Graphen durch Reduktion auf ein Matching-Problem in polynomieller Zeit lösbar ist (siehe Satz 4.1.9 von GALLAI), ist SET COVER NP-vollständig, da es u.a. die Probleme NODE COVER, DOMINATING SET, GRAPH COLORING oder CLIQUE COVER als Spezialfälle enthält: Für NODE COVER definiere dazu F_v, $v \in V$, als die Menge der mit $v \in V$ inzidierenden Kanten, für DOMINATING SET $F_v := \Gamma(v) + v$, $v \in V$, für GRAPH COLORING $\mathcal{F}$ als die Menge aller stabilen Knotenmengen in G sowie für CLIQUE COVER $\mathcal{F}$ als die Menge aller Kantenmengen von Cliquen in G. Beachte, daß bei NODE COVER jedes Element von nur konstant vielen Mengen überdeckt wird und bei DOMINATING SET immerhin noch $m = |\mathcal{F}|$ polynomiell in $n = |U|$ beschränkt ist. Bei GRAPH COLORING und CLIQUE COVER dagegen ist $\mathcal{F}$ nur noch implizit gegeben und i.a. exponentiell in n groß. Diese Unterschiede spiegeln sich deutlich in der Approximierbarkeit der Teilprobleme wieder. Während NODE COVER selbst in der gewichteten Form einen 2-Approximationsalgorithmus besitzt, sahen wir in den Abschnitten 11.3 und 12.1, daß die Probleme GRAPH COLORING und CLIQUE COVER sehr schwer zu approximieren sind und insbesondere keinen Approximationsalgorithmus mit endlicher Güteratio gestatten.

Das Problem DOMINATING SET hingegen ist genauso schwierig zu approximieren wie das allgemeine Problem SET COVER.

Satz 12.4.1 (PAZ, MORAN 1981) *Genau dann gibt es einen Approximationsalgorithmus für* DOMINATING SET *mit Güteratio r, wenn es auch einen Approximationsalgorithmus für* SET COVER *mit Güteratio r gibt.*

Beweis. Die obige Reduktion von DOMINATING SET auf SET COVER erhält die Approximationsratio eines Approximationsalgorithmus.
Es bleibt, auch umgekehrt SET COVER auf DOMINATING SET zu reduzieren. Sei $\mathcal{F} = \{F_1, \ldots, F_m\}$ eine Instanz von SET COVER, $U := \bigcup_{i=1}^{m} F_i$.
Konstruiere daraus einen Graphen auf der Knotenmenge $U \cup \mathcal{F}$, in dem U eine stabile Menge und $\mathcal{F}$ eine Clique induziert und ein $u \in U$ genau dann mit einer Hyperkante F_i durch eine Kante verbunden ist, wenn $u \in F_i$.
Seien $D^* \subseteq U \cup \mathcal{F}$ und $J^* \subseteq \mathcal{F}$ eine minimum dominierende Menge in G bzw. eine minimum Mengenüberdeckung zu $\mathcal{F}$. Da J^* insbesondere auch eine dominierende Menge in G ist, gilt $|D^*| \leq |J^*|$. Andererseits gilt aber auch $|J^*| \leq |D^*|$, denn indem man in D^* jedes Element $u \in D^* \cap U$ durch einen beliebigen Knoten $F_{i(u)} \in \mathcal{F}$ mit $F_{i(u)} \ni u$ ersetzt, erhält man eine Mengenüberdeckung von $\mathcal{F}$ der Kardinalität $\leq |D^*|$.
Also gilt $|D^*| = |J^*|$, und die Reduktion erhält die Approximationsratio eines Approximationsalgorithmus für DOMINATING SET. □

Betrachten wir die Greedy-Heuristik für SET COVER. Da später der Beweis für ihre Güteratio natürlicher und erhellender für das gewichtete Problem geführt werden kann, gehen wir davon aus, daß zudem eine Kostenfunktion $c : \mathcal{F} \to \mathbb{Q}^+$ auf den Hyperkanten gegeben ist. Wir schreiben hier abkürzend $c_j := c(F_j)$ für $F_j \in \mathcal{F}$. Gesucht ist nun eine Indexmenge $J \subseteq \{1, \ldots, m\}$, so daß die zugehörigen Hyperkanten $\{F_j : j \in J\}$ eine Mengenüberdeckung von U mit minimalen Kosten $c(J) := \sum_{j \in J} c_j$ bilden. Der folgende Algorithmus nimmt stets eine solche Hyperkante F_j in die Überdeckung J auf, mit der sich „am billigsten" bisher noch nicht überdeckte Knoten $u \in U$ überdecken lassen.

```
PROCEDURE Greedy_Set_Cover;

BEGIN
  J_GSC := ∅;
  WHILE ∃F_j ≠ ∅ DO BEGIN
    k := argmin { c_j / |F_j| : 1 ≤ j ≤ m, F_j ≠ ∅ };
    J_GSC := J_GSC + k;
    FOR F_j ≠ ∅ DO  F_j := F_j \ F_k;
  END;
END;
```

Laufzeit. Da in jedem Schritt mindestens ein Element von U überdeckt wird, wird die `WHILE`-Schleife höchstens n-mal durchlaufen. Der Aufwand zur Bestimmung aller $argmin$'s beträgt also insgesamt $\mathcal{O}(mn)$ Zeit. Auch die Anzahl sämtlicher Operationen, die mit den Updates der F_j's verbunden sind (summiert über alle `WHILE`-Schleifendurchläufe), ist offenbar $\mathcal{O}(mn)$, wenn es gelingt, ein Element in konstanter Zeit aus jeder Menge F_j zu entfernen, der es angehört (beachte: jedes Element $u \in U$ wird genau einmal aus den Hyperkanten entfernt). Dies ist leicht möglich, wenn man die beiden folgenden Dinge beachtet. Zum einen stellt man für jedes Element $u \in U$ eine Liste derjenigen F_j's zur

Verfügung, in denen u enthalten ist (diese Liste kann offenbar zu Beginn in Zeit $\mathcal{O}(mn)$ berechnet werden). Des weiteren werden die Elemente einer jeden Menge F_j in einer doppelt verketteten Liste gespeichert, wobei ein zusätzliches Pointerfeld $Position[i,j]$ auf die Position des Elementes $u_i \in U$ in der Liste von F_j verweist (falls $u_i \in F_j$). Somit kann der Algorithmus GREEDY_SET_COVER (kurz GSC-Algorithmus) mit Zeitkomplexität $\mathcal{O}(nm)$ implementiert werden. Man beachte, daß m exponentiell in n sein kann. Und tatsächlich ist beispielsweise in den Fällen GRAPH COLORING und CLIQUE COVER der Schritt, das *argmin* zu bestimmen, NP-schwer, denn er läuft darauf hinaus, eine maximum stabile Menge bzw. eine maximum gewichtete Clique in einem Graphen zu finden.

Approximationsgüte. Der GSC-Algorithmus spezialisiert, angewandt auf das Mengensystem für NODE COVER (mit Gewichten identisch 1), auf den schon bekannten Greedy-Algorithmus für NODE COVER. Daher zeigt die dort untersuchte Graphenfamilie U_k, für die der Greedy-Algorithmus für NODE COVER Approximationsgüte $\Omega(\ln k \ - 1)$, $k = \Delta(U_k) = \max_j |F_j|$, hatte, daß auch die Approximationsgüte von GREEDY_SET_COVER i.a. nicht besser sein kann. Das folgende Ergebnis zeigt, daß der GSC-Algorithmus zumindest nie schlechter approximiert als in diesem Beispiel.

Satz 12.4.2 (CHVÁTAL 1979) *Bezeichne J_{opt} eine optimale Mengenüberdeckung für eine gewichtete Mengenfamilie $(\mathcal{F}, c)$. Dann gilt für die vom Algorithmus* GREEDY_SET_COVER *konstruierte Mengenüberdeckung J_{GSC}:*

$$\frac{c(J_{GSC})}{c(J_{opt})} \quad \leq \quad H(\Delta(\mathcal{F})) \quad \leq \quad 1 + \ln \Delta(\mathcal{F}), \tag{12.13}$$

wobei $H(x) := \sum_{i=1}^{x} \frac{1}{i}$ für $x \in \mathbb{N}$ und $\Delta(\mathcal{F}) := \max\{|F| : F \in \mathcal{F}\}$.

Die zweite Abschätzung in (12.13) ist offenbar lediglich die Integralschranke $1 + \int_1^{\Delta} \frac{dx}{x}$. Die folgende Instanzenfamilie $\mathcal{F}^m$, $m \in \mathbb{N}^{\geq 2}$, zeigt, daß die erste Ungleichung in (12.13) bestmöglich ist. Sei $\mathcal{F}^m = \{F_j : \ 1 \leq j \leq m+1\}$ mit $F_j := \{j\}$, $1 \leq j \leq m$, und $F_{m+1} := U = \{1, \ldots, m\}$. Setze $c_j := 1/j$ für $1 \leq j \leq m$ und $c_{m+1} := 1 + \epsilon$, $\epsilon > 0$. Dann wählt der GSC-Algorithmus sukzessive die Mengen $F_m, F_{m-1}, \ldots, F_1$ aus mit Gesamtkosten $\sum_{j=1}^{m} 1/j$, während die Optimallösung $J_{opt} = \{m+1\}$ heißt mit Kosten $1 + \epsilon$. Für $\epsilon \to 0$ konvergiert die Approximationsratio also gegen die im Satz behauptete (beachte: $\Delta(\mathcal{F}^m) = m$).
Die erste Ungleichung in (12.13) beruht auf folgendem Lemma.

Lemma 12.4.3 *Zu jeder Greedy-Mengenüberdeckung J_{GSC} von $(\mathcal{F}, c)$ existieren nichtnegative Zahlen $\pi(u)$, $u \in U$, so daß*

(i) $\sum_{u \in U} \pi(u) \quad = \quad \sum_{j \in J_{GSC}} c_j \quad = \quad c(J_{GSC})$;

(ii) $\sum_{u \in F_j} \pi(u) \quad \leq \quad H(|F_j|) \cdot c_j \qquad$ *für alle* $1 \leq j \leq m$.

Beweis. Wir lassen uns von der folgenden anschaulichen Bedeutung der $\pi(u)$ leiten. Der Wert $\pi(u)$ läßt sich interpretieren als der Preis, den der GSC-Algorithmus bezahlt, um den Knoten $u \in U$ zu überdecken.
Um zu einer mathematischen Definition der $\pi(u)$ zu gelangen, müssen die einzelnen Schritte des GSC-Algorithmus analysiert werden. O.B.d.A. wählt der GSC-Algorithmus in $t \in \mathbb{N}$ Schritten (Iterationen) die Hyperkanten $J_{GSC} = \{1, \ldots, t\}$ aus, so daß

$c(J_{GSC}) = \sum_{s=1}^{t} c_s$. Wir benötigen folgende Notation. $F_j^s \subseteq F_j$ bezeichne die Menge der Knoten in der j-ten Hyperkante, die zu Beginn des s-ten Schritts noch nicht überdeckt sind. Eingangs gilt also $F_j^1 = F_j$, $j = 1, \ldots, m$. $k_j^s := |F_j^s|$ bezeichne die Kardinalität der F_j^s. Dann gilt nach Definition des GSC-Algorithmus:

(1) Im s-ten Schritt, $s = 1, \ldots, t$, werden die Knoten F_s^s überdeckt (so daß $F_s^{s+1} = \emptyset$);

(2) Im s-ten Schritt, $s = 1, \ldots, t$, gilt:

$$\frac{c_s}{k_s^s} \;\leq\; \frac{c_j}{k_j^s} \qquad \text{für alle } j \in \{1, \ldots, m\};$$

(3) Für alle $j \in \{1, \ldots, m\}$ und für alle $s \in \{1, \ldots, t\}$ gilt:

$$F_j^s \setminus F_j^{s+1} \;=\; F_j^s \cap F_s^s;$$

(4) Die Mengen F_s^s bilden eine Partition des Universums U:

$$U \;=\; \dot{\bigcup}_{s=1}^{t} F_s^s.$$

Aufgrund von (4) können wir Zahlen $\pi(u)$ wie folgt definieren:

$$\pi(u) := \frac{c_s}{k_s^s} \quad \Leftrightarrow \quad u \in F_s^s, \tag{12.14}$$

d.h. wir verteilen die im s-ten Schritt anfallenden Kosten c_s gleichmäßig auf die im s-ten Schritt neu überdeckten Knoten F_s^s. Damit haben wir trivialerweise schon (i) erfüllt:

$$\begin{array}{lcll} \sum_{u \in U} \pi(u) & \overset{(4)}{=} & \sum_{s=1}^{t} \sum_{u \in F_s^s} \pi(u) & \\ & \overset{(12.14)}{=} & \sum_{s=1}^{t} |F_s^s| \, \frac{c_s}{k_s^s} & \\ & = & \sum_{s=1}^{t} c_s & = \quad c(J_{GSC}). \end{array}$$

Die linke Seite von (ii) schätzen wir wie folgt nach oben ab. Sei $j \in \{1, \ldots, m\}$ beliebig aber fest, und bezeichne $1 \leq r \leq t$ den Schritt, in dem die Elemente aus F_j vollständig überdeckt werden, d.h. $r := \max\{s : 1 \leq s \leq t \;\wedge\; k_j^s > 0\}$.

$$\begin{aligned} \sum_{u \in F_j} \pi(u) \quad &\overset{(4)}{=} \quad \sum_{s=1}^{t} \sum_{u \in F_j \cap F_s^s} \pi(u) \\ &\overset{(3)}{=} \quad \sum_{s=1}^{t} \sum_{u \in F_j^s \setminus F_j^{s+1}} \pi(u) \\ &\overset{(12.14)}{=} \quad \sum_{s=1}^{t} \left| F_j^s \setminus F_j^{s+1} \right| \, \frac{c_s}{k_s^s} \\ &= \quad \sum_{s=1}^{r} \left(k_j^s - k_j^{s+1} \right) \, \frac{c_s}{k_s^s} \\ &\overset{(2)}{\leq} \quad c_j \cdot \sum_{s=1}^{r} \left(k_j^s - k_j^{s+1} \right) \, \frac{1}{k_j^s} \end{aligned}$$

Auf der rechten Seite von (ii) ergibt sich eine Teleskopsumme, die wir wie folgt nach unten abschätzen.

$$\begin{aligned} H(|F_j|) \;=\; H(k_j^1) \;&=\; \sum_{s=1}^{r} H(k_j^s) - H(k_j^{s+1}) \\ &=\; \sum_{s=1}^{r} \frac{1}{k_j^{s+1}+1} + \cdots + \frac{1}{k_j^s} \\ &\geq\; \sum_{s=1}^{r} \left(k_j^s - k_j^{s+1}\right) \frac{1}{k_j^s} \end{aligned}$$

Setzt man beide Abschätzungen zusammen, ergibt sich (ii). □

Beweis von Satz 12.4.2:
Sei J_{opt} eine optimale Mengenüberdeckung von $(\mathcal{F}, c)$. Dann überdeckt J_{opt} also jedes $u \in U$, und es folgt

$$\begin{array}{lllll} c(J_{GSC}) & \overset{(i)}{=} & \sum_{u\in U} \pi(u) & \leq & \sum_{k\in J_{opt}} \sum_{u\in F_k} \pi(u) \\ & \overset{(ii)}{\leq} & \sum_{k\in J_{opt}} H(|F_k|)\, c_k & \leq & H(\Delta(\mathcal{F}))\ \sum_{k\in J_{opt}} c_k \;=\; H(\Delta(\mathcal{F}))\ c(J_{opt}). \end{array}$$

□

Anwendung auf DOMINATING SET. Die Spezialisierung des Algorithmus GREEDY_SET_COVER für das DOMINATING SET Problem führt zum Algorithmus GREEDY_DS aus Lemma 5.4.8. Da die Graphenfamilie $\{U_k\}_{k\in\mathbb{N}}$ von Seite 282 auch eine schwer zu approximierende Instanzenfamilie für GREEDY_DS ist, folgt:

Korollar 12.4.4 GREEDY_DS *ist ein Approximationsalgorithmus für* DOMINATING SET *mit Güteratio* $R_{\text{GREEDY\_DS}} \sim \ln(\Delta(G))$. □

Nicht-Approximierbarkeit. Tatsächlich gibt es für DOMINATING SET (und damit für SET COVER) modulo einer komplexitätstheoretischen Beziehung, die als unwahrscheinlich angesehen wird, keinen Approximationsalgorithmus mit endlicher Gütegarantie – während der GSC-Approximationsalgorithmus mit Güteratio $1 + \ln n$ bis auf Terme kleinerer Ordnung optimal ist:

Satz 12.4.5 (BELLARE, GOLDWASSER, LUND, RUSSELL 1993; FEIGE 1996)

(i) Es ist NP*-schwer,* SET COVER *mit konstanter Güteratio zu approximieren;*

(ii) Für jedes $\epsilon > 0$ *gilt: wenn es einen Approximationsalgorithmus für* SET COVER *mit Güteratio* $(1-\epsilon)\ \ln n$ *gibt, so ist* $\text{NP} \subseteq \text{DTIME}\left(n^{\mathcal{O}(\log\log n)}\right)$. □

Im Vergleich mit NODE COVER und GRAPH COLORING ist also das Approximationsproblem für DOMINATING SET von mittlerem Schwierigkeitsgrad.

Anmerkungen und Übung

Angewandt auf das EDGE COVER Problem, stellt GREEDY_SET_COVER ein 3/2-Approximationsalgorithmus dar. Doch reduziert sich EDGE COVER auf das allgemeine Matching-Problem (vgl. Satz 4.1.9), liegt also in P.

Die Schranke (12.13) kann wie folgt verbessert werden [Sla95]:

$$\frac{c(J_{GSC})}{c(J_{opt})} \leq \ln n - \Theta(\ln \ln n).$$

Im ungewichteten Fall zeigten die Schranke (12.13) bereits JOHNSON [Joh74a], STEIN [Ste74] und LOVÁSZ [Lov75b]. Für das sogenannte Baum-repräsentierbare SET COVER Problem gibt es einen 2-Approximationsalgorithmus [GVV96]. Auch im Fall beschränkter „VC-Dimension" gibt es einen besseren Approximationsalgorithmus, siehe [BG95].

Übung 12.4.6 GREEDY_DS *kann (im ungewichteten Fall) mit Laufzeit* $\mathcal{O}(n^2)$ *(*$\mathcal{O}(n+m)$*) implementiert werden.*

Übung 12.4.7 *Sei* $k \geq 3$ *fest. Das Problem* PARTITION INTO INDEPENDENT SETS OF SIZE $\leq k$ *(vergleiche Übung 5.4.12), die minimale Anzahl von Farben, mit denen ein Graph gefärbt werden kann, ohne daß eine Farbklasse mehr als* k *Knoten enthält, zu bestimmen, besitzt*
a) *einen* $H(k)$*-Approximationsalgorithmus auf der Menge aller Graphen;*
b) *einen 2-Approximationsalgorithmus auf jeder Graphenklasse, auf der man eine optimale (gewöhnliche) Knotenfärbung in polynomieller Zeit berechnen kann* [Jan96]*;*
c) *einen 5/3-Approximationsalgorithmus für* $k = 3$ [GHY93]. [*Hinweis:* vergleiche Übung 10.9.18b]

Einen Approximationsalgorithmus für den Fall $k = 3$ mit Güteratio $1.4 + \epsilon$ für jedes $\epsilon > 0$ gibt [Hal96b].

Kapitel 13

Zufällige Graphen und probabilistische Algorithmen

Wir werden in diesem Kapitel zeigen, daß es eine „typische Größe“ sowohl der größten Clique als auch der chromatischen Zahl in einem zufällig herausgegriffenen Graphen auf n Knoten gibt und diese berechnen. Außerdem werden wir nachweisen, daß die Greedy-Algorithmen für CLIQUE und GRAPH COLORING für fast alle Graphen eine gute Approximation liefern.

Zur Untersuchung von typischen Eigenschaften von Graphen erweist es sich als nützlich, auf Begriffe aus der Wahrscheinlichkeitstheorie zurückzugreifen. Im Anhang A.1 befindet sich eine Zusammenstellung der in diesem Kapitel verwendeten grundlegenden Definitionen und Sätze aus der Wahrscheinlichkeitstheorie. Zufällige Graphen wurden bereits im Abschnitt 1.1.4 eingeführt. Wir betrachten in diesem Kapitel den mit dem LAPLACEmaß versehenen Wahrscheinlichkeitsraum $\mathcal{G}_n = \mathcal{G}_{n,1/2}$ aller markierten Graphen auf n Knoten. G_n bezeichne ein zufälliges Element aus $\mathcal{G}_n$, also einen Graphen, bei dem alle Kanten unabhängig voneinander mit der Wahrscheinlichkeit $p = 1/2$ vorhanden sind.

Zufällige Graphen aus $\mathcal{G}_{n,p}$ sind Gegenstand zahlreicher Untersuchungen. [Bol85] bietet eine Übersicht über die Ergebnisse bis etwa 1985. Einige spätere Entwicklungen sind in [ASE92] dargestellt. In [Prö94] werden die in diesem Kapitel behandelten und angrenzende Fragestellungen diskutiert. Die meisten Resultate dieses Kapitels gelten in leicht abgewandelter Form auch in $\mathcal{G}_{n,p}$ für konstantes p und häufig sogar für variables $p = p(n)$. Die Beweise für den allgemeinen Fall sind fast mit den hier vorgestellten identisch, und aus Gründen der Übersichtlichkeit beschränken wir uns deshalb auf zufällige Graphen aus $\mathcal{G}_n = \mathcal{G}_{n,1/2}$.

Wir sagen, daß *fast alle* Graphen die Eigenschaft $\mathcal{A}$ haben, beziehungsweise die Aussage $\mathcal{A}$ *fast sicher* gilt, falls $Prob\,(G_n \text{ hat } \mathcal{A}) = 1 - o(1)$ für $n \to \infty$. Für die meisten der hier untersuchten Eigenschaften gilt, daß sogar nur ein exponentiell kleiner Anteil aller Graphen die jeweilige Eigenschaft nicht hat. Im allgemeinen werden wir aber darauf verzichten, die besten bekannten Abschätzungen für diese Wahrscheinlichkeiten anzugeben, sondern nur Aussagen der Form „$\mathcal{A}$ gilt fast sicher“ beweisen.

13.1 Die Cliquenzahl

Cliquenzahl versus Stabilitätszahl. Intuitiv muß ein „dichter“ Graph, d.h. ein Graph

mit vielen Kanten, eine „große“ Cliquenzahl und ein „dünner“ Graph eine „große“ Stabilitätszahl haben; es können, so wird man vermuten, nicht beide Graphenparameter gleichzeitig „sehr klein“ sein. Der folgende Satz von RAMSEY, der eine ganze Disziplin der Diskreten Mathematik – die sogenannte Ramsey-Theorie – ins Leben rief, macht dies präzise. Wir betrachten ihn hier nur in seiner einfachsten Version, nämlich in seiner Formulierung für endliche Graphen.

Satz 13.1.1 (RAMSEY 1930) *Zu jedem $k \in \mathbb{N}$ gibt es ein $n \in \mathbb{N}$ mit $n \leq 0.5 + 2^{2k-3}$, so daß jeder Graph auf mindestens n Knoten eine k-Clique oder eine stabile Menge der Größe k enthält.*

Beweis. Wir induzieren nach k. Für $k = 1, 2$ ist die Aussage des Satzes offenbar trivial. Sei also $r \geq 3$ und G ein Graph der Ordnung mindestens 2^{2k-3}. Wir zeigen, daß $G = (V, E)$ eine k-Clique oder eine stabile Menge der Größe k enthält. Dazu konstruieren wir eine Folge $v_1, v_2, \ldots$ von mindestens $2k-2$ vielen Knoten, so daß für jedes i, $1 \leq i < 2k-2$, der Knoten v_i entweder zu allen späteren Knoten $v_{i+1}, v_{i+2}, \ldots$ benachbart oder aber zu all diesen Knoten nicht benachbart ist. Im ersten Fall heiße v_i N-Knoten, im zweiten $\overline{N}$-Knoten. Da $2k-3$ ungerade ist, gibt es in der Folge $v_1, v_2, \ldots, v_{2k-3}$ mindestens $k-1$ N-Knoten oder mindestens $k-1$ $\overline{N}$-Knoten. Im ersten Fall hat man eine $(k-1)$-Clique, im zweiten eine stabile Menge der Größe $k-1$ gefunden. Dieser fügt man noch den Knoten v_{2k-2} hinzu.

Die Knotenfolge $v_1, v_2, \ldots$ läßt sich wie folgt konstruieren.

```
i := 1;
V_1 := V;
WHILE V_i ≠ ∅ DO BEGIN
  wähle v_i ∈ V_i beliebig;
  Γ := Γ_{G[V_i]}(v_i);                                   {Γ ⊆ V_i}
  IF |Γ| > |V_i \ (Γ + v_i)| THEN
    {v_i ist N-Knoten}
    V_{i+1} := Γ;
  ELSE
    {v_i ist N̄-Knoten}
    V_{i+1} := V_i \ (Γ + v_i):
  i := i + 1;
END;
```

Es bleibt lediglich zu zeigen, daß die konstruierte Folge tatsächlich mindestens $2k-2$ viele Knoten enthält. Dazu wiederum genügt es offenbar, zu zeigen, daß $|V_i| \geq 2^{2k-2-i}$ ist, denn dann ist V_{2k-2} noch nicht leer. Dies gilt aber per Initialisierung zu Beginn für $i = 1$, und es gilt, vorausgesetzt, es gilt im i-ten Schritt, auch im $(i+1)-$ten Schritt, weil $\Gamma \dot\cup (V_i \setminus (\Gamma + v_i))$ eine Partition von $V_i - v_i$ ist, die größere der beiden Partitionsmengen als V_{i+1} gewählt wird und somit V_{i+1} mindestens

$$|V_{i+1}| \;\geq\; \lceil (|V_i| - 1)/2 \rceil \;\geq\; \left\lceil (2^{2k-2-i} - 1)/2 \right\rceil \;\geq\; 2^{2k-2-i}/2 \;=\; 2^{2k-2-(i+1)}$$

viele Knoten enthält. □

Obiger Satz besagt, daß jeder Graph der Ordnung $n \geq 2^{2k-3}$ eine Clique oder eine stabile Menge der Größe k enthält. Anders formuliert (vgl. mit Lemma 10.9.16):

Korollar 13.1.2 *Für jeden Graphen G gilt:* $\alpha(G) \geq \frac{1}{2}\log n \quad \vee \quad \omega(G) \geq \frac{1}{2}\log n$

Wenn man Cliquen und stabile Mengen als regelmäßige Unterstrukturen eines Graphen auffaßt, so zeigte ERDŐS im Jahr 1947 mit Hilfe eines einfachen Abzählarguments die Existenz von Graphen, die keine nennenswert größeren regelmäßigen Strukturen enthalten, als sie Satz 13.1.1 garantiert. Dieses Ergebnis stellt eine der frühesten Anwendungen probabilistischer Argumente in der Graphentheorie dar.

Satz 13.1.3 (ERDŐS 1947) *Zu jedem $n \in \mathbb{N}$ mit $n \geq 3$ gibt es einen Graphen H auf n Knoten mit*

$$\alpha(H) < 2\log n \qquad \textit{und} \qquad \omega(H) < 2\log n.$$

Beweis. Sei $G_n = (V_n, E)$ ein zufälliger Graph auf n Knoten $\ell \in \mathbb{N}$, und sei $L \subseteq V_n$ eine ℓ-elementige Teilmenge der Knotenmenge von G_n. Dann bezeichne C_L das Ereignis, daß L in G_n eine Clique induziert, und S_L das Ereignis, daß L in G_n eine stabile Menge induziert. Dann gilt

$$Prob\,(C_L) \;=\; Prob\,(S_L) \;=\; Prob\left(\bigcap_{\{u,v\}\in\binom{L}{2}} \{\{u,v\} \notin G_n\}\right) \;=\; 2^{-\binom{\ell}{2}}.$$

Daraus folgt

$$\begin{aligned} Prob\;((\omega(G_n) \geq \ell) \text{ oder } (\alpha(G_n) \geq \ell)) \;&\leq\; 2Prob\,(\omega(G_n) \geq \ell) \\ &=\; 2Prob\left(\bigcup_{L\in\binom{V_n}{\ell}} C_L\right) \;\leq\; \binom{n}{\ell} 2^{1-\binom{\ell}{2}}. \end{aligned}$$

Für $n \geq \ell \geq 2\log n$ und $n \geq 5$ gilt weiterhin

$$\begin{aligned} \binom{n}{\ell} 2^{1-\binom{\ell}{2}} \;&\overset{(A.3)}{\leq}\; \left(\frac{en}{\ell}\right)^{\ell} 2^{1-\frac{\ell-1}{2}\ell} \\ &=\; 2^{1+\ell(\log e+\log n-\log \ell-\ell/2+1/2)} \\ &\leq\; 2^{1-2\log n(-\log e+\log\log n+1-1/2)} \\ &\leq\; 2^{-\frac{1}{100}\log n\log\log n} \qquad < \qquad 1, \end{aligned}$$

wobei sich die vorletzte Ungleichung durch Einsetzen unter Verwendung von $n \geq 5$ leicht verifizieren läßt. Also gilt

$$\begin{aligned} &Prob\,(\omega(G_n) < 2\log n \text{ und } \alpha(G_n) < 2\log n) \\ &\quad =\; 1 - Prob\,(\omega(G_n) \geq 2\log n \text{ oder } \alpha(G_n) \geq 2\log n) > 0, \end{aligned}$$

und es gibt zu jedem $n \geq 5$ einen Graphen auf n Knoten, der weder eine stabile Menge noch eine Clique der Größe $2\log n$ enthält. Für $n = 3$ und $n = 4$ gilt die Behauptung ebenfalls. □

Anmerkungen und Übung

Genauer haben wir im Beweis zu Satz 13.1.3 sogar gezeigt, daß für große n die Unabhängigkeitszahl und die Cliquenzahl fast aller Graphen auf n Knoten höchstens $2\log n$ beträgt, da $2^{-\frac{1}{100}\log n \log\log n} \to 0$ für $n \to \infty$. Andererseits ist es bisher nicht gelungen, eine unendliche Graphenfamilie mit dieser Eigenschaft zu konstruieren. Die besten konstruktiven Schranken für Graphen mit kleiner Cliquen- und Unabhängigkeitszahl stammen von FRANKL und WILSON [FW81]. Sie bestimmten Graphen H_n auf n Knoten mit $\alpha(H_n) < k(n) := \exp\left((1+o(1))\sqrt{2\ln n \ln\ln n}\right)$ und $\omega(H_n) < k(n)$. Diese Funktion erfüllt $k(n) = o(n^\epsilon)$ für jedes $\epsilon > 0$ und $k(n) = \Omega((\log n)^M)$ für jedes $M > 0$.

Die kleinste Zahl n, so daß jeder Graph auf mindestens n Knoten eine k-Clique oder eine stabile Menge der Größe k enthält, heißt die RAMSEY-Zahl $R(k)$. In Satz 13.1.1 haben wir damit $R(k) \le 2^{2k-3}$ gezeigt. Satz 13.1.3 liefert dagegen eine untere Schranke von $R(k) > 2^{k/2}$.

Übung 13.1.4 a) *$R(k) \le 6$ oder auf jeder Party mit mindestens 6 Personen gibt es 3 Personen, die sich paarweise kennen, oder 3 Personen, die sich paarweise nicht kennen.* **b)** *$R(k) = 6$.*

Die Cliquenzahl fast aller Graphen. Der folgende Satz beinhaltet nicht nur eine untere Schranke für die Cliquenzahl und die Unabhängigkeitszahl, sondern zeigt sogar, daß sie fast sicher jeweils auf nur zwei Werte konzentriert sind. Wir formulieren den Satz nur für die Cliquenzahl. Da die Kantenwahrscheinlichkeiten 1/2 betragen, sind die Cliquenzahl und die Unabhängigkeitszahl gleich verteilt, und die folgende Aussage gilt auch für die Unabhängigkeitszahl.

Satz 13.1.5 (MATULA 1976; BOLLOBÀS, ERDŐS 1976) *Es gibt eine Funktion $\ell(n)$ mit*

$$\ell(n) \;=\; \lfloor 2\log n - 2\log\log n + 2\log e - \frac{3}{2} + o(1)\rfloor \qquad \textit{für } n \to \infty,$$

so daß $Prob\,(\omega(G_n) \in \{\ell(n), \ell(n)+1\}) \to 1$ *für* $n \to \infty$.

Beweis. Für eine natürliche Zahl r und einen Graphen H bezeichne $X_r(H)$ die Anzahl der Cliquen der Größe r in H. Dann gilt

$$E(X_r) \;=\; \sum_{L \in \binom{V}{r}} Prob\,(C_L) \;=\; \binom{n}{r} \cdot 2^{-\binom{r}{2}}. \tag{13.1}$$

Offenbar ist die Funktion $r \to E(X_r)$ für große r monoton fallend. Die Cliquenzahl eines Graphen H ist dann die größte natürliche Zahl r, für die $X_r(H) > 0$ gilt. Um diese Zahl für fast alle Graphen zu bestimmen, definiere $\ell = \ell(n)$ als die größte natürliche Zahl, die

$$E(X_\ell) \;\ge\; \sqrt{n} \tag{13.2}$$

erfüllt. Um ℓ abzuschätzen, sei

$$f(x) := \left(\frac{2^{\frac{-x+1}{2}} en}{x}\right)^x (2\pi n x)^{-1/2},$$

und sei $z := 2\log n - 2\log\log n + 2\log e - \frac{3}{2}$. Durch Einsetzen folgt leicht, daß für alle $\epsilon > 0$ gilt $f(z+\epsilon) = o(1)$ und $f(z-\epsilon) \to \infty$ für $n \to \infty$. Aus (A.4) folgt für $1 << r^2 << n$

$$E(X_r) = \binom{n}{r} \cdot 2^{-\binom{r}{2}} = (1+o(1)) f(r)\sqrt{n}.$$

Also ist das in (13.2) definierte ℓ bestimmt durch $\ell = \lfloor z + o(1) \rfloor$; das heißt

$$\ell = \lfloor 2\log n - 2\log\log n + 2\log e - \frac{3}{2} + o(1) \rfloor. \tag{13.3}$$

Aus (13.1) folgt

$$\frac{E(X_{r+1})}{E(X_r)} = \frac{n!(n-r)!r!}{(n-r-1)!(r+1)!n!} \cdot 2^{\binom{r}{2}-\binom{r+1}{2}} = \frac{n-r}{r+1} 2^{-r}. \tag{13.4}$$

Für hinreichend großes n folgt wegen $E(X_{\ell+1}) < \sqrt{n}$ (vgl. (13.2))

$$\begin{aligned} Prob\,(\omega(G_n) \geq \ell+2) \quad &= \quad Prob\,(X_{\ell+2} \geq 1) \quad \overset{(A.1.3)}{\leq} \quad E(X_{\ell+2}) \\ &\overset{(13.4)}{\leq} \quad \frac{n}{\log n} 2^{-\ell-1} E(X_{\ell+1}) \\ &\overset{(13.3)}{\leq} \quad \frac{n}{\log n} \frac{(\log n)^2}{n^2} n^{1/2} \quad = \quad o(1). \end{aligned}$$

Andererseits sind für hinreichend großes n die Ungleichungen $\frac{9}{7}\log n \leq \ell \leq 2\log n$ erfüllt. Die Tschebyscheffsche Ungleichung (vgl. Proposition A.1.4 b)) und Teil b) des folgenden Lemmas 13.1.6 implizieren dann

$$Prob\,(\omega(G_n) \leq \ell - 1) = Prob\,(X_\ell = 0) = \frac{Var(X_\ell)}{E(X_\ell)^2} = o(1).$$

Insgesamt gilt also $\ell \leq \omega(G_n) \leq \ell + 1$ mit Wahrscheinlichkeit $1 - o(1)$, und der Satz ist bewiesen. □

Es läßt sich sogar zeigen, daß für die meisten natürlichen Zahlen n die Cliquenzahl mit hoher Wahrscheinlichkeit auf nur einen Wert konzentriert ist (vgl. [Bol85], S. 253).

Lemma 13.1.6 *Für* $\frac{9}{7}\log n \leq \ell \leq 2\log n$ *gilt*

a) $\sum_{j=2}^{\ell-1} \binom{\ell}{j}\binom{n-\ell}{\ell-j} 2^{-\binom{\ell}{2}+\binom{j}{2}} \quad = \quad \mathcal{O}\left(\frac{(\log n)^4}{n^2} E(X_\ell)\right) + o(1)$

b) $Var(X_\ell) \quad = \quad o(E(X_\ell)^2).$

Beweis. a) Wir spalten die Summe in zwei Teilsummen auf. Einerseits ist (vgl. (13.1))

$$\begin{aligned} \frac{\sum_{j=2}^{\frac{3}{4}\ell} \binom{\ell}{j}\binom{n-\ell}{\ell-j} 2^{-\binom{\ell}{2}+\binom{j}{2}}}{E(X_\ell)} \quad &= \quad \sum_{j=2}^{\frac{3}{4}\ell} \binom{\ell}{j}\binom{n-\ell}{\ell-j}\binom{n}{\ell}^{-1} 2^{\binom{j}{2}} \\ &\leq \quad \sum_{j=2}^{\frac{3}{4}\ell} \left(\frac{\ell!}{(\ell-j)!}\right)^2 \frac{(n-\ell)!^2}{n!(n-2\ell+j)!} 2^{\binom{j}{2}}. \end{aligned}$$

Für hinreichend großes n und wegen $\ell \leq 2\log n$ ist dies kleiner als

$$\begin{aligned} \sum_{j=2}^{\frac{3}{4}\ell} \ell^{2j} \left(\frac{2}{n}\right)^j 2^{\frac{j-1}{2}j} \quad &\leq \quad \sum_{j=2}^{8} \left(\frac{8(\log n)^2}{n} 2^{\frac{8-1}{2}}\right)^j + \sum_{j=9}^{\frac{3}{4}\ell} \left(\frac{8(\log n)^2}{n} 2^{\frac{3}{8}\ell}\right)^j \\ &\leq \quad \mathcal{O}\left(\frac{(\log n)^4}{n^2}\right) + \sum_{j=9}^{\frac{3}{4}\ell} \left(8(\log n)^2 n^{-1/4}\right)^j \\ &= \quad \mathcal{O}\left(\frac{(\log n)^4}{n^2}\right). \end{aligned}$$

Wenn wir j durch $\ell - k$ ersetzen, läßt sich der zweite Teil der Summe abschätzen durch

$$\begin{aligned}
\sum_{j=\frac{3}{4}\ell+1}^{\ell-1} \binom{\ell}{j}\binom{n-\ell}{\ell-j}2^{-\binom{\ell}{2}+\binom{j}{2}} &\leq \sum_{k=1}^{\frac{1}{4}\ell-1} \binom{\ell}{\ell-k}\binom{n-\ell}{k}2^{-\binom{\ell}{2}+\binom{\ell-k}{2}} \\
&\leq \sum_{k=1}^{\frac{1}{4}\ell-1} (\ell n)^k 2^{(-\ell^2+\ell+\ell^2-2k\ell-\ell+k^2+k)/2} \\
&= \sum_{k=1}^{\frac{1}{4}\ell-1} (\ell n)^k 2^{(\frac{k+1}{2}-\ell)k} \\
&\leq \sum_{k=1}^{\frac{1}{4}\ell-1} \left(\ell n 2^{-\frac{7}{8}\ell}\right)^k \\
&\leq \sum_{k=1}^{\frac{1}{4}\ell-1} \left(\ell n^{-1/8}\right)^k \qquad = \qquad o(1),
\end{aligned}$$

wobei die vorletzte Ungleichung aus $\ell \geq \frac{9}{7}\log n$ folgt. Damit ist Teil a) von Lemma 13.1.6 bewiesen.

Zum Beweis von Teil b) berechnen wir zunächst das in der Varianz auftretende zweite Moment $E(X_\ell^2)$ der Zufallsvariablen X_ℓ. Die Anzahl geordneter Paare von Knotenmengen der Größe ℓ, die genau j gemeinsame Knoten haben, beträgt $\binom{n}{\ell}\binom{\ell}{j}\binom{n-\ell}{\ell-j}$, denn es gibt $\binom{n}{\ell}$ Wahlmöglichkeiten für die erste Knotenmenge der Größe ℓ, und die zweite Knotenmenge enthält j Knoten aus der ersten Knotenmenge ($\binom{\ell}{j}$ Möglichkeiten) und $\ell - j$ Knoten von den restlichen Knoten ($\binom{n-\ell}{\ell-j}$ Möglichkeiten). Ein solches Paar enthält $2\binom{\ell}{2} - \binom{j}{2}$ Kanten. Daher gilt

$$E(X_\ell^2) \quad = \quad \sum_{j=0}^{\ell} \binom{n}{\ell}\binom{\ell}{j}\binom{n-\ell}{\ell-j}2^{-2\binom{\ell}{2}+\binom{j}{2}}.$$

Daraus folgt

$$\begin{aligned}
Var(X_\ell) &= E(X_\ell^2) - E(X_\ell)^2 \\
&\overset{(13.1)}{=} E(X_\ell)\left(\sum_{j=0}^{\ell}\binom{\ell}{j}\binom{n-\ell}{\ell-j}2^{-\binom{\ell}{2}+\binom{j}{2}} - \binom{n}{\ell}2^{-\binom{\ell}{2}}\right) \\
&\overset{(A.5)}{=} E(X_\ell)\left(\sum_{j=0}^{\ell}\binom{\ell}{j}\binom{n-\ell}{\ell-j}2^{-\binom{\ell}{2}}\left(2^{\binom{j}{2}}-1\right)\right) \\
&\leq E(X_\ell)\left(\sum_{j=2}^{\ell-1}\binom{\ell}{j}\binom{n-\ell}{\ell-j}2^{-\binom{\ell}{2}+\binom{j}{2}}+1\right) \\
&\overset{(13.1.6a))}{=} E(X_\ell)\left(\mathcal{O}\left(\frac{(\log n)^4}{n^2}E(X_\ell)\right)+o(1)+1\right) \\
&\overset{(13.2)}{=} o(E(X_\ell)^2),
\end{aligned}$$

und das Lemma ist bewiesen. □

Greedy Clique. Der Greedy-Algorithmus für CLIQUE baut in einem Graphen $G_n \in \mathcal{G}_n$ sukzessive eine Clique C auf, indem er, beginnend mit $C := \{v_1\}$, nacheinander die Knoten $v_2, \ldots, v_n$ durchgeht und v_i der Clique C zufügt, falls v_i mit allen Knoten aus der bisherigen Clique C verbunden ist. Der folgende Satz zeigt zusammen mit Satz 13.1.5, daß der Greedy-Algorithmus für CLIQUE für fast alle Graphen die Approximationsgüte $2+o(1)$ hat. Bisher ist kein Algorithmus bekannt, der für ein $\epsilon > 0$ in fast allen Graphen G_n eine Clique der Größe $(1/2 - \epsilon)\omega(G_n)$ findet.

Satz 13.1.7 (GRIMMETT, MCDIARMID 1975) *Sei $\omega_g(G_n)$ die Größe der vom Greedy-Algorithmus für* CLIQUE *im Graphen G_n gefundenen Clique. Dann gilt für fast alle Graphen*

$$\omega_g(G_n) \sim \log n.$$

Beweis. Wir zeigen zunächst, daß

$$Prob\,(\omega_g(G_n) \geq \log n + 2(\log n)^{1/2}) \quad \leq \quad 2^{-2\log n + (\log n)^{1/2}}. \tag{13.5}$$

Sei $r := 2(\log n)^{1/2}$. Die Wahrscheinlichkeit dafür, daß ein fester Knoten v_j der i-te Knoten in der vom Greedy-Algorithmus bestimmten Clique ist, beträgt höchstens 2^{-i+1}, denn v_j muß zu allen Knoten einer $(i-1)$-elementigen Menge benachbart sein. Für eine feste Menge von Indizes $j_1 < \ldots < j_r$ ist die Wahrscheinlichkeit dafür, daß der Knoten v_{j_i} für $i = 1, \ldots, r$ der $(\log n + i)$-te Knoten in der vom Greedy-Algorithmus bestimmten Clique ist, daher höchstens

$$\prod_{i=1}^{r} 2^{-\log n - i + 1} = 2^{-r\log n - \frac{r(r-1)}{2}} = n^{-r} 2^{-\frac{r^2}{2} + \frac{r}{2}}.$$

Insgesamt gibt es $\binom{n}{r} \leq n^r$ verschiedene Möglichkeiten, die Indizes $j_1 \ldots j_r$ zu wählen. Also gilt

$$Prob\,(\omega_g(G_n) \geq \log n + r) \quad \leq \quad \binom{n}{r} n^{-r} 2^{-\frac{r^2}{2} + \frac{r}{2}} \quad \leq \quad 2^{-2\log n + (\log n)^{1/2}}$$

nach der Definition von r. Damit ist (13.5) gezeigt.

Zum Beweis der anderen Ungleichung sei X_j der Index des j-ten Knotens in der vom Greedy-Algorithmus bestimmten Clique und $X_j := n+1$ für $j > \omega_g(G_n)$. Setze $Y_j := X_{j+1} - X_j$. Falls für ein $j \leq \omega_g(G_n)$ gilt $Y_j \geq i$, dann darf keiner der $i-1$ Knoten $X_j + 1, \ldots, X_j + i - 1$ zu allen j Knoten aus der bisher gefundenen Clique adjazent sein. Die Wahrschienlichkeit dafür, daß ein Knoten zu j fest gewählten Knoten adjazent ist, beträgt 2^{-j}. Also gilt

$$Prob\,(Y_j \geq i) \quad = \quad (1 - 2^{-j})^{i-1}.$$

Sei nun $s := \log n - 2\log\log n$. Wegen $\sum_{j=1}^{\omega_g(G_n)} Y_j = n$ folgt aus $\omega_g(G_n) \leq s$, daß $Y_j(G_n) \geq \frac{n}{s}$ für ein $j \leq s$ gilt. Daraus folgt

$$\begin{aligned} Prob\,(\omega_g(G_n) \leq s) \quad &\leq \quad Prob\,(\bigcup_{j=1}^{s} \{Y_j(G_n) \geq \frac{n}{s}\}) \\ &\leq \quad \sum_{j=1}^{s} Prob\,(Y_j(G_n) \geq \frac{n}{s}) \end{aligned}$$

$$\begin{aligned} &\leq \sum_{j=1}^{s}(1-2^{-j})^{\frac{n}{s}-1} \\ &\leq s\cdot(1-2^{-s})^{\frac{n}{s}-1}. \end{aligned}$$

Dies ist wegen

$$1-x \leq e^{-x} \tag{13.6}$$

(, da die Fuktion $x \mapsto e^{-x}$ an der Stelle $x=0$ konvex ist) angewandt auf $x=2^{-s}$ und nach der Definition von s höchstens

$$\exp(\ln s - 2^{-s}(\frac{n}{s}-1)) \leq \exp\left(\ln\log n - \frac{(\log n)^2}{n}\cdot(\frac{n}{\log n - 2\log\log n}-1)\right) = o(1),$$

woraus zusammen mit (13.5) die Behauptung des Satzes folgt. □

13.2 Die chromatische Zahl

In Abschnitt 14.2 haben wir Graphen mit beschränkter Cliquenzahl betrachtet und gesehen, wie man mittels Abzählmethoden zeigt, daß ihre chromatische Zahl fast immer mit der Cliquenzahl übereinstimmt. Wir wollen nun die Beschränkung an die Cliquenzahl fallen lassen und die chromatische Zahl fast aller Graphen der Ordnung n studieren. Eine unserer ersten Abschätzungen für die chromatische Zahl war

$$\chi(G) \;\geq\; \frac{n}{\alpha(G)}.$$

Wir sahen darüberhinaus, daß diese untere Schranke im allgemeinen beliebig schlecht werden kann. Es wurde jedoch schon 1975 in [GM75] vermutet, daß sie für fast alle Graphen asymptotisch scharf ist. Der Beweis dieser Vermutung wurde erst im Jahr 1988 von Bollobàs [Bol88] erbracht unter Benutzung der Theorie der Martingale. Um den Beweis nachvollziehen zu können, benötigen wir einige Vorbereitungen. Zunächst führen wir Martingale ein.

Martingale. Sei Z eine reellwertige Zufallsvariable auf $\mathcal{G}_n$. Sei $e_1,\ldots,e_{\binom{n}{2}}$ eine Aufzählung der Kanten des K_n. Setze $E_i := \{e_1,\ldots,e_i\}$ und $E_0 := \emptyset$. Für einen Graphen $H \in \mathcal{G}_n$ sei $\mathcal{G}_n^i(H)$ die Klasse aller Graphen auf n Knoten, die mit H auf den ersten i Kanten übereinstimmen, also

$$\mathcal{G}_n^i(H) := \{G_n \in \mathcal{G}_n : E(G_n)\cap E_i = E(H)\cap E_i\}.$$

Dann definieren wir für jedes $i=1,\ldots,\binom{n}{2}$ eine Zufallsvariable Z^i, wobei $Z^i(H)$ für $H\in\mathcal{G}_n$ den Erwartungswert der Zufallsvariablen Z über alle Graphen aus $\mathcal{G}_n^i(H)$ angibt; sei also

$$Z^i(H) := E(Z(G_n)|G_n\in\mathcal{G}_n^i(H)) = \frac{1}{|\mathcal{G}_n^i(H)|}\sum_{G_n\in\mathcal{G}_n^i(H)} Z(G_n).$$

Für wachsendes i enthält $Z^i(H)$ also immer mehr Information über den Wert $Z(H)$. Während $Z^0(H)=E(Z)$ ist, gilt $Z^{\binom{n}{2}}(H)=Z(H)$. Dies entspricht der Tatsache, daß die

Klassen $\mathcal{G}_n^i(H)$ mit wachsendem i immer kleiner werden. Die $\mathcal{G}_n^i(H)$ lassen sich als Äquivalenzklassen der Äquivalenzrelationen $H \sim_i G :\Leftrightarrow G \in \mathcal{G}_n^i(H) \Leftrightarrow H \in \mathcal{G}_n^i(G)$ auffassen, und die Zufallsvariable Z^i ist auf jeder Äquivalenzklasse von $\sim_i$ konstant und entspricht gerade dem mittleren Wert von Z in dieser Äquivalenzklasse. Die Äquivalenzrelationen sind geschachtelt; das heißt, daß aus $H \sim_{i+1} G$ folgt, daß $H \sim_i G$. Die $\mathcal{G}_n^i(H)$ lassen sich folgendermaßen rekursiv definieren:

$$\mathcal{G}_n^i(H) \quad := \quad \begin{cases} \{H\} & \text{falls } i = \binom{n}{2}, \\ \mathcal{G}_n^{i+1}(H \cup \{e_{i+1}\}) \cup \mathcal{G}_n^{i+1}(H \backslash \{e_{i+1}\}) & \text{falls } i = 0, \ldots, \binom{n}{2} - 1. \end{cases}$$

Daraus folgt unmittelbar, weil die Äquivalenzklassen bezüglich $\sim_i$ alle gleich groß sind, daß

$$Z^i(G_n) = \frac{1}{2}\left(Z^{i+1}(G_n \cup \{e_{i+1}\}) + Z^{i+1}(G_n \backslash \{e_{i+1}\})\right). \tag{13.7}$$

Die Folge von Zufallsvariablen $(Z^i)_{i=0,\ldots,\binom{n}{2}}$ ist ein sogenanntes *Martingal*, das heißt, daß für jedes $a \in \mathbb{R}$ gilt, daß der Erwartungswert von Z^{i+1} über alle Graphen aus $\mathcal{G}_n$, für die $Z^i(G_n) = a$ gilt, gerade a beträgt. Symbolisch läßt sich dies schreiben als (vgl. A.1)

$$E(Z^{i+1}|Z^i) = Z^i \qquad \text{für alle } i = 0, \ldots, \binom{n}{2} - 1.$$

Für die oben definierten Z^i kann dies folgendermaßen nachgewiesen werden. Wegen $G_n \cup \{e_{i+1}\} \sim_i G_n \backslash \{e_{i+1}\}$ sind mit $Z^i(G_n) = a$ auch $Z^i(G_n \cup \{e_{i+1}\}) = Z^i(G_n \backslash \{e_{i+1}\}) = a$. Daraus folgt mit (13.7)

$$\begin{aligned} & E(Z^{i+1}(G_n)|Z^i(G_n) = a) \\ &= \frac{1}{|\{G_n \in \mathcal{G}_n : Z^i(G_n) = a\}|} \sum_{G_n \in \mathcal{G}_n : Z^i(G_n) = a} Z^{i+1}(G_n) \\ &= \frac{1}{|\{G_n \in \mathcal{G}_n : Z^i(G_n) = a\}|} \sum_{G_n \in \mathcal{G}_n : Z^i(G_n) = a} \frac{1}{2}\left(Z^{i+1}(G_n \cup \{e_{i+1}\}) + Z^{i+1}(G_n \backslash \{e_{i+1}\})\right) \\ &= \frac{1}{|\{G_n \in \mathcal{G}_n : Z^i(G_n) = a\}|} \sum_{G_n \in \mathcal{G}_n : Z^i(G_n) = a} Z^i(G_n) \\ &= a \end{aligned}$$

$(Z^i)_{i=0,\ldots,\binom{n}{2}}$ heißt das *Kantenaufdeckungsmartingal* zu Z, denn in jedem Schritt von i zu $i+1$ wird eine Kante mehr „aufgedeckt".

Die Bedeutung von Martingalen liegt darin, daß man mit ihrer Hilfe nachweisen kann, daß Zufallsvariablen unter bestimmten Voraussetzungen nur mit exponentiell kleiner Wahrscheinlichkeit weit von ihrem Erwartungswert abweichen können. Der folgende Satz macht dies präzise. In der oben beschriebenen Situation gilt $Z^r(H) = Z(H)$ und $Z^0(H) = E(Z)$ für eine Zufallsvariable Z.

Satz 13.2.1 (Azuma 1967) *Sei $Z^0, \ldots, Z^r$ ein Martingal auf $\mathcal{G}_n$ mit $|Z^{i+1}(H) - Z^i(H)| \leq 1$ für alle $H \in \mathcal{G}_n$ und $i = 0, \ldots, r-1$, und Z^0 sei konstant. Sei $\lambda > 0$ beliebig. Dann gilt*

$$Prob\,(|Z^r - Z^0| \geq \lambda\sqrt{r}) < 2e^{-\lambda^2/2}.$$

Beweis. Sei $Y^i(H) := Z^i(H) - Z^{i-1}(H)$ für $i = 1, \ldots, r$. Definiere $t := \lambda/\sqrt{r}$. Dann gilt wegen der Monotonie der Exponentialfunktion und nach der MARKOFFschen Ungleichung (vgl. Proposition A.1.3)

$$\begin{aligned} Prob\,(Z^r - Z^0 \geq \lambda\sqrt{r}) &= Prob\,(e^{t(Z^r - Z^0)} \geq e^{t\lambda\sqrt{r}}) \\ &\leq e^{-t\lambda\sqrt{r}} E(e^{t(Z^r - Z^0)}). \end{aligned} \tag{13.8}$$

Nach Proposition A.1.2 a) gilt

$$\begin{aligned} E(e^{t(Z^r - Z^0)}) &= E(e^{t(Z^{r-1} - Z^0)}\, e^{tY^r}) \\ &= E(e^{t(Z^{r-1} - Z^0)}\, E(e^{tY^r} | e^{t(Z^{r-1} - Z^0)})) \\ &= E(e^{t(Z^{r-1} - Z^0)}\, E(e^{tY^r} | Z^{r-1})), \end{aligned} \tag{13.9}$$

wobei die letzte Gleichung aus der Definition des bedingten Erwartungswertes folgt, denn wegen der Bijektivität der Exponentialfunktion und weil Z^0 konstant ist, ist für Graphen K und L $e^{t(Z^{r-1}(K) - Z^0(K))} = e^{t(Z^{r-1}(L) - Z^0(L))}$ äquivalent zu $Z^{r-1}(K) = Z^{r-1}(L)$.

Um $E(e^{tY^r} | Z^{r-1})$ abzuschätzen, definieren wir die Funktion

$$f(x) := \frac{1}{2}(e^t + e^{-t}) + \frac{1}{2}(e^t - e^{-t})x.$$

Dann gilt $e^{tx} \leq f(x)$ für $x \in [-1, 1]$, denn die Funktion $x \to e^{tx}$ ist konvex und liegt deshalb im Bereich $[-1, 1]$ unterhalb ihrer Sekante f durch die beiden Punkte -1 und 1. Da nach Voraussetzung $|Y^r| \leq 1$ ist, gilt also

$$E(e^{tY^r} | Z^{r-1}) \leq E(f(Y^r) | Z^{r-1}).$$

Wegen $E(Y^r | Z^{r-1}) = E(Z^r - Z^{r-1} | Z^{r-1}) = E(Z^r | Z^{r-1}) - Z^{r-1} = 0$ (vgl. Proposition A.1.2 b)) und der Linearität des bedingten Erwartungswertes folgt daraus

$$\begin{aligned} E(e^{tY^r} | Z^{r-1}) &\leq \frac{1}{2}(e^t + e^{-t}) + \frac{1}{2}(e^t - e^{-t}) E(Y^r | Z^{r-1}) \\ &= \frac{1}{2}(e^t + e^{-t}) \end{aligned}$$

Da für alle $x \in \mathbb{R}$ gilt $e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$, ist dies gleich

$$\frac{1}{2}\left(\sum_{k=0}^{\infty} \frac{t^k + (-t)^k}{k!}\right) = \frac{1}{2}\left(2\sum_{j=0}^{\infty} \frac{t^{2j}}{(2j)!}\right) \leq \sum_{j=0}^{\infty} \frac{t^{2j}}{2^j j!} = e^{\frac{t^2}{2}}.$$

Mit (13.9) folgt daraus, daß $E(e^{t(Z^r - Z^0)}) \leq E(e^{t(Z^{r-1} - Z^0)}) e^{\frac{t^2}{2}}$, womit sich per Induktion $E(e^{t(Z^r - Z^0)})$ durch $\prod_{i=1}^{r} e^{\frac{t^2}{2}} = e^{rt^2/2}$ abschätzen läßt. Mit (13.8) und der Definition von t folgt daraus

$$Prob\,(Z^r - Z^0 \geq \lambda\sqrt{r}) \leq e^{-t\lambda\sqrt{r} + \frac{rt^2}{2}} = e^{-\lambda^2/2}.$$

Genauso läßt sich $Prob\,(Z^r - Z^0 \leq -\lambda\sqrt{r})$ abschätzen, woraus die Behauptung des Satzes folgt. □

Als erste Anwendung von Satz 13.2.1 zeigen wir, daß die chromatische Zahl fast aller Graphen in einem kleinem Intervall liegt, das ungefähr die Länge $\sqrt{n}$ hat.

Satz 13.2.2 (SHAMIR, SPENCER 1987) *Es gilt*

$$Prob\left(|\chi(G_n) - E(\chi)| \geq \lambda\sqrt{n-1}\right) \quad < \quad 2e^{-\lambda^2/2}.$$

Beweis. Sei $v_1, \ldots v_n$ eine Aufzählung der Knoten des K_n und $V_i := \{v_1, \ldots v_i\}$. Sei die Zufallsvariable X^i für $H \in \mathcal{G}_n$ definiert durch

$$X^i(H) \quad := \quad E(\chi(G_n)|G_n[V_i] = H[V_i]).$$

Ähnlich wie oben läßt sich leicht nachweisen, daß $(X^i)_{i=1\ldots n}$ ein Martingal ist, das sogenannte Knotenaufdeckungsmartingal zur Funktion χ. Ferner folgt leicht[1] aus der Tatsache, daß sich die chromatische Zahl bei Hinzunahme eines Knotens höchstens um eins erhöht, daß für alle Graphen $H \in \mathcal{G}_n$ gilt $|X^{i+1}(H) - X^i(H)| \leq 1$ (für $i = 1, \ldots, n-1$). Also sind alle Voraussetzungen von Satz 13.2.1 erfüllt, und dieser impliziert direkt die Aussage. □

Die chromatische Zahl fast aller Graphen. Satz 13.2.2 macht keine Aussage über die Größenordnung der chromatischen Zahl fast aller Graphen. Diese wollen wir nun genauer bestimmen. Dazu erweist sich die Untersuchung der folgenden Zufallsvariablen als nützlich. Sei

$$Y_k(G) \text{ die maximale Anzahl kantendisjunkter } k\text{-Cliquen in } G.$$

In dem nächsten Lemma werden wir den Erwartungswert von Y_k nach unten abschätzen und im darauffolgenden Lemma unter Verwendung der Martingaltheorie eine Aussage über die Konzentration der Cliquenzahl zeigen. Da die Cliquenzahl und die Unabhängigkeitszahl die gleiche Verteilung haben, können wir dieses Lemma später benutzen, um viele unabhängige Mengen in einem zufälligen Graphen zu finden. Diese unabhängigen Mengen können wir jeweils zu einer Farbklasse zusammenfassen.

Lemma 13.2.3 *Es gibt eine Konstante $c > 0$, so daß das folgende gilt. Sei für eine natürliche Zahl n $\ell(n)$ die Funktion aus Satz 13.1.5 und $k = k(n) := \ell(n) - 2$. Dann ist*

$$E(Y_k) \quad \geq \quad \frac{cn^2}{(\log n)^4}.$$

Beweis. Zu einem Graphen G konstruieren wir folgendermaßen einen Graphen $R = R(G)$. Die Knoten von R seien die Kopien von K_k in G, und zwei (verschiedene) Knoten in R seien adjazent, wenn die dazugehörigen Kopien im Graphen G mindestens eine gemeinsame Kante haben. $|R|$, die Knotenzahl des Graphen R, ist gerade die im Beweis von Satz 13.1.5 definierte Zufallsvariable $X_k(G)$. Die Anzahl der Kanten von R bezeichnen wir wie gewohnt mit $m(R)$. Eine unabhängige Menge im Graphen R entspricht also gerade einer Menge von kantendisjunkten k-Cliquen in G, und es gilt $Y_k(G) = \alpha(R(G))$. Proposition 1.5.6 liefert die folgende Schranke für die Unabhängigkeitszahl von R

$$\alpha(R) \quad \geq \quad \frac{|R|}{\frac{2m(R)}{|R|} + 1} \quad = \quad \frac{|R|^2}{2m(R) + |R|}. \tag{13.10}$$

Wir zeigen zunächst, daß die Zufallsvariable $|R(G)|$ $(= X_k(G))$ um ihren Erwartungswert konzentriert ist. Für hinreichend großes n gilt $\frac{9}{7}\log n \leq k \leq 2\log n$ (vgl. (13.3)). Für

[1] Eine ähnliche Aussage wird später in Lemma 13.2.4 bewiesen.

große n können wir also Lemma 13.1.6 b) anwenden, und mit der TSCHEBYSCHEFFschen Ungleichung (Proposition A.1.4 a)) folgt

$$Prob\left(|X_k(G_n) - E(X_k)| > \frac{1}{2}E(X_k)\right) \quad \leq \quad \frac{4Var(X_k)}{E(X_k)^2} \quad = \quad o(1).$$

Also gilt fast sicher $\frac{1}{2}E(X_k) \leq |R| \leq \frac{3}{2}E(X_k)$, und es folgt aus (13.10), daß

$$E(\alpha(R)) \;\geq\; E\left(\frac{|R|^2}{2m(R)+|R|}\right) \;\geq\; \frac{1}{4}E(X_k)^2 E\left(\frac{1}{2(m(R))+2E(X_k)}\right).$$

Die Funktion $f : x \to \frac{1}{2x+2E(X_k)}$ ist auf $\mathbb{R}^+$ konvex, denn es gilt für $x > 0$
$f''(x) = \frac{8}{(2x+2E(X_k))^3} > 0$. Mit der JENSENschen Ungleichung (Proposition A.1.1 a)) folgt also

$$E(\alpha(R)) \quad \geq \quad \frac{\frac{1}{4}E(X_k)^2}{2E(m(R))+2E(X_k)} \quad = \quad \frac{1}{8}\left(\frac{E(m(R))}{E(X_k)^2} + \frac{1}{E(X_k)}\right)^{-1}. \qquad (13.11)$$

Nun schätzen wir $E(m(R))$ und $E(X_k)$ ab. Ähnliche Überlegungen wie im Beweis von Lemma 13.1.6 b) zeigen, daß sich $E(m(R))$ darstellen läßt als

$$\begin{aligned} E(m(R)) &= \sum_{j=2}^{k-1} \binom{n}{k}\binom{k}{j}\binom{n-k}{k-j} 2^{-2\binom{k}{2}+\binom{j}{2}} \\ &= E(X_k)\sum_{j=2}^{k-1} \binom{k}{j}\binom{n-k}{k-j} 2^{-\binom{k}{2}+\binom{j}{2}} \\ &\leq E(X_k)C\left(\frac{(\log n)^4}{n^2}E(X_k)+1\right), \qquad (13.12) \end{aligned}$$

wobei die letzte Ungleichung aus Lemma 13.1.6 a) folgt, wenn C als eine hinreichend große Konstante gewählt wird.

Für hinreichend großes n gilt nach der Definition von $k = \ell - 2$

$$E(X_k) \overset{(13.4)}{=} E(X_{k+2})\frac{(k+1)(k+2)}{(n-k)(n-k-1)}2^{2k+1} \overset{(13.2),(13.3)}{\geq} n^{\frac{1}{2}}\frac{1}{n^2}\frac{n^4}{(\log n)^4}2^{-5} \geq \frac{n^2}{(\log n)^4}.$$

Zusammen mit (13.11) und (13.12) folgt daraus, daß

$$\begin{aligned} E(\alpha(R)) &\geq \frac{1}{8}\left(C\frac{(\log n)^4}{n^2} + \frac{1+C}{E(X_k)}\right)^{-1} \\ &\geq \frac{1}{8}\left(\frac{(\log n)^4}{n^2}(2C+1)\right)^{-1} \\ &= (16C+8)^{-1}\,\frac{n^2}{(\log n)^4} \end{aligned}$$

und damit die Behauptung des Lemmas. □

Lemma 13.2.4 *Es gibt eine Konstante $b > 0$, so daß für k wie in Lemma 13.2.3 gilt*

$$Prob\,(\omega(G_n) < k) \quad \leq \quad 2e^{-b\frac{n^2}{(\log n)^8}}.$$

Beweis. Sei $(Y_k^i)_{i=0,\ldots,\binom{n}{2}}$ das zur Zufallsvariable Y_k gehörende Kantenaufdeckungsmartingal. Das heißt, daß für $i = 0, \ldots, \binom{n}{2}$ und $H \in \mathcal{G}_n$

$$Y_k^i(H) \quad = \quad E(Y_k(G_n)|G_n \in \mathcal{G}_n^i(H))$$

ist. Um Satz 13.2.1 anwenden zu können, müssen wir zeigen, daß $|Y^{i+1} - Y^i| \leq 1$ gilt. Sei also $H \in \mathcal{G}_n$, $i \in \{1, \ldots, \binom{n}{2}\}$ und zum Beispiel die Kante e_{i+1} in H enthalten. Dann gilt wegen (13.7)

$$\begin{aligned} |Y_k^i(H) - Y_k^{i+1}(H)| &= \left|\frac{1}{2}\left(Y_k^{i+1}(H) + Y_k^{i+1}(H\backslash\{e_{i+1}\})\right) - Y_k^{i+1}(H)\right| \\ &= \left|\frac{1}{2|\mathcal{G}_n^{i+1}(H)|}\left(\sum_{G_n \in \mathcal{G}_n^{i+1}(H)} Y_k(G_n\backslash\{e_{i+1}\}) - Y_k(G_n)\right)\right| \leq \frac{1}{2}, \end{aligned}$$

wobei die letzte Ungleichung aus der Tatsache folgt, daß sich $Y_k(G_n)$ und $Y_k(G_n\backslash\{e_{i+1}\})$ höchstens um eins unterscheiden können. An dieser Stelle wird wesentlich ausgenutzt, daß in Y_k *kantendisjunkte* k-Cliquen erfaßt werden.

Nach Satz 13.2.1 mit $\lambda := E(Y_k)/\sqrt{\binom{n}{2}}$ und wegen Lemma 13.2.3 gilt also

$$\begin{aligned} Prob\,(\omega(G_n) < k) &= Prob\,(Y_k = 0) \\ &\leq Prob\,(|Y_k - E(Y_k)| \geq E(Y_k)) \\ &\leq 2e^{-E(Y_k)^2/(2\binom{n}{2})} \\ &\leq 2e^{-\frac{1}{n^2}\left(\frac{cn^2}{(\log n)^4}\right)^2} \qquad \leq \qquad 2e^{-c^2\frac{n^2}{(\log n)^8}}, \end{aligned}$$

und die Behauptung des Lemmas folgt mit $b := c^2$. □

Nun haben wir alle Hilfsmittel beisammen, um den folgenden Satz beweisen zu können.

Satz 13.2.5 (Bollobás 1988) *Für fast jeden Graphen G_n der Ordnung n gilt*

$$\chi(G_n) \sim \frac{n}{2\log n}.$$

Beweis. Nach Satz 13.1.5 gilt $\alpha(G_n) \leq 2\log n$ für fast alle Graphen $G_n \in \mathcal{G}_n$. Daher ist fast sicher

$$\chi(G_n) \quad \geq \quad \frac{n}{\alpha(G_n)} \quad \geq \quad \frac{n}{2\log n}.$$

Zum Beweis der anderen Ungleichung weisen wir nach, daß wir mit hoher Wahrscheinlichkeit so lange nacheinander unabhängige Mengen der Größe $(2 + o(1))\log n$ von G_n abspalten können, bis nur noch $o(\frac{n}{\log n})$ Knoten übrig sind. Diese unabhängigen Mengen fassen wir zu je einer Farbklasse zusammen, und die restlichen Knoten färben wir einzeln. Bei diesem Verfahren benötigen wir insgesamt dann $(1 + o(1))\frac{n}{2\log n}$ Farben.

Sei nun W eine fest gewählte Untermenge der Knotenmenge der Kardinalität $r := \lfloor n/(\log n)^2 \rfloor$. Dann ist $G_n[W]$ gleichverteilt unter allen Graphen aus $\mathcal{G}_r$, falls G_n gleichverteilt unter allen Graphen aus $\mathcal{G}_n$ ist. Sei $k = k(r)$ wie in Lemma 13.2.3 definiert. Dann gilt (vgl. (13.3))

$$k \quad = \quad (2 + o(1)) \log r \quad = \quad (2 + o(1)) \log n. \tag{13.13}$$

Wegen $Prob\,(\alpha(H) < k) = Prob\,(\omega(H) < k)$ ist nach Lemma 13.2.4

$$Prob\,(\alpha(G_n[W]) < k) \quad \leq \quad 2e^{-b\frac{r^2}{(\log r)^8}}.$$

Insgesamt gibt es $\binom{n}{r} \leq n^r = 2^{r \log n}$ Untermengen der Knotenmenge der Kardinalität r. Daher gilt wegen der Definition von r

$$Prob\,(\text{Es gibt ein } W \text{ mit } |W| = r \text{ und } \alpha(G_n[W]) < k) \quad \leq \quad 2^{r \log n} 2e^{-b\frac{r^2}{(\log r)^8}} \; = \; o(1).$$

Also hat fast jeder Graph aus $\mathcal{G}_n$ die Eigenschaft, daß je r Knoten eine unabhängige Menge der Größe k enthalten. Einen solchen Graphen G_n können wir färben, indem wir nach und nach unabhängige Mengen der Größe k entfernen und alle Knoten dieser Menge mit der gleichen Farbe färben. Dies können wir so lange fortführen, bis nur noch r Knoten vorhanden sind, die wir dann alle einzeln färben. Dann gilt für fast alle G_n (vgl. (13.13))

$$\chi(G_n) \; \leq \; \left\lfloor \frac{n-r}{k} \right\rfloor + r \; \leq \; \frac{n}{k} + o\left(\frac{n}{\log n}\right) \; = \; (1 + o(1)) \frac{n}{2 \log n},$$

und der Satz ist bewiesen. □

Greedy Knotenfärbung. Der folgende Satz zeigt, daß auch der Greedy-Algorithmus für GRAPH COLORING (vgl. Abschnitt 5.2) für fast alle Graphen die Approximationsgüte $2 + o(1)$ hat. Auch für dieses Problem ist kein Algorithmus bekannt, der eine bessere asymptotische Approximationsgüte für fast alle Graphen hat.

Satz 13.2.6 (GRIMMETT, MCDIARMID 1975) *Sei $\chi_g(G_n)$ die Anzahl der Farben, die der Greedy-Algorithmus für* GRAPH COLORING *benötigt, um den Graphen G_n zu färben. Dann gilt fast sicher*

$$\chi_g(G_n) \sim \frac{n}{\log n}.$$

Beweis. Für $k \geq 1$ sei C_k die k-te Farbklasse der vom Greedy-Algorithmus erzeugten Färbung. Die Farbklassen des Greedy-Algorithmus lassen sich als vom Greedy-Algorithmus für INDEPENDENT-SET konstruierte unabhängige Mengen auffassen. Daher folgt genauso wie im Beweis von (13.5) in Satz 13.1.7, daß

$$Prob\,(|C_k| \geq \log n + 2(\log n)^{1/2}) \quad \leq \quad 2^{-2 \log n + (\log n)^{1/2}}.$$

Also ist mit Wahrscheinlichkeit $1 - n2^{-2 \log n + (\log n)^{1/2}} = 1 - 2^{-\log n + (\log n)^{1/2}} = 1 - o(1)$ jede der maximal n Farbklassen kleiner als $\log n + 2(\log n)^{1/2}$. Daher benötigt der Greedy-Algorithmus mit Wahrscheinlichkeit $1 - o(1)$ mindestens

$$\frac{n}{\log n + 2(\log n)^{1/2}} \quad = \quad (1 - o(1)) \frac{n}{\log n}$$

Farben.

Zum Beweis der anderen Ungleichung sei für $1 \leq j,k \leq n$ B_j^k das Ereignis, daß der j-te Knoten mit der Farbe k gefärbt wird, und $A_j^k = \bigcup_{i=1}^{j} B_i^k$ das Ereignis, daß auf den ersten j Knoten mindestens k Farben benötigt werden. Dann ist $A_n^k = \{G_n : \chi_g(G_n) \geq k\}$. Wir sind also an einer oberen Schranke für $Prob\,(A_n^k)$ mit $k \sim \log n$ interessiert.

Dazu schätzen wir zunächst die Wahrscheinlichkeiten $Prob\,(B_{j+1}^{k+1}|A_j^k)$ ab. Von den ersten j Knoten seien jeweils j_i Knoten mit der Farbe i gefärbt für $i = 1,\ldots,k$. Wenn der $(j+1)$-te Knoten die Farbe $k+1$ erhält, muß er für jedes $i = 1\ldots k$ mit mindestens einem Knoten der Farbe i verbunden sein. Dieses Ereignis hat die Wahrscheinlichkeit $1-2^{-j_i}$. Daraus folgt

$$Prob\,(B_{j+1}^{k+1}|A_j^k) = \prod_{i=1}^{k}(1-2^{-j_i}). \tag{13.14}$$

Um dies abzuschätzen, betrachten wir die Funktion $f : x \mapsto -\log(1-2^{-x})$. Diese ist für $x > 0$ konvex, da $f'(x) = \frac{-2^{-x}}{1-2^{-x}}$ und $f''(x) = \frac{(\ln 2)2^{-x}}{(1-2^{-x})^2} > 0$. Wegen der JENSENschen-Ungleichung (Proposition A.1.1 b)) gilt dann also

$$\sum_{i=1}^{k} f(j_i) \geq kf\left(\frac{\sum_{i=1}^{k} j_i}{k}\right) \geq kf\left(\frac{j}{k}\right),$$

wobei die zweite Ungleichung aus $\sum_{i=1}^{k} j_i \leq j$ und der Tatsache folgt, daß f monoton fallend ist. Wegen (13.14) gilt dann

$$\begin{aligned} Prob\,(B_{j+1}^{k+1}|A_j^k) &= 2^{-\sum_{i=1}^{k} f(j_i)} \leq 2^{-kf(\frac{j}{k})} \\ &= (1-2^{-j/k})^k \overset{(13.6)}{\leq} \exp(-2^{-j/k}k) \leq \exp(-2^{-n/k}k). \end{aligned} \tag{13.15}$$

Wenn der $(j+1)$te Knoten mit Farbe $k+1$ gefärbt wird, dann muß mindestens einer der ersten j Knoten mit der Farbe k gefärbt worden sein. Daher gilt $B_{j+1}^{k+1} \subseteq A_j^k \subseteq A_n^k$. Daraus folgt

$$Prob\,(B_{j+1}^{k+1}|A_n^k) = \frac{Prob\,(B_{j+1}^{k+1} \cap A_n^k)}{Prob\,(A_n^k)} = \frac{Prob\,(B_{j+1}^{k+1})}{Prob\,(A_n^k)} \leq \frac{Prob\,(B_{j+1}^{k+1})}{Prob\,(A_j^k)} = Prob\,(B_{j+1}^{k+1}|A_j^k).$$

Wegen $A_n^{k+1} \subseteq A_n^k$ folgt daraus zusammen mit (13.15)

$$Prob\,(A_n^{k+1}) \leq Prob\,(A_n^{k+1}|A_n^k) \leq \sum_{j=k+1}^{n} Prob\,(B_j^{k+1}|A_n^k) \leq n \cdot \exp(-2^{-n/k}k).$$

Sei nun $k := \frac{n}{\log n(1-\frac{3\log\log n}{\log n})}$. Dann gilt

$$\begin{aligned} Prob\,(\chi_g(G_n) > k) &= Prob\,(A_n^{k+1}) \\ &\leq n \cdot \exp(-2^{-n/k}k) \\ &= \exp\left(\ln n - 2^{-\log n(1-\frac{3\log\log n}{\log n})} \cdot \frac{n}{\log n(1-\frac{3\log\log n}{\log n})}\right) \end{aligned}$$

$$= \exp\left(\ln n - \frac{(\log n)^3}{n} \cdot \frac{n}{\log n(1-o(1))}\right)$$
$$= \exp\left(\ln n - \frac{(\log n)^2}{1-o(1)}\right) \qquad = \qquad o(1).$$

Also gilt fast sicher

$$\chi_g(G_n) \leq k = (1+o(1))\frac{n}{\log n}.$$ □

Kapitel 14

Ein Blick in die extremale und asymptotische Graphentheorie

14.1 Die Sätze von Turán und von Erdős und Stone

In diesem Abschnitt beschäftigen wir uns mit dem Problem die maximale Anzahl von Kanten zu bestimmen, die ein Graph auf $n \geq \min\{|H| : H \in \mathcal{H}\}$ Knoten haben kann, ohne einen Subgraphen $H \in \mathcal{H}$ aus einer Graphenfamilie $\mathcal{H}$ zu enthalten. Diese maximale Anzahl von Kanten werde mit $ex_n(\mathcal{H})$ bezeichnet. Da der vollständige Graph K_n jeden Subgraph auf höchstens n Knoten als Subgraph enthält, gilt offenbar $ex_n(\mathcal{H}) \leq \binom{n}{2}$, und $ex_n(\mathcal{H})$ ist wohldefiniert.

Beispiel. In Korollar 1.1.27 haben wir gezeigt, daß ein Graph mit mehr als $n-1$ Kanten einen Kreis enthält, oder anders ausgedrückt, $ex_n(\{C_\ell\}_{\ell \geq 3}) \leq n-1$. Da andererseits Bäume kreisfrei mit $m = n-1$ Kanten sind, gilt hier sogar Gleichheit. □

Graphen (wie die Bäume in obigem Beispiel) mit $m = ex(n, \mathcal{H})$ vielen Kanten heißen die *extremalen Graphen* für die verbotene Subgraphenfamilie $\mathcal{H}$. Wir wollen in diesem Kapitel Schranken für $ex_n(\mathcal{H})$ herleiten, wenn wir das Dreieck K_3 oder allgemeiner Cliquen K_p, $p \geq 3$, oder sogar beliebige Graphenfamilien $\mathcal{H}$ als Subgraphen verbieten.

Als Einstieg bestimmen wir die maximale Anzahl von Kanten in einem dreiecksfreien Graphen (ein Graph G heißt dreiecksfrei genau dann, wenn $\omega(G) \leq 2$). Vollständig bipartite Graphen mit gleichgroßen Bipartitionsmengen sind natürlich dreiecksfrei und haben offenbar $\frac{n^2}{4}$ viele Kanten. Es gilt nun:

Satz 14.1.1 (Mantel 1907) *Die maximale Anzahl Kanten in einem dreiecksfreien Graphen ist* $ex_n(K_3) = \left\lfloor \frac{n^2}{4} \right\rfloor$. *Die extremalen Graphen sind bipartit.*

Beweis. Sei G ein dreiecksfreier Graph und C eine minimale Knotenüberdeckung von G, d.h. $|C| = \tau(G)$. Nach Definition inzidieren also alle Kanten von G mit Knoten aus C. Da zudem die Nachbarn eines Knotens aufgrund der Dreiecksfreiheit eine stabile Menge bilden, folgt

$$|E| \leq \sum_{v \in C} d(v) \leq \tau(G) \cdot \alpha(G) \leq \left(\frac{\tau(G) + \alpha(G)}{2} \right)^2 \overset{4.1.9}{=} \frac{n^2}{4}.$$

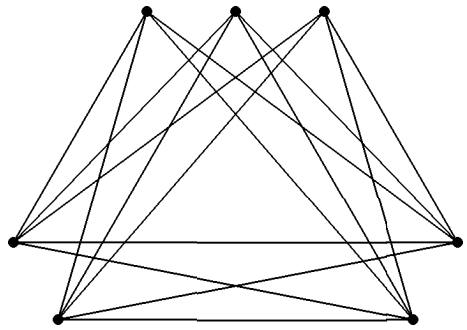


Abbildung 14.1: Der TURÁN-Graph $T_7(3)$

Die letzte Ungleichung gilt dabei wegen der Beziehung $\sqrt{ab} \leq \frac{a+b}{2}$ zwischen geometrischem und arithmetischem Mittel. An der ersten Ungleichung ist zu erkennen, daß diese Schranke nur für bipartite Graphen scharf ist: ein ungerader Kreis enthält zwei benachbarte $u, v \in C$. Die Kante $\{u, v\}$ wird also auf der rechten Seite der Ungleichung doppelt gezählt, woraus $|E| \leq \frac{n^2}{4} - 1$. □

Übung 14.1.2 *Ein Graph mit mehr als* $\frac{1}{2}n\sqrt{n-1}$ *Kanten hat Taillenweite* ≤ 4 *(vergleiche mit* C_5*).*

TURÁN verallgemeinerte den Satz von MANTEL auf K_{p+1}-freie Graphen (siehe die Übersetzung in [Tur54]). Wieviele Kanten kann ein K_{p+1}-freier Graph haben? Ein p-partiter Graph enthält sicher keinen K_{p+1}. Ein vollständig p-partiter Graph hat maximal viele Kanten, wenn die p Klassen der Partition von V alle möglichst gleich groß sind, d.h. entweder $\lfloor \frac{n}{p} \rfloor$ oder $\lceil \frac{n}{p} \rceil$ Knoten enthalten. Denn angenommen, eine Klasse hätte n_1 viele Knoten und eine andere Klasse $n_2 \geq n_1 + 2$ viele. Dann enthielte der vollständig p-partite Graph, den man erhält, wenn man einen Knoten der zweiten Klasse in die erste transferiert, mindestens $(n_2 - 1)(n_1 + 1) - n_1 n_2 = n_2 - (n_1 + 1) \geq 1$ Kanten mehr. Der vollständig p-partite Graph auf n Knoten mit p möglichst gleichgroßen Partitionsklassen heißt der TURÁN-Graph $T_n(p)$. Seine Kantenzahl wird mit $t_n(p)$ notiert. Offenbar gilt

$$t_n(p) = \binom{n}{2} - p\binom{n/p}{2} + \mathcal{O}(n) \tag{14.1}$$

$$= \frac{1}{2}\left\{n^2 - n - p\frac{n}{p}(\frac{n}{p} - 1)\right\} + \mathcal{O}(n) \tag{14.2}$$

$$= (1 - \frac{1}{p})\frac{n^2}{2} + \mathcal{O}(n) \tag{14.3}$$

Zum Vergleich gilt nach dem Satz von MANTEL beispielsweise $t_n(2) = \lfloor n^2/4 \rfloor$. Der Satz von TURÁN besagt nun, daß in der trivialen Ungleichung $ex_n(K_{p+1}) \geq t_n(p)$ tatsächlich Gleichheit gilt.

Satz 14.1.3 (TURÁN 1941) *Für jedes* $p \geq 1$ *gilt:*

$$ex_n(K_{p+1}) = t_n(p),$$

und der Graph $T_n(p)$ *ist der eindeutige extremale* K_{p+1}*-freie Graph auf* n *Knoten.*

Beweis (vgl. auch Übung 10.9.20). Zum Beweis von „$\leq$“ zeigen wir, daß die Gradsequenz eines jeden K_{p+1}-freien Graphen $G = (V, E)$ durch die Gradsequenz eines p-partiten Graphen $H = (V, E')$ auf derselben Knotenmenge majorisiert wird, d.h. es gilt

$$d_G(v) \;\leq\; d_H(v) \qquad \forall v \in V. \tag{14.4}$$

Wir induzieren nach p. Da ein 1-partiter Graph leer ist, gilt im Fall $p = 1$ trivialerweise $ex_n(K_2) = 0 = t_1(n)$, und der Graph $T_1(n)$ ist offensichtlich der eindeutige extremale Graph.

Sei $G = (V, E)$ also K_{p+1}-frei, $p \geq 2$. Sei $w_0 \in V$ ein Knoten maximalen Grades in G. Setze $Z := \Gamma(w_0)$ und $W := V \setminus Z$. Nach Annahme an G ist $G[Z]$ K_p-frei. Nach Induktionsannahme können wir daher $G[Z]$ durch einen $(p-1)$-partiten Graphen H_Z ersetzen, so daß $d_{G[Z]}(z) \leq d_{H_Z}(z)$ für alle $z \in Z$. Um nun einen p-partiten Graphen H auf V zu erhalten, füge W als stabile Menge zu H_Z hinzu und verbinde die Knoten aus W vollständig mit denen von H_Z.
Der Graph H hat die gewünschten Eigenschaften: für alle $w \in W$ gilt $d_G(w) \leq d_G(w_0) = d_H(w_0) = d_H(w)$ und für alle $z \in Z$ gilt $d_G(z) \leq d_{G[Z]}(z) + |V - Z| \leq d_{H_Z}(z) + n - |Z| = d_H(z)$.

Zur Eindeutigkeit des $T_n(p)$. Falls der obige Graph G ein extremaler K_{p+1}-freier Graph ist, so gilt $d_G(v) = d_H(v)$ für alle $v \in V$. Da jeder Knoten $z \in Z$ im Graphen H maximal viele Nachbarn in $V - Z$ hat, folgt $d_{H[Z]}(z) = d_{G[Z]}(z)$ für alle $z \in Z$. $G[Z]$ ist also ein extremaler K_p-freier Graph, und nach Induktionsannahme $(p-1)$-partit. Da nun aber wegen $d_G(z) = d_H(z)$ für alle $z \in Z$ auch

$$|\langle W, Z\rangle_G| = |W||Z| = |\langle W, Z\rangle_H|,$$

ist $G[W]$ schon der leere Graph, d.h. G ist p-partit. □

Der Satz von TURÁN markiert den Startpunkt für die sogenannte extremale Graphentheorie, die sich seitdem rasch entwickelt und eine Reihe tiefliegender Sätze hervorgebracht hat. Wir wollen den Faden noch ein wenig weiter spinnen. Für die meisten Graphen H erscheint das Problem, $ex_n(H)$ zu bestimmen, äußerst schwierig. Als ersten Schritt in diese Richtung hat man deshalb versucht, zumindest die Größenordnung dieser Funktion abzuschätzen. Es stellte sich heraus, daß dieses Problem eine ebenso einfache wie reizvolle Lösung hat. Die folgenden Ergebnisse von ERDŐS, STONE und SIMONOVITS zeigen, daß die Größenordnung von $ex_n(H)$ nur von der chromatischen Zahl des Graphen H abhängt. Wir behandeln zunächst den Fall, daß H der vollständige r-partite Graph mit je t Knoten in jeder Partitionsklasse - der $K_r(t)$ - ist.

Satz 14.1.4 (ERDŐS, STONE 1946) *Für ganze Zahlen $r \geq 2$ und $t \geq 1$ gilt*

$$ex_n(K_r(t)) \;=\; (1 - \frac{1}{r-1})\frac{n^2}{2} + o(n^2) \qquad (n \to \infty).$$

ERDŐS and STONE in fact proved a slightly more general result. Namely, they showed that for every $\epsilon > 0$ and for sufficiently large n every graph on n vertices with $(1 - \frac{1}{r-1} + \epsilon)\frac{n^2}{2}$ edges contains a $K_r((\ln^{(r-1)} n)^{1/2})$ subgraph. This was later generalized, first by BOLLOBÁS, ERDŐS and SIMONOVITS (1976) and then by CHVÁTAL and SZEMERÉDI (1981,1983), who showed that every such graph contains in fact already a $K_r(\log n/(500 \log \epsilon^{-1}))$ subgraph.

D.h. wenn man den extremalen K_r-freien Graphen nur ϵn^2 viele Kanten mehr „spendiert" für ein $\epsilon > 0$, so findet man in diesen Graphen schon einen $K_r(t)$ mit $t = \Omega(\log n)$ für $n \to \infty$. Dieses Resultat ist bis auf einen konstanten Faktor bestmöglich [BE73].

Dem Beweis von Satz 14.1.4 schicken wir zwei Lemmata voraus.

Lemma 14.1.5 *For every graph H, $ex_n(H)/\binom{n}{2}$ is decreasing as $n \to \infty$.*

Proof. Choose $k \leq n$ arbitrarily and let $G = (V_n, E)$ be a H-free graph on n vertices with $|E| = ex_n(H)$ many edges. Consider all $\binom{n}{k}$ induced subgraphs $G_1, \ldots, G_{\binom{n}{k}}$ of G on k vertices. Then each edge of G belongs to exactly $\binom{n-2}{k-2}$ of these subgraphs. Hence,

$$\binom{n-2}{k-2} \cdot ex_n(H) \;=\; \sum_{i=1}^{\binom{n}{k}} |E(G_i)| \;\leq\; \binom{n}{k} \cdot ex_k(H). \qquad \square$$

Corollary 14.1.6 *For every graph H, $\lim_{n\to\infty} ex_n(H)/n^2$ exists.* $\square$

Lemma 14.1.7 (KŐVARI, SÓS, TURÁN 1954) *Für $r, s \in \mathbb{N}$ gilt*

$$ex_n(K_{r,s}) \leq \frac{1}{2}(s-1)^{\frac{1}{r}} n^{2-\frac{1}{r}} + \mathcal{O}(n).$$

Proof. Let $G = (V_n, E)$ be a graph without $K_{r,s}$-subgraph. Then the number of $K_{1,r}$-subgraphs is bounded from above by $(s-1)\binom{n}{r}$. As on the other hand the number of $K_{1,r}$-subgraphs is given by $\sum_{i=1}^{n} \binom{d(v_i)}{r}$. This shows that

$$(s-1)\binom{n}{r} \;\geq\; \sum_{i=1}^{n} \binom{d(v_i)}{r} \;\geq\; n \cdot \binom{2|E|/n}{r},$$

from which the claimed bound follows by straightforward calculations. $\square$

Proof of Theorem 14.1.4. We proceed by induction on r. For $r = 2$ the theorem follows from 14.1.7. So assume now that the theorem is true for $r-1$ and all t and consider an arbitrary but fixed $t \in \mathbb{N}$. Define a constant $0 \leq c \leq 1$ by $1 - \frac{1}{r-1} + c = \lim_{n\to\infty} 2ex_n(K_r(t))/n^2$ (cf. 14.1.6). If $c = 0$ there is nothing to show, so assume $c > 0$. Let $q = 2t^2$ and choose $n_0 \geq 4q^{rt}$ sufficiently large so that $ex_n(K_{r-1}(q)) \leq (1 - \frac{1}{r-1})\frac{n^2}{2}$ for all $n \geq n_0/2$ (the existence of such an n_0 follows from the induction hypothesis) and

$$\left(1 - \frac{1}{r-1} + c(1 - \frac{1}{t(r-1)})\right) \cdot \frac{n^2}{2} < ex_n(K_r(t)) < \left(1 - \frac{1}{r-1} + c(1 + \frac{1}{t(r-1)})\right) \cdot \frac{n^2}{2}$$

for all $n \geq n_0/2$.

Let $G = (V_n, E)$ be a $K_r(t)$-free graph on $n \geq n_0$ vertices with $|E| \geq (1 - \frac{1}{r-1} + c(1 - \frac{1}{2t})) \cdot \frac{n^2}{2}$ many edges. By the choice of n_0 we know that G contains at least one $K_{r-1}(q)$-subgraph. Fix one such copy, say K, and let

$$X = \{v \in V_n \setminus K \mid |\Gamma(v) \cap K| \geq (r - 2 + \frac{1}{2t})q\}.$$

We claim that $|X| < q^{(r-1)t}$. To see this we count the $K_{r-1}(t)$-subgraphs of K in two ways. First, every vertex in X is connected to at least one such subgraph. Second, every such

subgraph can be connected to at most $(t-1)$ vertices in X. Therefore, $|X| \leq (t-1)\binom{q}{t}^{r-1} \leq q^{(r-1)t}$.

We now remove K and all adjacent edges from G to obtain the graph $G^{(1)}$. Obviously, $|V(G^{(1)})| = n-(r-1)q$ and

$$\begin{aligned} |E(G^{(1)})| &\geq |E| - \binom{(r-1)q}{2} - |X|(r-1)q - (n-(r-1)q-|X|)(r-2+\frac{1}{2t})q \\ &\geq |E| - (r-2+\frac{1}{t})qn. \end{aligned}$$

In particular, $G^{(1)}$ is at least as dense as G, so $G^{(1)}$ also contains a $K_{r-1}(q)$-subgraph. By removing such a copy we obtain the graph $G^{(2)}$, and so on. Repeating this procedure $cn\,/\,(r-1)q$ many times, we obtain a graph $G' \subseteq G$ so that

$$|V(G')| = (1-c)n$$

and

$$\begin{aligned} |E(G')| &\geq |E| - \left(1 - \frac{1}{r-1} - \frac{1}{t(r-1)}\right) \cdot cn^2 \\ &\geq \left(1 - \frac{1}{r-1} + c(1 - \frac{1}{t(r-1)})\right) \cdot \frac{n^2}{2} - \left(1 - \frac{1}{r-1} - \frac{1}{t(r-1)}\right) \cdot cn^2 \\ &\geq (1 - \frac{1}{r-1}) \cdot \frac{(1-c)^2n^2}{2} + \left(1 + \frac{1}{t(r-1)} - c\right) \cdot \frac{cn^2}{2} \\ &= (1 - \frac{1}{r-1}) \cdot \frac{[(1-c)n]^2}{2} + \frac{1 + \frac{1}{t(r-1)(1-c)}}{1-c} \cdot \frac{c \cdot [(1-c)n]^2}{2}. \end{aligned}$$

By choice of n_0 this implies that $\frac{1+\frac{1}{t(r-1)(1-c)}}{1-c} \leq 1 + \frac{1}{t(r-1)}$, which can only hold for $c > 1$. As by definition $c \leq \frac{1}{r-1}$ this is the desired contradiction. □

Ähnlich zur Situation, wo $\mathcal{H}$ die Menge der ungeraden Kreise ist, hat es asymptotisch dieselbe Wirkung, eine Menge von (nicht-leeren) Graphen auszuschließen, wie einen Graphen $H \in \mathcal{H}$ mit der kleinsten chromatischen Zahl auszuschließen.

Korollar 14.1.8 (ERDŐS, SIMONOVITS 1966) *Sei $\mathcal{H}$ eine endliche Menge von nicht-leeren Graphen und $c := \min\{\chi(H) : H \in \mathcal{H}\}$. Dann gilt:*

$$ex_n(\mathcal{H}) = (1 - \frac{1}{c-1})\frac{n^2}{2} + o(n^2) \qquad (n \to \infty). \tag{14.5}$$

Beweis. Für einen Graphen $H = argmin\,\{\chi(H) : H \in \mathcal{H}\}$ gilt offenbar $ex_n(\mathcal{H}) \leq ex_n(H)$. Da jeder H-freie Graph insbesondere $K_{\chi(H)}(|V(H)|)$-frei ist, liefert der Satz 14.1.4 von ERDŐS und STONE wiederum die obere Schranke

$$ex_n(H) \leq ex_n(K_{\chi(H)}(|V(H)|)) = (1 - \frac{1}{\chi(H)-1})\,\frac{n^2}{2} + o(n^2).$$

Für die untere Schranke beachte, daß der $(c-1)$-partite TURÁN-Graph $T_n(c-1)$ sicherlich $\mathcal{H}$-frei ist, so daß

$$ex_n(\mathcal{H}) \geq t_n(c-1) = (1 - \frac{1}{c-1})\frac{n^2}{2} + \mathcal{O}(n).$$ □

14.2 Fast alle dreiecksfreien Graphen sind bipartit

Wir kommen nun zu asymptotischen Strukturaussagen für Graphenklassen. ERDŐS, KLEITMAN und ROTHSCHILD gelang es 1976, die Struktur eines typischen dreiecksfreien Graphen zu bestimmen, d.h. eines dreiecksfreien Graphen, der gleichverteilt zufällig aus der Menge der dreiecksfreien Graphen ausgewählt (gezogen) wird.

Während wir in Abschnitt 5.3 dreiecksfreie Graphen mit beliebig hoher chromatischer Zahl konstruiert haben, zeigt sich hier, daß ein dreiecksfreier Graph aber in der Regel bipartit, also zweifärbbar, ist. D.h. wenn man den trivialen Grund dafür, daß ein Graph nicht zweifärbbar ist, nämlich den, daß er eine Clique der Größe mindestens 3 besitzt, ausschließt, dann ist er immerhin schon fast immer zweifärbbar. Dreiecksfreie Graphen hoher chromatischer Zahl sind also recht selten. Anders ausgedrückt: da bipartite Graphen trivialerweise dreiecksfrei sind, besagt das folgende Resultat, daß die Menge der bipartiten Graphen in der Menge der dreiecksfreien Graphen dicht liegt.

Theorem 14.2.1 (ERDŐS, KLEITMAN, ROTHSCHILD 1976) *Almost all triangle-free graphs are bipartite, that is*

$$\text{Prob}\ [G \in \mathcal{C}ol_n(2) |\ G \in \mathcal{F}orb_n(K_3)] \ = \ 1 - o(1) \qquad (n \to \infty).$$

Proof. While this result is nice and clear-cut, to prove it it was necessary to substitute the probabilistic methods known from random graph theory by subtle enumeration techniques. The proof of Theorem 14.2.1 will consist of three parts:

I. Define sets $\mathcal{A}(n)$, $\mathcal{B}(n)$ and $\mathcal{C}(n)$ such that

$$\mathcal{F}orb_n(K_3) \subseteq \mathcal{C}ol_n(2) \cup \mathcal{A}(n) \cup \mathcal{B}(n) \cup \mathcal{C}(n).$$

II. Bound the cardinalities of the sets $\mathcal{A}(n)$, $\mathcal{B}(n)$ and $\mathcal{C}(n)$ in terms of those of $\mathcal{F}orb_{n-x}(K_3)$ for appropriate $x \in \mathbb{N}$.

III. Show inductively that each of the sets $\mathcal{A}(n)$, $\mathcal{B}(n)$ and $\mathcal{C}(n)$ is negligible compared to $\mathcal{C}ol_n(2)$.

Figure 14.2 outlines the desired partition of $\mathcal{F}orb_n(K_3)$ and indicates how the sets $\mathcal{A}(n)$, $\mathcal{B}(n)$ and $\mathcal{C}(n)$ are defined. Note however, that the drawing is not according to scale.

Part I. The three sets $\mathcal{A}(n)$, $\mathcal{B}(n)$ and $\mathcal{C}(n)$ will reflect properties which hold almost surely in a (random) bipartite graph on n vertices. As we shall see in the proof of III, the color classes in a random bipartite graph are of approximately the same size. For the time being it therefore suffices to view a random bipartite graph as a bipartite graph with color classes as equal as possible, in which the edges are inserted randomly with probability 1/2. It should be intuitively clear that with very high probability in such a graph every vertex has degree at least, say, $\log n$, and that every set of size $\log n$ has at least, say, $(\frac{1}{2} - \frac{1}{1000})n$ neighbors. The definition of $\mathcal{A}(n)$ and $\mathcal{B}(n)$ captures these two properties, while the definition of $\mathcal{C}(n)$ reflects the property that in bipartite graphs all edges $\{x, y\}$ satisfy $\Gamma(\Gamma(x)) \cap \Gamma(\Gamma(y)) = \emptyset$. Let

$\mathcal{A}(n)$ denote the set of all graphs in $\mathcal{F}orb_n(K_3)$ which contain a vertex v such that $|\Gamma(v)| \leq \log n$,

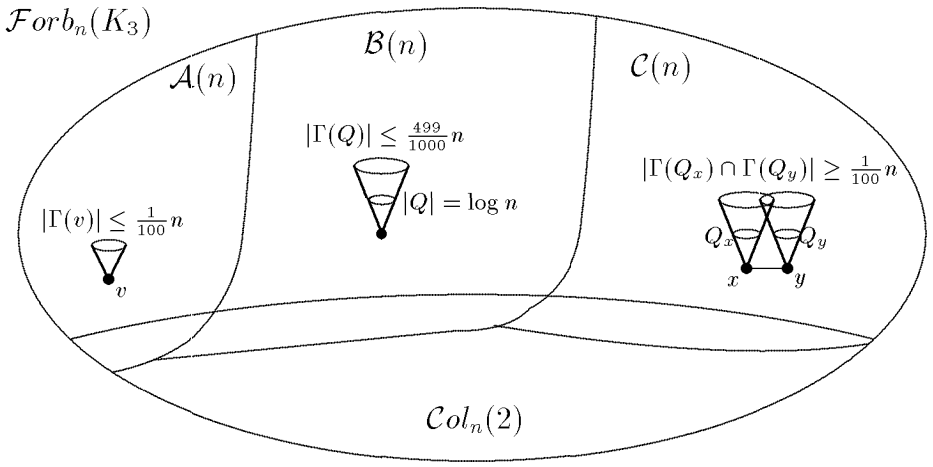


Abbildung 14.2: Partition of the set $\mathcal{F}orb_n(K_3)$

$\mathcal{B}(n)$ denote the set of all graphs in $\mathcal{F}orb_n(K_3) \setminus \mathcal{A}(n)$ which contain a vertex v and a set $Q \subseteq \Gamma(v)$ of size $\log n$ such that $|\Gamma(Q)| \leq (\frac{1}{2} - \frac{1}{1000})n$, and

$\mathcal{C}(n)$ denote the set of all graphs in $\mathcal{F}orb_n(K_3) \setminus [\mathcal{A}(n) \cup \mathcal{B}(n)]$ which contain an edge $\{x, y\}$ and sets $Q_x \subseteq \Gamma(x)$ and $Q_y \subseteq \Gamma(y)$ of size $|Q_x| = |Q_y| = \log n$ such that $|\Gamma(Q_x) \cap \Gamma(Q_y)| \geq \frac{1}{100}n$.

It is not difficult to show that these sets have the required properties.

Lemma 14.2.2 $\mathcal{F}orb_n(K_3) \subseteq \mathcal{C}ol_n(2) \cup \mathcal{A}(n) \cup \mathcal{B}(n) \cup \mathcal{C}(n)$.

Proof. Let $G = (V_n, E)$ be an arbitrary but fixed graph in $\mathcal{F}orb_n(K_3) \setminus [\mathcal{A}(n) \cup \mathcal{B}(n) \cup \mathcal{C}(n)]$. We need to show that $G \in \mathcal{C}ol_n(2)$. To see this fix for every vertex $v \in V_n$ a set $Q_v \subseteq \Gamma(v)$ of size $\log n$ and let $R_v = \Gamma(Q_v)$. Observe that by the definition of the set $\mathcal{A}(n)$ such sets Q_v exist and that by the definition of $\mathcal{B}(n)$ the sets R_v satisfy $|R_v| \geq (\frac{1}{2} - \frac{1}{1000})n$. Furthermore, by the definition of the set $\mathcal{C}(n)$, we have $|R_x \cap R_y| \leq \frac{1}{100}n$ for all $\{x, y\} \in E$. One easily checks that this implies that G can neither contain a C_5 nor a C_7 or C_9.

Choose now an arbitrary edge $\{x, y\} \in E$. As G contains no C_5 we conclude that R_x and R_y are stable sets and that $R_x \cap R_y = \emptyset$. Let $S = V_n \setminus (Q_x \cup Q_y \cup R_x \cup R_y)$ and

$$S_x = \{v \in S |\ R_v \cap R_x \neq \emptyset\} \qquad \text{and} \qquad S_y = \{v \in S |\ R_v \cap R_y \neq \emptyset\}.$$

Observe that certainly $S_x \cup S_y = S$ and that, as G contains no C_7 or C_9, we also know that $R_x \cup S_x$ and $R_y \cup S_y$ are stable and $S_x \cap S_y = \emptyset$. At this point a proper two coloring of G is easily defined (cf. Figure 14.3). □

Part II. We now bound the cardinalities of the sets $\mathcal{A}(n)$, $\mathcal{B}(n)$ and $\mathcal{C}(n)$ in terms of those of $\mathcal{F}orb_{n-x}(K_3)$ for an appropriate $x \in \mathbb{N}$. This is done by first closing an appropriate subset of the vertex set, in case of $\mathcal{A}(n)$, for example, the vertex v, a K_3-free graph on the remaining vertices and then connecting the vertices of the special set with the remaining vertices.

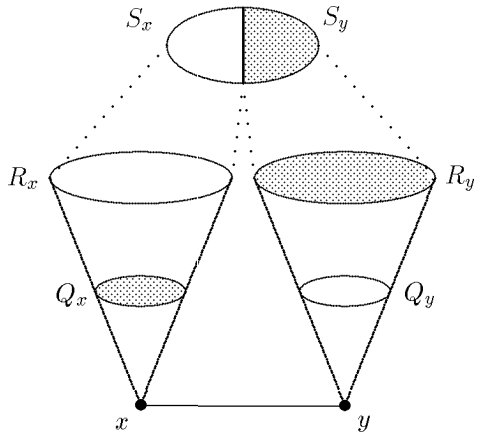


Abbildung 14.3: 2-coloring of a graph G in $G \in \mathcal{F}orb_n(K_3) \setminus [\mathcal{A}(n) \cup \mathcal{B}(n) \cup \mathcal{C}(n)]$

Lemma 14.2.3 *For all sufficiently large n*

$$\log \frac{|\mathcal{A}(n)|}{|\mathcal{F}orb_{n-1}(K_3)|} \leq 2(\log n)^2.$$

Proof. Construct all graphs in $\mathcal{A}(n)$ as follows. First choose the vertex v and a triangle-free graph on $V_n \setminus \{v\}$ (in at most $n \cdot |\mathcal{F}orb_{n-1}(K_3)|$ ways). Then choose the set $\Gamma(v)$. As there are exactly

$$\sum_{i=0}^{\log n} \binom{n-1}{i} \leq n^{\log n}$$

ways to do this, the desired bound follows immediately. □

Lemma 14.2.4 *For all sufficiently large n*

$$\log \frac{|\mathcal{B}(n)|}{|\mathcal{F}orb_{n-\log n}(K_3)|} \leq (\frac{1}{2} - \frac{1}{2000})n \log n.$$

Proof. Construct all graphs in $\mathcal{B}(n)$ as follows. First choose the set Q and a triangle-free graph on $V_n \setminus Q$ (in at most $\binom{n}{\log n} \cdot |\mathcal{F}orb_{n-\log n}(K_3)|$ ways). Then choose the set $R = \Gamma(Q)$ (less than 2^n ways) and connect Q to the set R (at most $2^{|Q| \cdot |R|} \leq 2^{\log n \cdot (\frac{1}{2} - \frac{1}{1000})n}$ ways). Together this gives

$$\begin{aligned} \log \frac{|\mathcal{B}(n)|}{|\mathcal{F}orb_{n-\log n}(K_3)|} &\leq (\log n)^2 + n + (\frac{1}{2} - \frac{1}{1000})n \log n \\ &\leq (\frac{1}{2} - \frac{1}{2000})n \log n, \end{aligned}$$

for n sufficiently large. □

Remark. Observe that in the proof of Lemma 14.2.4 we did not use the fact that the set Q is a subset of the neighbourhood of some vertex v. That is, we could also have let $\mathcal{B}(n)$ denote the (larger) set of all graphs in $\mathcal{F}orb_n(K_3)$ which contain a set $Q \subseteq \Gamma(v)$ of size $\log n$ such that $|\Gamma(Q)| \leq (\frac{1}{2} - \frac{1}{1000})n$.

Lemma 14.2.5 *For all sufficiently large n*

$$\log \frac{|\mathcal{C}(n)|}{|\mathcal{F}orb_{n-2}(K_3)|} \leq (1 - \frac{1}{2000})n.$$

Proof. Construct all graphs in $\mathcal{C}(n)$ as follows. First choose the two vertices x and y, a triangle-free graph on $V_n \setminus \{x, y\}$, and appropriate sets Q_x, Q_y (in less than $n^2 \cdot |\mathcal{F}orb_{n-2}(K_3)| \cdot n^{2\log n}$ ways). Let for conciseness of notation $R_x = \Gamma(Q_x)$ and $R_y = \Gamma(Q_y)$ and observe that R_x and R_y are determined by the choice of Q_x and Q_y. Finally, connect x and y to $V_n \setminus \{x, y\}$. As no vertex in R_x (R_y) may be connected to y (x) and no vertex in $V_n \setminus (R_x \cup R_y)$ may be connect to both x and y there are at most

$$2^{|R_x \setminus R_y| + |R_y \setminus R_x|} \cdot 3^{n - |R_x \cup R_y|} \leq 2^{\frac{7}{4}n - \frac{3}{4}(|R_x| + |R_y|) - \frac{1}{4}|R_x \cap R_y|} \leq 2^{n - \frac{1}{1000}n}.$$

ways to do this (recall that by definition of $\mathcal{B}(n)$ we have $|R_x|, |R_y| \geq (\frac{1}{2} - \frac{1}{1000})n$ and that by assumption $|R_x \cap R_y| \geq \frac{1}{100}n$). Together this gives

$$\begin{aligned} \log \frac{|\mathcal{C}(n)|}{|\mathcal{F}orb_{n-2}(K_3)|} &\leq 2\log n + 2(\log n)^2 + n - \frac{1}{1000}n \\ &\leq (1 - \frac{1}{2000})n, \end{aligned}$$

for n sufficiently large. □

Part III. To show inductively that the sets $\mathcal{A}(n)$, $\mathcal{B}(n)$ and $\mathcal{C}(n)$ are negligible compared to $\mathcal{C}ol_n(2)$ we first establish bounds on the growth rate of the set $\mathcal{C}ol_n(2)$.

Lemma 14.2.6 $\log \frac{|\mathcal{C}ol_{n-1}(2)|}{|\mathcal{C}ol_n(2)|} \leq -\frac{1}{2}n + \frac{1}{2}$ *for all* $n \in \mathbb{N}$.

Proof. Rewriting the statement of the lemma, we see that we have to prove that

$$|\mathcal{C}ol_n(2)| \geq 2^{\frac{1}{2}(n-1)} |\mathcal{C}ol_{n-1}(2)|.$$

This, however, follows easily from the observation that we can obtain every bipartite graph on n vertices by first choosing a bipartite graph on $n-1$ vertices ($|\mathcal{C}ol_{n-1}(2)|$ choices), fixing an arbitrary 2-coloring of it, and connecting the n-th vertex in all possible ways to the *larger* color class (there are at least $2^{\frac{1}{2}(n-1)}$ ways to do this). □

Proof of Theorem 14.2.1.
Since $\mathcal{C}ol_n(2) \subseteq \mathcal{F}orb_n(K_3)$ it suffices to show that there exist constants $c \geq 0$ and $\gamma > 1$ such that

$$|\mathcal{F}orb_n(K_3)| \leq (1 + c\gamma^{-n})|\mathcal{C}ol_n(2)| \qquad (*)$$

holds for all $n \in \mathbb{N}$. Let $\gamma = 2^{-\frac{1}{3000}}$ and choose n_0 large enough so that all lemmas above hold for all $n \geq n_0$ and such that for all $n \geq n_0$:

$$\max\{2^{-\frac{1}{2}n + 3(\log n)^2},\ 2^{-\frac{1}{2000}n\log n + 2(\log n)^2},\ 2^{-\frac{1}{2000}n + 3\log n}\} \leq \gamma^{-n}.$$

Subsequently, choose $c \geq 1$ such that $(*)$ holds for all $n \leq n_0$. We conclude the proof by induction on n. So assume $(*)$ holds for some $n - 1 \geq n_0$. By Lemma 14.2.2 we conclude that

$$|\mathcal{F}orb_n(K_3)| \leq |\mathcal{A}(n)| + |\mathcal{B}(n)| + |\mathcal{C}(n)| + |\mathcal{C}ol_n(2)|.$$

Hence, it suffices to show that the ratio of each of the set $\mathcal{A}(n)$, $\mathcal{B}(n)$ and $\mathcal{C}(n)$ with $\mathcal{C}ol_n(2)$ is at most $\frac{c}{3}\gamma^{-n}$. By Lemma 14.2.3, induction hypothesis, and Lemma 14.2.6 we conclude

$$\begin{aligned}\frac{|\mathcal{A}(n)|}{|\mathcal{C}ol_n(2)|} &\leq \frac{|\mathcal{A}(n)|}{|\mathcal{F}orb_{n-1}(K_3)|}\cdot\frac{|\mathcal{F}orb_{n-1}(K_3)|}{|\mathcal{C}ol_{n-1}(2)|}\cdot\frac{|\mathcal{C}ol_{n-1}(2)|}{|\mathcal{C}ol_n(2)|}\\ &\leq 2^{2(\log n)^2}\cdot\underbrace{(1+c\gamma^{-n+1})}_{\leq 2c}\cdot 2^{-\frac{1}{2}n+\log n} \leq \frac{c}{3}\cdot 2^{-\frac{1}{2}n+3(\log n)^2} \leq \frac{c}{3}\gamma^{-n},\end{aligned}$$

by choice of γ. Similarly, we obtain

$$\begin{aligned}\frac{|\mathcal{B}(n)|}{|\mathcal{C}ol_n(2)|} &\leq \frac{|\mathcal{B}(n)|}{|\mathcal{F}orb_{n-\log n}(K_3)|}\cdot\frac{|\mathcal{F}orb_{n-\log n}(K_3)|}{|\mathcal{C}ol_{n-\log n}(2)|}\cdot\prod_{i=1}^{\log n}\frac{|\mathcal{C}ol_{n-i}(2)|}{|\mathcal{C}ol_{n-i+1}(2)|}\\ &\leq 2^{(\frac{1}{2}-\frac{1}{2000})n\log n}\cdot(1+c\gamma^{-n+\log n})\cdot 2^{\sum_{i=1}^{\log n}(-\frac{1}{2}(n-i)+\log(n-i))}\\ &\leq \frac{c}{3}\cdot 2^{-\frac{1}{2000}n\log n+2(\log n)^2} \leq \frac{c}{3}\gamma^{-n}\end{aligned}$$

and

$$\begin{aligned}\frac{|\mathcal{C}(n)|}{|\mathcal{C}ol_n(2)|} &\leq \frac{|\mathcal{C}(n)|}{|\mathcal{F}orb_{n-2}(K_3)|}\cdot\frac{|\mathcal{F}orb_{n-2}(K_3)|}{|\mathcal{C}ol_{n-2}(2)|}\cdot\frac{|\mathcal{C}ol_{n-2}(2)|}{|\mathcal{C}ol_{n-1}(2)|}\cdot\frac{|\mathcal{C}ol_{n-1}(2)|}{|\mathcal{C}ol_n(2)|}\\ &\leq 2^{(1-\frac{1}{2000})n}\cdot(1+c\gamma^{-n+2})\cdot 2^{-n+2\log n+\frac{1}{2}}\\ &\leq \frac{c}{3}\cdot 2^{-\frac{1}{2000}n+3\log n} \leq \frac{c}{3}\gamma^{-n}.\end{aligned}$$

□

Wenn man größere Cliquen verbietet, ergibt sich eine analoge Aussage:

Satz 14.2.7 (KOLAITIS, PRÖMEL, ROTHSCHILD 1987) *Sei $p \geq 2$ konstant. Dann ist fast jeder K_{p+1}-freie Graph schon p-chromatisch:*

$$\text{Prob}\,[G_n \in \mathcal{C}ol_n(p) |\, G_n \in \mathcal{F}orb_n(K_{p+1})] = 1 - o(1) \qquad (n \to \infty).$$ □

Wenn man also die Cliquenzahl festhält, d.h. nur die Menge aller Graphen mit $\omega(G) \leq k$ für ein $k \in \mathbb{N}$ betrachtet, so ist die Cliquenzahl „in der Regel“ eine „gute“ untere Schranke für die chromatische Zahl, auch wenn die beiden Parameter i.a. beliebig weit auseinanderfallen können.

Darüberhinaus zeigt der Beweis zu Satz 14.2.1, daß fast alle dreiecksfreien Graphen Untergraphen des extremalen dreiecksfreien Graphen sind. Die Anzahl dreiecksfreier Graphen ist also im wesentlichen durch die Anzahl Subgraphen eines extremalen dreiecksfreien Graphen bestimmt. Dies verallgemeinert auf beliebige verbotene Subgraphen, wo die Anzahl Kanten im extremalen Graphen durch (14.5) gegeben ist.

Satz 14.2.8 (ERDŐS, FRANKL, RÖDL 1986) *Sei H ein Graph. Dann gilt*

$$|\mathcal{F}orb_n(H)| = 2^{(1-\frac{1}{\chi(H)-1})\frac{n^2}{2}+o(n^2)} \qquad (n \to \infty).$$ □

14.3 Mehr über dreiecksfreie Graphen

Nach Satz 14.2.1 ist die Menge der dreiecksfreien Graphen asymptotisch nur unwesentlich größer als die der bipartiten Graphen. Trotzdem unterscheiden sich die beiden Graphenklassen drastisch. So sahen wir bereits in Abschnitt 5.3, daß $\mathcal{F}orb(K_3)$ Graphen beliebig hoher chromatischer Zahl enthält. Darüberhinaus werden die Probleme 3-COLORABILITY und INDEPENDENT SET, für die es auf bipartiten Graphen polynomielle Algorithmen gibt, auf dreiecksfreien Graphen bereits NP-vollständig, vergleiche Satz 5.4.7 bzw. Übung 1.5.10. Dieser Abschnitt untersucht, inwieweit sich die „klassischen" Schranken an $\chi(G)$ und $\alpha(G)$ für dreiecksfreie Graphen verschärfen lassen.

Die chromatische Zahl. Der Satz von BROOKS gibt eine obere Schranke von $\Delta(G)$ für die chromatische Zahl eines Graphen G, der nicht gerade vollständig oder ein ungerader Kreis ist. Man kann sich nun fragen, ob man für Graphen mit Taillenweite mindestens g, $g \geq 4$, bessere Schranken angeben kann. Tatsächlich gilt für dreiecksfreie Graphen

$$\chi(G) \;\leq\; \frac{2}{3}\Delta(G) + 2,$$

wie KOSTOCHKA (siehe [Kim95]) zeigen konnte. Andererseits gibt es Graphen beliebig hoher Taillenweite mit

$$\chi(G) \;\geq\; c\frac{\Delta(G)}{\log \Delta(G)},$$

siehe [Bol78b]. Für Graphen mit Taillenweite $g(G) \geq 5$ ist dies tatsächlich die korrekte Größenordnung; KIM [Kim95] konnte zeigen, daß für Graphen mit $g(G) \geq 5$ gilt:

$$\chi(G) \;\leq\; (1 + o(1))\,\frac{\Delta(G)}{\ln \Delta(G)} \qquad (\text{ für } \Delta(G) \to \infty).$$

Stabile Mengen. Ein klassisches Resultat von TURÁN [Tur41] (siehe Proposition 1.5.6) kann wie folgt formuliert werden, wobei $\bar{d} := \frac{1}{n}\sum_{v \in V} d(v)$ den Durchschnittsgrad in G bezeichnet:

$$\alpha(G) \;\geq\; \frac{n}{\bar{d}+1}.$$

Während diese Abschätzung für eine disjunkte Vereinigung von Cliquen scharf ist, gilt für dreiecksfreie Graphen

$$\alpha(G) \;\geq\; (1 - o(1))\,\frac{n \ln \bar{d}}{\bar{d}} \qquad (\text{ für } \bar{d} \to \infty).$$

Genauer (siehe auch [AKS80, Gri83b]):

Satz 14.3.1 (SHEARER 1983) *Sei $G = (V, E)$ ein dreiecksfreier Graph mit Durchschnittsgrad d. Definiere eine Funktion $f : \mathbb{R}_0^+ \to \mathbb{R}$ durch*

$$f(x) \;:=\; \frac{d \ln d - d + 1}{(d-1)^2},$$

$f(0) = 1$ und $f(1) := 1/2$. Dann gilt $\alpha(G) \geq nf(d)$.

Beweis. Wir induzieren nach n. Für $n = 1$ ist die Aussage sicher richtig. Ebenso für relativ dichte Graphen, denn für $n \leq d/f(d)$ stellen schon die Nachbarn eines Knotens von maximalem Grad in G eine stabile Menge der Größe $\geq d \geq nf(d)$ dar.

Die Funktion f ist stetig und erfüllt $0 < f(d) \leq 1$. Sei $v \in V$ ein Knoten von G, $d_1 = |\Gamma(v)|$ sein Grad und $d_2 = \frac{1}{d_1}\sum_{u\in\Gamma(v)} d(u)$ der Durchschnittsgrad seiner Nachbarn. Wir behaupten, daß v so gewählt werden kann, daß

$$(d_1 + 1)f(d) \;\leq\; 1 + (dd_1 + d - 2d_1d_2)f'(d). \tag{14.6}$$

Hierzu wiederum genügt es, zu zeigen, daß (14.6) im Mittel über alle Knoten von G gilt. f genügt der Differentialgleichung

$$(d+1)\,f(d) \;=\; 1 + (d - d^2)\,f'(d). \tag{14.7}$$

Der Mittelwert über alle Knoten $v \in V$ der linken Seite von (14.6) ist aber gleich der linken Seite von (14.7). Der Mittelwert von d_1d_2 ist gleich dem Mittelwert über d_1^2, was nach der JENSENschen Ungleichung $\geq d^2$ ist. Also ist der Mittelwert der rechten Seite von (14.6) wegen $f'(d) < 0$ mindestens so groß wie die rechte Seite von (14.7), womit (14.6) gezeigt ist.

Sei nun v ein Knoten von G, der (14.6) erfüllt, und G' der Graph $G - (v + \Gamma(v))$. Dann ist G' ebenso dreiecksfrei, enthält $n - d_1 - 1$ Knoten und $\frac{1}{2}nd - d_1d_2$ Kanten. Also hat G' Durchschnittsgrad

$$d' \;=\; \frac{nd - 2d_1d_2}{n - d_1 - 1}.$$

Nach Induktionsannahme enthält G' eine stabile Knotenmenge der Größe $(n-d_1-1)f(d')$. Wir fügen v hinzu und erhalten eine stabile Knotenmenge in G der Größe (wegen $f''(d) \geq 0$ ist f konvex und verläuft oberhalb jeder Tangente)

$$\begin{array}{rcl}
(n - d_1 - 1)f(d') + 1 & \geq & (n - d_1 - 1)f(d) + (n - d_1 - 1)(d' - d)f'(d) + 1 \\
& = & (n - d_1 - 1)f(d) + (dd_1 + d - 2d_1d_2)f'(d) + 1 \\
& \geq & (n - d_1 - 1)f(d) + (d_1 + 1)f(d) \\
& = & nf(d).
\end{array}$$

□

Aus dem Beweis ergibt sich unmittelbar ein polynomieller Algorithmus, um eine stabile Knotenmenge der geforderten Größe zu konstruieren. Bis auf einen konstanten Faktor ist die Schranke aus obigem Satz bestmöglich, da es Graphen von beliebig hoher Taillenweite gibt mit

$$\alpha(G) \;\leq\; (2 + o(\bar{d}))\,\frac{n \ln \bar{d}}{\bar{d}}$$

(dies ergibt sich durch Betrachtung zufälliger Graphen).

MOON und MOSER [MM65] bestimmten die maximale Anzahl $MM(n)$ maximaler Cliquen (rsp. maximaler stabiler Mengen) in einem Graphen auf n Knoten: es gilt $MM(n) = \Theta(3^{n/3}) \approx \Theta(1.44^n)$. Sie bestimmten auch die (eindeutigen) extremalen Graphen; Es sind dies beispielsweise für $n \equiv 0 \pmod 3$ die TURÁN-Graphen $T_n(n/3)$.

Satz 14.3.2 (MOON, MOSER 1965) *Sei $G = (V, E)$ ein Graph auf n Knoten und $MM(G)$ die Anzahl der (inklusions-) maximalen Cliquen in G. Dann gilt:*

$$MM(G) \leq \begin{cases} 3^{\frac{n}{3}} & \text{falls } n \equiv 0 \pmod 3, \\ 4 \cdot 3^{\frac{n-4}{3}} & \text{falls } n \equiv 1 \pmod 3, \\ 2 \cdot 3^{\frac{n-2}{3}} & \text{falls } n \equiv 2 \pmod 3. \end{cases}$$

Beweis. Für einen Knoten $x \in V$ bezeichnen wir mit $MM(x)$ die Anzahl maximaler Cliquen in G, die x enthalten, und mit $MM_\Gamma(x)$ die Anzahl der Cliquen in $\Gamma(x)$, die maximale Cliquen in $G - x$ sind.

Seien nun x und y zwei Knoten, die in G nicht durch eine Kante verbunden sind. Dann definieren wir einen Graphen $G_{x \to y}$ wie folgt: zunächst löschen wir alle Kanten, die von x ausgehen und dann fügen wir neue Kanten von x zu allen Nachbarn von y hinzu.

Wie verhalten sich nun $MM(G)$ und $MM(G_{x \to y})$ zueinander? Offenbar gehen alle maximalen Cliquen in G, die x enthalten ($MM(x)$ viele), verloren. Neu hinzu kommen alle Cliquen in $\Gamma(x)$, die in $G - x$ maximal sind ($MM_\Gamma(x)$ viele), und „Kopien" für jede maximale Clique, die y enthält ($MM(y)$ viele). D.h., es gilt

$$MM(G_{x \to y}) \;=\; MM(G) - MM(x) + MM_\Gamma(x) + MM(y). \tag{14.8}$$

Zwei Knoten $\{x, y\} \in \binom{V}{2} \setminus E$ mögen *äquivalent* heißen, falls $\Gamma(x) = \Gamma(y)$ (beachte: dies ist eine Äquivalenzrelation auf V). Sei nun $G = (V, E)$ ein Graph mit maximalem $MM(G)$ unter allen Graphen auf n Knoten und unter allen solchen einer mit maximal vielen äquivalenten Knotenpaaren. Wir behaupten, daß in G dann schon *alle* nicht benachbarten Knoten äquivalent sind. Denn angenommen, G enthielte ein Paar $\{x, y\} \in \binom{V}{2} \setminus E$ mit $\Gamma(x) \neq \Gamma(y)$. Sei o.B.d.A. $MM(x) \leq MM(y)$, und sei $X \subset V$ die Menge aller zu x äquivalenten Knoten in $V \setminus \Gamma(x)$ vereinigt mit $\{x\}$. Dann gilt für alle $x' \in X$: $\{x', y\} \notin E$ und $MM(G_{x' \to y}) \geq MM(G)$ nach (14.8). Mithin wird $MM(G_{X \to y}) \geq MM(G)$ für den Graphen $G_{X \to y}$, der entsteht, wenn man die Operation $x' \to y$ sukzessive für alle $x' \in X$ ausführt. Der Graph $G_{X \to y}$ enthält jedoch mehr äquivalente Knoten als G: um dies einzusehen, genügt es offenbar, zu zeigen, daß keine äquivalenten Knotenpaare „verloren" gegangen sind, d.h., daß es kein „schlechtes" Paar $\{x', y'\} \in \binom{V}{2} \setminus E$ gibt, das in G, nicht jedoch in $G_{X \to y}$ äquivalent ist. Weil x' und y' äquivalent sind, stehen sie in demselben Nachbarschaftsverhältnis zu x bzw. y, d.h. sie liegen beide in einer der folgenden 4 Mengen: $\Gamma(x) \cap \Gamma(y)$, $\Gamma(x) \setminus \Gamma(y)$, $\Gamma(y) \setminus \Gamma(x)$, $V \setminus (\Gamma(x) \cup \Gamma(y))$. Es ist leicht einzusehen, daß für ein schlechtes Paar keiner dieser Fälle in Frage kommt.

Man überlegt sich nun weiter, daß ein Graph G mit $\{x, y\} \notin E \;\Rightarrow\; \Gamma(x) = \Gamma(y)$ ein vollständig r-partiter Graph ist. Für einen r-partiten Graphen mit Partitionsklassen $V = V_1 \cup \ldots \cup V_r$ gilt andererseits offenbar

$$MM(G) \;=\; |V_1| \cdot \ldots \cdot |V_r|.$$

Es ist also nur noch zu zeigen, daß

$$\max\{n_1 \cdot \ldots \cdot n_r \mid r \in \mathbb{N}, n_i \in \mathbb{N}, \textstyle\sum_{i=1}^r n_i = n\} = \begin{cases} 3^{\frac{n}{3}} & \text{falls } n \equiv 0 \pmod 3, \\ 4 \cdot 3^{\frac{n-4}{3}} & \text{falls } n \equiv 1 \pmod 3, \\ 2 \cdot 3^{\frac{n-2}{3}} & \text{falls } n \equiv 2 \pmod 3. \end{cases}$$

Dies rechnet man leicht nach. □

Bemerkenswerterweise enthalten schon dreiecksfreie Graphen nur noch höchstens $2^{n/2}$ viele maximale stabile Knotenmengen [HT93].

Anhang A

A.1 Grundbegriffe der Wahrscheinlichkeitstheorie

Wir betrachten hier nur sogenannte LAPLACEsche Wahrscheinlichkeitsräume. Für allgemeine Wahrscheinlichkeitsräume haben die meisten der folgenden Definitionen und Sätze eine weitaus kompliziertere Form.

Ein LAPLACE*scher Wahrscheinlichkeitsraum* besteht aus einer endlichen Menge Ω, der ein sogenanntes LAPLACEsches Wahrscheinlichkeitsmaß $Prob : 2^\Omega \to [0,1]$ zugeordnet ist, das durch

$$Prob\,(A) \;=\; \frac{|A|}{|\Omega|}$$

für alle $A \subseteq \Omega$ definiert ist. Jede Menge $A \subseteq \Omega$ heißt *Ereignis*, ein Ereignis A mit $|A| = 1$ heißt *Elementarereignis*. Ein LAPLACEscher Wahrscheinlichkeitsraum ist dadurch charakterisiert, daß in ihm die Elementarereignisse alle gleich wahrscheinlich sind. Im folgenden bezeichnen wir einen LAPLACEschen Wahrscheinlichkeitsraum einfach nur als Wahrscheinlichkeitsraum oder sogar nur als Raum. Für k Ereignisse $A_1, \ldots, A_k$ gilt die folgende einfache, aber häufig benutzte Abschätzung:

$$Prob\,(\bigcup_{i=1}^{k} A_i) \;\leq\; \sum_{i=1}^{k} Prob\,(A_i).$$

Eine (reelle) *Zufallsvariable* auf einem Raum Ω ist eine Funktion $X : \Omega \to \mathbb{R}$. Für eine Menge $C \subseteq \mathbb{R}$ bezeichnet $X^{-1}(C)$ das Ereignis $\{\alpha \in \Omega : X(\alpha) \in C\}$. Für reelle Zahlen a und b ist dann zum Beispiel $|X - a| \geq b$ das Ereignis $\{\alpha \in \Omega : |X(\alpha) - a| \geq b\}$. Der *Erwartungswert* einer Zufallsvariablen X, in Zeichen $E(X)$, ist der Mittelwert von X über alle $\alpha \in \Omega$; das heißt

$$E(X) := \;\frac{1}{|\Omega|}\sum_{\alpha\in\Omega} X(\alpha) \;=\; \sum_{\alpha\in\Omega} Prob\,(\{\alpha\})\cdot X(\alpha) \;=\; \sum_{x\in X(\Omega)} Prob\,(X = x)\cdot x.$$

Der Erwartungswert hat einige offensichtliche Eigenschaften wie Monotonie:

$$(\forall \alpha \in \Omega :\; X(\alpha) \geq Y(\alpha)) \;\Rightarrow\; E(X) \geq E(Y)$$

und Linearität, d.h. für $a, b \in \mathbb{R}$ und zwei Zufallsvariablen X und Y gilt

$$E(aX + bY) \;=\; aE(X) + bE(Y).$$

Eine Funktion $f : C \to \mathbb{R}$ auf einem Intervall $C \subseteq \mathbb{R}$ heißt *konvex*, falls für alle $a, b \in C$ gilt, daß f im Intervall $[a, b]$ unterhalb der Sekante durch die Punkte $(a, f(a))$ und $(b, f(b))$ verläuft, d.h. für alle $x \in [a, b]$ gilt:

$$f(x) \quad \leq \quad f(a) + \frac{x-a}{b-a} \cdot (f(b) - f(a)).$$

Für differenzierbares f ist dies gleichbedeutend damit, daß die Funktion f für alle $a \in C$ oberhalb ihrer Tangente durch $(a, f(a))$ verläuft. Ist f sogar zweimal stetig differenzierbar, so ist f konvex genau dann, wenn $f''(x) > 0$ für alle $x \in C$.

Die folgende Proposition zeigt, wie sich der Erwartungswert von X unter Transformation durch eine konvexe Funktion verhält.

Proposition A.1.1 (JENSENsche Ungleichung) *Sei $C \subseteq \mathbb{R}$ ein Intervall und $f : C \to \mathbb{R}$ eine konvexe Funktion auf C. Sei ferner X eine Zufallsvariable mit Werten in C.*
a) Dann gilt $E(f(X)) \quad \geq \quad f(E(X))$.
b) Seien $x_1, \ldots, x_k \in C$. Dann gilt

$$\frac{1}{k} \sum_{i=1}^{k} f(x_i) \quad \geq \quad f\left(\frac{1}{k} \sum_{i=1}^{k} x_i\right).$$

Beweis. a) Weil f konvex ist, gibt es eine lineare Funktion h auf C mit $h(y) = a + by \leq f(y)$ für alle $y \in C$ und $h(E(X)) = f(E(X))$. Wenn f in $E(X)$ differenzierbar ist, ist h gerade die Tangente an f an der Stelle $E(X)$. Dann gilt

$$E(f(X)) \geq E(h(X)) = a + bE(X) = h(E(X)) = f(E(X)).$$

Zum Beweis von b) definiere eine Zufallsvariable X auf $\{1, \ldots k\}$ durch $X(i) := x_i$. Dann gilt $E(X) = \frac{1}{k} \sum_{i=1}^{k} x_i$ und mit a) folgt

$$\frac{1}{k} \sum_{i=1}^{k} f(x_i) = E(f(X)) \geq f(E(X)) = f\left(\frac{\sum_{i=1}^{k} x_i}{k}\right). \qquad \square$$

Für zwei Ereignisse A und B ist die *bedingte Wahrscheinlichkeit* $Prob\,(A|B)$ definiert als

$$Prob\,(A|B) := \frac{Prob\,(A \cap B)}{Prob\,(B)}.$$

Zwei Ereignisse A und B heißen *unabhängig*, falls $Prob\,(A \cap B) = Prob\,(A) \cdot Prob\,(B)$. Äquivalente Bedingungen sind $Prob\,(A|B) = Prob\,(A)$ oder $Prob\,(B|A) = Prob\,(B)$. Sei X eine Zufallsvariable auf Ω und $A \subseteq \Omega$ ein Ereignis. Dann ist der bedingte Erwartungswert von X bezüglich A, in Zeichen $E(X|A)$, definiert als der Mittelwert von X über A, das heißt

$$E(X|A) := \frac{1}{|A|} \sum_{\alpha \in A} X(\alpha) = \sum_{\alpha \in \Omega} X(\alpha) Prob\,(\{\alpha\}|A).$$

Für zwei Zufallsvariablen X und Y ist der bedingete Erwartungswert von X bezüglich Y eine Zufallsvariable, die den Erwartungswert von X angibt unter der Annahme, daß Y bekannt ist; das heißt

$$E(X|Y)(\alpha) := E(X|Y^{-1}(Y(\alpha))) = \frac{1}{|\{\beta : Y(\beta) = Y(\alpha)\}|} \sum_{\beta : Y(\beta) = Y(\alpha)} X(\beta). \qquad (A.1)$$

Mit Hilfe des bedingten Erwartungswertes kann der Erwartungswert eines Produktes von Zufallsvariablen folgendermaßen dargestellt werden.

Proposition A.1.2 *Seien X und Y zwei Zufallsvariablen auf dem Raum Ω. Dann gilt*
a) $E(X \cdot Y) = E(X \cdot E(Y|X))$
b) $E(X|X) = X$.

Beweis. a) Es gilt

$$\begin{aligned} E(X \cdot E(Y|X)) &= \frac{1}{|\Omega|} \sum_{\alpha \in \Omega} \left(X(\alpha) \frac{1}{|\{\beta : X(\beta) = X(\alpha)\}|} \sum_{\beta : X(\beta) = X(\alpha)} Y(\beta) \right) \\ &= \frac{1}{|\Omega|} \sum_{\beta \in \Omega} X(\beta) \cdot Y(\beta) \quad = \quad E(X \cdot Y). \end{aligned}$$

b) folgt direkt aus der Definition von $E(X|X)$. □

Die *Varianz* einer Zufallsvariablen X ist definiert als

$$Var(X) := E((X - E(X))^2) = E(X^2 - 2XE(X) + E(X)^2) = E(X^2) - E(X)^2.$$

Die beiden folgenden Propositionen zeigen, daß eine Zufallsvariable unter bestimmten Voraussetzungen nicht zu weit von ihrem Erwartungswert abweichen kann.

Proposition A.1.3 (MARKOFFsche Ungleichung) *Sei X eine Zufallsvariable auf Ω und $\lambda > 0$. Dann gilt*

$$Prob\,(X \geq \lambda) \quad \leq \quad \frac{E(X)}{\lambda}.$$

Beweis. Es gilt

$$E(X) \;=\; \frac{1}{|\Omega|} \sum_{\alpha \in \Omega} X(\alpha) \;\geq\; \frac{1}{|\Omega|} \sum_{\alpha \in \Omega : X(\alpha) \geq \lambda} \lambda \;=\; Prob\,(X \geq \lambda) \cdot \lambda.$$ □

Proposition A.1.4 (TSCHEBYSCHEFFsche Ungleichung) *Sei X eine Zufallsvariable auf Ω mit $E(X) \neq 0$ und $\lambda > 0$. Dann gilt*

a) $Prob\,(|X - E(X)| \geq \lambda E(X)) \quad \leq \quad \frac{Var(X)}{\lambda^2 E(X)^2}$
b) $Prob\,(X = 0) \quad \leq \quad \frac{Var(X)}{E(X)^2}$.

Beweis. a) Sei $Y := (X - E(X))^2$. Dann ist $E(Y) = Var(X)$, und mit der MARKOFFschen Ungleichung folgt

$$Prob\,((|X - E(X)| \geq \lambda E(X)) \quad = \quad Prob\,(Y \geq \lambda^2 E(X)^2) \leq \frac{E(Y)}{\lambda^2 E(X)^2} = \frac{Var(X)}{\lambda^2 E(X)^2}.$$

b) folgt aus a) mit $\lambda = 1$. □

Schließlich notieren wir noch ohne Beweis einige Beziehungen für Fakultäten und Binomialkoeffizienten.

$$n! \;=\; (1 + o(1)) \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \qquad \text{für } n \to \infty; \tag{A.2}$$

$$\binom{n}{k} \;\leq\; \left(\frac{en}{k}\right)^k; \tag{A.3}$$

$$\binom{n}{k} \;=\; (1 + o(1)) \, (2\pi k)^{-1/2} \left(\frac{en}{k}\right)^k \quad (\text{für } n >> k^2 >> 1). \tag{A.4}$$

Gleichung (A.2) heißt die STIRLINGsche Formel, für einen Beweis siehe z.B. [GKP89, Slo91]. Für $k \leq n$ erhält man alle k-elementigen Untermengen einer n-elementigen Menge, indem man eine k-elementige Menge festhält und für alle $j = 0, \ldots, k$ zuerst j Elemente aus dieser Menge und dann $k - j$ Elemente von den übrigen $n - k$ Elementen auswählt. Also gilt

$$\sum_{j=0}^{k} \binom{n-k}{k-j}\binom{k}{j} = \binom{n}{k}. \tag{A.5}$$

A.2 Hinweise und Lösungen zu ausgewählten Übungen

1.1.1 Nach Gleichung (1.1) gilt $\sum_{d(v_i)\text{gerade}} d(v_i) + \sum_{d(v_i)\text{ungerade}} d(v_i) = 2m$, und es wird $\sum_{d(v_i)\text{ungerade}} d(v_i) \equiv 0 \bmod 2$.

1.1.2 Falls ein innerer Knoten v_i, $1 \leq i < \ell$, doppelt auftritt, entferne das Zwischenstück (diese „Schleife“) aus dem $u-v$–Weg. Iteriere, bis alle inneren Knoten verschieden sind.

1.1.7 a) Als Knotengrade kommen nur die Zahlen $0, \ldots, n-1$ in Frage. Die Grade 0 und $n-1$ schließen sich aber gegenseitig aus.

1.1.9 $\sum_{v\in V} t(v) = \sum_{e=\{u,v\}\in E} \frac{d(u)}{d(v)} + \frac{d(v)}{d(u)} \geq 2m$. Da die Knoten von G also im Mittel die Abschätzung $t(v) \geq 2m/n$ erfüllen, muß es insbesondere mindestens einen Knoten geben, der sie erfüllt.

1.1.13 Sei G unzusammenhängend, $A \subset V$ eine seiner Komponenten, $B := V \setminus A$ sowie $a \in A$ und $b \in B$. Dann ist a in $\bar{G}$ vollständig zu allen Knoten aus B verbunden und b vollständig zu allen Knoten aus A. Wegen $\{a, b\} \in E(\bar{G})$ ist also $\bar{G}$ zusammenhängend.

1.1.14 Angenommen, G besitzt eine Komponente $\emptyset \neq V_1 \neq V$. Mit $V_2 := V \setminus V_1$ ist also der Schnitt $\langle V_1, V_2\rangle$ leer. G hat daher höchstens

$$\binom{n}{2} - |V_1||V_2| \leq n(n-1)/2 - (n-1) = \binom{n-1}{2}$$

viele Kanten, Widerspruch.

1.1.38 a) Trivial für $|T| = 1, 2$. Sei also $3 \leq |T| \leq \delta(G)+1$ und b ein Blatt von T. Dann ist $T - b$ nach Induktionsannahme gleich einem Subgraph H von G. Sei a' der Knoten in H, der dem Nachbarn a von b in T entspricht. Wegen $|H - a'| \leq \delta(G) - 1$ gehört mindestens ein Nachbar von a' in G nicht zu H. Identifiziere diesen mit b. **b)** Klar.

1.1.40 Weise jeder Kante $\{A_i, A_j\} \in E$ die Marke $A_i \Delta A_j \in S$ zu. Ein maximaler Wald W in G hat höchstens $n-1$ Kanten, die folglich eine Marke $x \in S$ nicht enthalten. Angenommen, $A_i \cup \{x\} = A_j \cup \{x\}$ für ein Paar $i \neq j$. Wegen $A_i \neq A_j$ gilt o.B.d.A. $x \in A_j$, d.h. es ist $A_i \Delta A_j = \{x\}$. Mithin existiert ein Pfad von A_i nach A_j in W, der eine Kante mit Marke x enthält, Widerspruch.

1.1.41 Betrachte zwei Knoten maximalen Abstandes und benutze Proposition 1.1.31 (iii).

1.1.42 „$\Rightarrow$“: Sei F minimal trennend und $e \in F$. Dann ist also der Graph $H := G-(F-e)$ noch zusammenhängend, d.h. e eine Brücke in H. Eine Brücke e in einem Graphen H verbindet aber genau zwei Komponenten A und B von $H-e$. Da nun F die Komponenten

A und B trennt, gilt offensichtlich $\langle A, B\rangle \subseteq F$. Die umgekehrte Inklusion folgt aus der Minimalität von F: angenommen, $f \in F$ aber $f \notin \langle A, B\rangle$. Dann ist f o.B.d.A. eine Kante aus $G[A]$. Es folgt, daß $F - f$ schon trennend wäre.
„$\Leftarrow$“: Da $G[A]$ und $G[B]$ zusammenhängend sind, ist $G - (F - e)$ noch zusammenhängend für jedes $e \in F$.

1.3.8 Von jedem Knoten aus eine Breitensuche, die nach der ersten Kante zwischen zwei bekannten Knoten sucht.

1.3.9 Zähle zunächst die Kanten von G. Falls $m < \binom{n}{2} - n^2/4$, so ist G nicht das Komplement eines bipartiten Graphen, andernfalls gilt $m = \Omega(n^2)$ und man „darf“ das Komplement $\overline{G}$ berechnen.

1.4.22 c) Sei G eine Instanz von NODE COVER. O.B.d.A. habe G keine isolierten Knoten. Verbinde für alle $e = \{x, y\} \in E$ die Knoten x und y zusätzlich durch einen Pfad (x, v_e, y) der Länge 2 und erhalte einen Graphen H. Eine Knotenüberdeckung $C \subset V$ in G stellt dann auch eine dominierende Menge in H dar, da jeder Knoten $v \in V(G)$ mit einer Kante inzidiert, die von C überdeckt wird, und jeder Knoten v_e, $e \in E(G)$, „parallel“ zu einer Kante ist, die von C überdeckt wird. Also gilt $\sigma(H) \leq \tau(G)$.
Sei umgekehrt D eine minimale dominierende Knotenmenge in H. Dann enthält D aus jedem Dreieck x, v_e, y wie oben mindestens einen Knoten, um v_e zu dominieren. Enthält D zwei Knoten aus x, v_e, y, dann genau die Knoten x und y. Aus D läßt sich wie folgt eine Knotenüberdeckung C von G mit $|C| = |D|$ gewinnen. Wir übernehmen alle Knoten aus $D \cap V(G)$ nach C, und zusätzlich nehmen wir für alle $v_e \in D$ beliebig einen der Knoten x oder y zu C hinzu, falls v_e im Dreieck x, v_e, y liegt. Es folgt $\tau(G) \leq \sigma(H)$ und damit insgesamt $\tau(G) = \sigma(H)$.

1.5.10 b) Es gilt $\alpha(H) = \alpha(G) + m(G) \cdot t$. Setze $t = n(G)^{\frac{2r}{1-r}}$.

2.1.5 Wir geben für die ersten sechs augmentierenden Pfade an, welche der Kanten e_i, $i = 0, 1, 2, 3$ sie in welcher Reihenfolge und Richtung durchlaufen; der übrige Verlauf der Pfade ergibt sich daraus. Die Notation $-e_i$ bedeutet hierbei, daß die Kante $e_i = (x_i, y_i)$ in der Richtung $y_i \to x_i$ durchlaufen wird.
0. augmentierender Pfad: e_0
1. augmentierender Pfad: $e_1 - e_0$
2. augmentierender Pfad: $e_2 - e_1 - e_0$
3. augmentierender Pfad: $e_3 - e_2 - e_1$
4. augmentierender Pfad: $e_0 - e_3 - e_2$
5. augmentierender Pfad: $e_1 - e_0 - e_3$ u.s.w.

Nach der i-ten Augmentierung ($i > 0$) sind die Flußwerte auf den Kanten e_i, $i = 0, 1, 2, 3$, also stets $c_{i+1} = c_{i-1} - c_i$, c_i, 0 und 0 in zyklischer Reihenfolge.

2.2.7 Wegen $d(v) \geq (n-1)/2$ hat jede Komponente von G mindestens $(n+1)/2$ vielen Knoten. Also gibt es nur eine. Falls G nun eine Artikulation v besitzt, so hat jede Komponente von $G - v$ mindestens $(n-1)/2$ Knoten, d.h. G ist der Ausnahmegraph.

2.2.9 a) Analog zum Beweis des Fächsatzes 2.2.5. Für „$\Leftarrow$“zeige zunächst $\delta(G) \geq k$. **b)** Angenommen, es gäbe eine $(2k-2)$−elementige trennende Knotenmenge S. Partitioniere S in zwei $(k-1)$−elementige Mengen und füge jeder einen Knoten aus je einer Komponente von $G - S$ hinzu.

2.2.10 Analog zum Beweis von Satz 2.2.6.

2.3.9 Die Notwendigkeit des zweifachen Kantenzusammenhangs ist offensichtlich (sonst gäbe es eine Brücke). Beachte: die beiden gerichteten $u-v-$ und $v-u-$Pfade sind nicht notwendig kantendisjunkt. Sei also G ein zweifach kantenzusammenhängender Graph. Um zu zeigen, daß G stark orientierbar ist, beweise ein Analogon des Satzes 2.3.1 oder nimm indirekt die Existenz einer Menge $S \subset V$, $S \neq V$, an, die knotenmaximal mit der Eigenschaft ist, daß $G[S]$ eine stark zusammenhängende Orientierung besitzt. Dann analog zum Beweis des Satzes 2.3.1. [Man überlege sich, an welchen Stellen ein konstruktiver Beweis hier mehr Aufwand erfordert.]

2.3.10 (i) $\Rightarrow$ (ii): O.B.d.A. sei $\{u,v\} \in E$ (sonst füge man sie noch ein). Nach Satz 2.3.3 gibt es einen Kreis, der $\{u,v\}$ und die weitere Kante enthält. (ii) $\Rightarrow$ (iii): trivial. (iii) $\Rightarrow$ (i): Satz 2.3.3 (iii). (i) $\Rightarrow$ (iv): Wenn jeder $u-v-$Weg den Knoten w enthielte, wäre w nach 1.1.31(ii) eine Artikulation und G nicht 2-zusammenhängend. (iv) $\Rightarrow$ (i): G ist zusammenhängend, enthält aber keine Artikulationen.

2.4.3 Graphen G_n: zwei disjunkte Kopien des K_n, die beide vollständig mit einem $(2n+1)$-ten Knoten verbunden sind; Graphen G'_n: zwei disjunkte Kopien des K_{n+1}, die durch eine Kante verbunden sind.

2.4.11 a) Nach Übung 1.1.42 hat $G-F$ genau zwei Komponenten H_1 und H_2. Sei $|H_1| \leq |H_2|$ und bezeichne $d_i(v)$ die Anzahl Nachbarn von v in H_i.
Fall 1: $\exists u \in H_1 : \Gamma(u) \subset H_1$.
Dann gilt also $|H_1| \geq \delta(G)+1$. Wegen $d(u,v)=2$ für alle $v \in H_2$, ist jeder Knoten in H_2 mit einem Knoten in $\Gamma(u) \subset V(H_1)$ adjazent. Es folgt

$$\lambda = |F| = \sum_{v \in H_2} d_1(v) \geq |H_2| \geq |H_1| \geq \delta(G)+1.$$

Fall 2: $d_2(u) > 0 \ \forall u \in H_1$.
Sei $u \in H_1$. Dann gilt

$$\delta(G) \leq d(u) = d_1(u) + d_2(u) \leq |H_1 - u| + d_2(u) \leq \sum_{v \in H_1 - u} d_2(v) + d_2(u) \leq |F| = \lambda.$$

2.4.13 Siehe [TS92, Seite 202] oder [BM76].

3.1.5 Gruppiere die $2k$ Knoten ungeraden Grades in k Knotenpaare und verbinde die Knoten eines jeden Paares durch eine neue Kante bzw. einen Pfad der Länge 2. Der entstehende Graph ist Eulersch.

3.1.7 „$\Rightarrow$“: Wenn man in u losläuft und als nächste Wegkante auf einem $u-v-$Weg in $G-e$ jeweils beliebig eine der noch nicht benutzten inzidierenden Kanten wählt, hat man stets ungerade viele Möglichkeiten, den Weg fortzusetzen. Insbesondere erreicht man also schließlich den Knoten v. Die Menge aller $u-v-$Wege (die u und v nur als Endknoten enthalten) läßt sich also durch einen Baum mit Wurzel u darstellen, der in jedem Knoten x entsprechend der verschiedenen Möglichkeiten, den $u-x-$Weg fortzusetzen, verzweigt. Da jeder Knoten in diesem Baum ungerade viele Söhne hat und eine Summe aus ungerade vielen ungeraden Summanden ungerade ist, hat der Baum ungerade viele Blätter, die der Menge der $u-v-$Wege entsprechen. Die $u-v-$Wege, die keine $u-v-$Pfade sind, treten aber in Paaren auf entsprechend der Umlaufrichtung des ersten Kreises, den sie enthalten.

Also gibt es in $G-e$ ungerade viele $u-v$–Pfade, und e liegt in ungerade vielen Kreisen. „$\Leftarrow$“: Betrachte die mit einem Knoten u inzidierenden Kanten e. Jede von ihnen liegt in ungerade vielen Kreisen, sei diese Anzahl $k(e)$. Da die Summe über alle $k(e)$, so daß e mit u inzidiert, gerade zweimal die Anzahl aller Kreise ist, in denen u enthalten ist, inzidieren gerade viele Kanten mit u.

3.2.1 a) Numeriere die Ecken eines regelmäßigen $2p$-Ecks im Uhrzeigersinn mit $-p, -(p-1), \ldots, -1, 1, \ldots, p$. Den ersten Hamiltonpfad bilden die Kanten $\{\{-i, i\} : 1 \leq i \leq p\}$ und $\{\{-i, i+1\} : 1 \leq i \leq p-1\}$. Die übrigen $p-1$ entstehen aus diesem durch Drehungen.
b) Schließe die Hamiltonpfade im K_{2p} über den $(2p+1)$-ten Knoten.

3.2.17 Durch Induktion nach d. Klar für $Q_2 \cong C_4$. Sei $v_1, \ldots, v_n$, $n = 2^d$, ein Hamiltonkreis in Q_d. Dann ist $(0, v_1), \ldots, (0, v_n), (1, v_n), \ldots, (1, v_1)$ ein Hamiltonkreis im Q_{d+1}.

3.2.19 a) [PS82] Füge zu G einen neuen Knoten w hinzu und verbinde ihn vollständig mit $\Gamma(u)$. Hefte nun u und w ein Blatt an.
b) Ersetze jede Kante durch einen Pfad der Länge zwei.

3.2.21 Nach Übung 2.2.7 ist G 2-zusammenhängend. Nach Satz 3.2.13 besitzt G einen Kreis C der Länge $\geq 2r$. Falls $|C| = 2r$, so sei $v \in V \setminus C$, $X := \Gamma(v) \subseteq C$ und $Y := C \setminus X$, d.h. es ist $|X| = r = |Y|$. Wenn es zwei auf C aufeinanderfolgende Knoten in X gibt, kann man C sofort vergrößern. Andernfalls wechseln sich die Knoten von X und Y auf C ab. Wegen $v \in \Gamma(x)$ für alle $x \in X$ ist $G[V-v]$ nicht der $K_{r,r}$, und es gibt benachbarte y_i und y_j in Y. Konstruiere daraus einen Hamiltonkreis wie in Beweis zu Satz 3.2.10.

3.2.24 a) Die untere Schranke folgt aus

$$\kappa(G) \leq \lambda(G) \leq \langle A, \bar{A} \rangle = \beta(G)|A||\bar{A}| \leq \beta(G)n^2/4$$

für ein entsprechend gewähltes A. Für die obere sei S eine minimum trennende Knotenmenge, also $s := |S| = \kappa(G)$, und A die kleinste Komponente von $G-S$, so daß $|A| \leq \frac{n-s}{2}$. Dann gilt wegen

$$|S||A| \geq \langle A, \bar{A} \rangle \geq \beta(G)|A|(n-|A|)$$

die Abschätzung

$$s \geq \beta(G)(n-|A|) \geq \beta(G)\left(n - \frac{n-s}{2}\right) = \beta(G)\frac{n+s}{2}.$$

b) Aus der oberen Schranke für $\beta(G)$ folgt $\kappa(G) \geq \frac{\beta(G)}{2-\beta(G)} \geq \alpha(G)$, und Satz 3.2.10 liefert die Behauptung.

3.2.25 Wende Korollar 3.2.8 auf den Graphen $G * K_1$ (einen neuen Knoten vollständig mit G verbinden) an.

3.2.26 a) Wie in Proposition 3.2.12. **b)** und **c)**: Füge G einen neuen Knoten hinzu und verbinde ihn mit u und v.

4.1.1 b) Keines oder genau eines. Denn sind M_1 und M_2 zwei perfekte Matchings in einem Baum T, so hat jeder Knoten im Graph $(V, M_1 \Delta M_2)$ den Grad 0 oder 2. Da T kreisfrei ist, folgt $M_1 \Delta M_2 = \emptyset$. **c)** Ordne die Knoten 1 bis $2p-1$ auf den Ecken eines regelmäßigen $(2p-1)$-Ecks mit dem Knoten 0 als Zentrum an. Jede der $2p-1$ „Speichen“ $\{0, i\}$ bildet mit den zu ihr orthogonalen Kanten ein perfektes Matching M_i in K_{2p}, und es gilt $\bigcup_{i=1}^{2p-1} M_i = E$ mit $M_i \cap M_j = \emptyset$.

4.1.10 Verbinde jeweils beide Endknoten von k P_4's mit zwei extra Knoten.

4.1.13 Sei M^* ein maximum Matching in G; es gilt also $|M^*| = \nu(G)$. Die symmetrische Differenz $M \Delta M^*$ der beiden Matchings ist ein Subgraph von G bestehend aus Pfaden und geraden Kreisen. Wegen $|M^*| = |M|+k$ enthält er genau k mehr Kanten aus M^* als aus M. Die Kreise und die Pfade gerader Länge (beachte: sie haben ungerade Ordnung) enthalten genauso viele Kanten aus M wie aus M^*. Ein Pfad ungerader Länge enthält genau eine Kante mehr von dem Matching, das seine Endknoten überdeckt, als von dem anderen – entsprechend ist er entweder M- oder M^*-augmentierend; da M^* jedoch maximum ist, ist jeder Pfad ungerader Länge in $M \Delta M^*$ schon M-augmentierend. Wegen $|M^*| = |M| + k$ gibt es mithin genau k M-augmentierende Pfade in $M \Delta M^*$. Da die Pfade und Kreise in $M \Delta M^*$ die Knotenmenge V partitionieren, ist die Summe der Knotenzahlen der k M-augmentierenden Pfade in $M \Delta M^*$ höchstens gleich n. Also gibt es mindestens einen M-augmentierenden Pfad in $M \Delta M^*$ auf höchstens $\lfloor n/k \rfloor$ vielen Knoten.

4.1.14 Proposition 4.1.4. Vergleiche auch Übung 3.2.25.

4.2.7 Sei $\{P_{2i+2} : i = 1, \ldots, k\}$ eine Menge von k knotendisjunkten Pfaden mit den ungeraden Längen $3, \ldots, 2k+1$, und sei $\{a, b\}$ eine dazu disjunkte Kante. Bei jedem der Pfade P_{2i+2}, $i = 1, \ldots, k$, sei jeweils der eine Endknoten mit a, der andere Endknoten mit b durch eine Kante verbunden. Der so definierte Graph G ist offenbar bipartit und zweifach zusammenhängend. Die erste Phase des HOPCROFT-KARP-Algorithmus konstruiert nun u.U. in jedem der Pfade P_{2i+2}, $i = 1, \ldots, k$, jeweils ein maximales Matching der Kardinalität i und matcht a mit b. Die kürzesten augmentierenden Pfade haben damit jeweils Länge $3, 5, 7, \ldots, 2k+1$, d.h. es werden insgesamt $k+1$ Phasen benötigt bei $n = 2 + \sum_{i=1}^{k}(2i+2) = k^2 + 3k + 2$ Knoten.

4.2.22 Wegen $0 \neq det\, A = \sum_{\sigma \in \mathcal{S}_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$ ist ein Summand ungleich Null.

4.2.23 Da jeder Knoten einer Knotenüberdeckung höchstens Grad n hat, gilt $\tau(G)n \geq m > (k-1)n$, also wegen Satz 4.2.15 $\nu(G) = \tau(G) \geq k$.

4.2.24 Für alle $S \subseteq V_1$ sei also $|\Gamma(S)| \geq |S| - d$ erfüllt. Füge V_2 d Knoten hinzu und verbinde sie vollständig zu V_1. Der resultierende Graph G' erfüllt dann die HALL-Bedingung. In G' existiert also ein Matching M, das alle Knoten von V_1 überdeckt. Mindestens $|M| - d$ Kanten von M gehören aber zu G, so daß gilt: $\nu(G) \geq |M| - d \geq |V_1| - d$. Wenn umgekehrt $S \subseteq V_1$ eine Knotenmenge ist mit $|\Gamma(G)| = |S| - d$, dann können mindestens d Knoten aus S nicht gematcht werden, und es folgt $\nu(G) \leq |V_1| - d$.

4.2.27 Konstruiere ein Matching in dem bipartiten Graphen, der die Menge der Variablen bzw. der Klauseln als Partitionsklassen hat, das alle Klauseln überdeckt.

4.2.29 a) Verbinde s vollständig mit U und t vollständig mit V und erhalte einen Graphen G_{st}. Dann entsprechen Matchings in G kreuzungsfreien $s-t$–Pfaden in G_{st} und Knotenüberdeckungen $s-t$–trennenden Knotenmengen.
b) „$\Rightarrow$“: Es ist zu zeigen, daß die HALL-Bedingung $(*)$ $|\Gamma(X)| \geq |X|$ für alle $X \subseteq U$ auch hinreichend ist für die Existenz eines Matchings, das ganz U überdeckt. Sei T eine minimale Knotenüberdeckung von G sowie $T_U := T \cap U$ und $T_V := T \cap V$. Es gilt also $|T| = \tau(G)$, und $(**)$ es gibt keine Kanten zwischen $U \setminus T_U$ und $V \setminus T_V$. Es folgt

$$\tau(G) - |T_U| = |T_V| \overset{(**)}{\geq} |\Gamma(U \setminus T_U)| \overset{(*)}{\geq} |U \setminus T_U| = |U| - |T_U|,$$

und mit dem Satz von KÖNIG $\nu(G) = \tau(G) \geq |U|$.
„$\Leftarrow$“: Hier ist nur noch $\nu(G) \geq \tau(G)$ zu zeigen. Sei T eine Knotenüberdeckung von G und T_i wie oben. Dann erfüllt der Graph $G_U := G[T_U \cup V \setminus T_V]$ die HALL-Bedingung, denn gäbe es ein $X \subseteq T_U$ mit $|\Gamma_{G_U}(X)| < |X|$, so wäre $(T \setminus X) \cup \Gamma_{G_U}(X) = (T \setminus X) \cup \Gamma(X)$ eine kleinere Knotenüberdeckung von G als T. Also gibt es nach dem Satz von HALL ein Matching M_U in G_U, daß ganz T_U überdeckt. Analog gibt es ein Matching M_V in $G_V := G[T_V \cup U \setminus T_U]$, daß ganz T_V überdeckt. Mithin ist $M := M_U \cup M_V$ ein Matching in G mit $|M| = |T_U| + |T_V| = |T| \geq \tau(G)$.

4.2.30 a) Satz 4.2.15 von KÖNIG in den Satz 4.1.9 von GALLAI einsetzen;
b) Für jeden dreiecksfreien Graphen G ohne isolierte Knoten gilt $\theta(G) = \rho(G)$.

4.2.31 Siehe [Lov79], Übung 7.7; [LP86], Theorem 4.1.1; oder [Aig93], Übung 7.20.

4.2.32 Sei $o : E \to V \times V$ die Orientierung von E gemäß einer Eulertour (vgl. Satz 3.1.2). Betrachte den bipartiten Graphen $B = (U_1, U_2, F)$ mit $U^{(1)}$ und $U^{(2)}$ Kopien von V und

$$\{u_i^{(1)}, u_j^{(2)}\} \in F \;\Leftrightarrow\; (v_i, v_j) \in o(E).$$

Da sich 1-Faktoren in B und 2-Faktoren in G bijektiv entsprechen, ist die Aussage äquivalent zu Korollar 4.2.11.

4.2.34 a) Angenommen, weder x noch y seien universal. Dann gibt es ein maximum Matching M_x, das x nicht trifft, und ein maximum Matching M_y, das y nicht trifft. Insbesondere gilt $e := \{x, y\} \notin M_x \cup M_y$. Wegen $|M_x| = \nu(G) = |M_y|$ besteht die symmetrische Differenz $M_x \Delta M_y$ aus geraden Kreisen und geraden Pfaden. Da die Matchings M_x und M_y maximal sind, wird y von M_x und x von M_y überdeckt; der Knoten x wie der Knoten y ist folglich jeweils Endpunkt eines (geraden) Pfades P_x bzw. P_y in $M_x \Delta M_y$. Offensichtlich gilt $P_x \neq P_y$, denn sonst enthielte der bipartite Graph G einen ungeraden Kreis. Durch Vertauschen der Kanten von M_x und M_y in einem der Pfade und Hinzunahme von e erhält man nun aber ein $\nu(G) + 1$-Matching in G, Widerspruch.
b) Falls $\nu(G) = 1$, so ist G ein Stern $K_{1,\Delta}$, und es gilt trivialerweise $\tau(G) \leq \nu(G)$. Sei andernfalls v ein universaler Knoten in G. Dann gilt

$$\tau(G) \;\leq\; |\{v\}| + \tau(G - v) \overset{(IV)}{\leq} 1 + \nu(G - v) \;=\; 1 + (\nu(G) - 1).$$

4.3.11 „$\Rightarrow$“: Der Satz von TUTTE liefert $q(G - v) \leq 1$. Wäre $q(G - v) = 0$ für ein $v \in V$, so wäre n ungerade.
„$\Leftarrow$“: Für alle $v \in V$ sei e_v die Kante, die v mit der ungeraden Komponente von $T - v$ verbindet. e_v ist eindeutig bestimmt, da T kreisfrei. Wir zeigen, daß $M := \{e_v |\ v \in V\}$ nun ein perfektes Matching in T ist. M überdeckt sicher alle Knoten. Die Unabhängigkeit der Kanten aus M sehen wir wie folgt ein. Sei $e_v = \{v, w\} \in M$. Die Komponente von $G - w$, die v enthält ist ungerade, denn sie enthält die geraden Komponenten von $G - v$ und v selbst. Also folgt $e_w = e_v$.

4.3.13 a) M ist nur eindeutig, wenn es auch das Matching im Graphen B der Behauptung 3 aus dem Beweis von Satz 4.3.1 ist. Nach Übung 4.3.12 gibt es eine Komponente C_i, von der nur eine Kante nach S führt. **b)** $G_1 := K_2$. G_{k+1} entsteht aus zwei Kopien von G_k, die jeweils vollständig zu einer zusätzlichen Kante (der Brücke) verbunden werden.

4.3.15 Zu zeigen ist: $2\nu(G) \geq n - d$ für Defekt $d \geq 1$. Definiere $H := G * K_d$, d.h. füge G eine d-Clique C hinzu und verbinde sie vollständig zu V. Dann hat G ein Matching, das alle bis auf d Knoten überdeckt genau dann, wenn H ein perfektes Matching besitzt. Um zu zeigen, daß H ein perfektes Matching besitzt, zeigen wir, daß H die TUTTE-Bedingung erfüllt. Wegen $d \equiv n \pmod 2$ folgt dies für $S = \emptyset$. Sei nun also $S \neq \emptyset$. *Fall 1:* $C - S \neq \emptyset$. Dann ist $H - S$ noch zusammenhängendund daher $q(G - S) \leq 1 \leq |S|$. *Fall 2:* $C \subseteq S$. Dann gilt nach Definition von d: $q(H - S) = q(G - (S \setminus C)) \leq d + |S \setminus C| = |S|$.

4.3.16 „$\Rightarrow$“ klar. Für „$\Leftarrow$“ ist nur noch $\nu(G) \geq \frac{n-1}{2}$ zu zeigen. Angenommen nun, $\nu(G) < \frac{n-1}{2}$, d.h. ein maximum Matching in G überdeckt mindestens zwei Knoten nicht. Seien M^* wie im Hinweis und x und y zwei von M^* nicht überdeckte Knoten mit $dist(x, y) = d_{M^*}$. Damit folgt $dist(x, y) \geq 2$, d.h. ein kürzester $x - y$–Weg in G enthält einen Knoten $z \notin \{x, y\}$. Das Matching M^* überdeckt z nach Wahl von x und y. Sei M' ein maximum Matching in $G - z$. Nach Voraussetzung gilt $|M^*| = |M'|$. Die symmetrische Differenz $M^* \Delta M'$ besteht also aus alternierenden geraden Pfaden und geraden Kreisen, vgl. Satz 4.1.6 von BERGE. Da z von M^*, nicht aber von M' überdeckt wird, beginnt in z ein alternierender gerader Pfad p. Vertauscht man im Matching M^* nun Matching- und Nicht-Matchingkanten entlang des Pfades p, gelang man also zu einem ebenfalls maximum Matching M, für das jedoch $d_M < d^*_M$ gilt, da M nun z und mindestens einen der Knoten x und y nicht überdeckt.

4.3.18 Für $\emptyset \neq R, S \subset V$ mit $R \cap S = \emptyset$ bezeichne $(R, S) := \{\{r, s\} : r \in R \wedge s \in S\}$. Sei $S \subset V$, C eine ungerade Komponente von $G - S$ und $m_C := m(G[C])$. Dann gilt

$$|(C, S)| \;=\; r|C| - 2m(G[C]) \;\equiv\; r|C| \;\equiv\; r \qquad \pmod 2.$$

Da G $(r-1)$-kantenzusammenhängend, gilt andererseits $|(C, S)| \geq r - 1$, und wir erhalten insgesamt, daß $|(C, S)| \geq r$. Es folgt

$$r \cdot q(G - S) \;\leq\; |(S, V \setminus S)| \;\leq\; r \cdot |S|,$$

und G erfüllt die TUTTE-Bedingung (4.9).

4.3.20 Spieler A hat eine Gewinnstrategie genau dann, wenn G *kein* perfektes Matching besitzt.

5.1.3 a) $\chi(G) \geq n/\alpha(G) \geq n/\theta(G)$. **b)** Wir induzieren nach n. Sei $|G| = n + 1$ und $v \in V$. Gilt $\chi(G - v) = \chi(G)$ oder $\chi(\overline{G - v}) = \chi(\overline{G})$, so folgt die Aussage sofort aus der Induktionsvoraussetzung. Im Falle

$$\begin{array}{rrcl} & \chi(G - v) & = & \chi(G) - 1 \\ \wedge & \chi(\overline{G - v}) & = & \chi(\overline{G}) - 1 \end{array}$$

folgt

$$\begin{array}{rrcl} & d_G(v) & \geq & \chi(G) - 1 \\ \wedge & d_{\overline{G}}(v) & \geq & \chi(\overline{G}) - 1, \end{array}$$

was sich summiert zu $n - 1 \geq \chi(G) + \chi(\overline{G}) - 2$.

5.1.5 Siehe [Har74], Sätze 12.16 und 12.17.

5.1.6 Vergleiche Proposition 1.1.19.

5.2.7 Man führt doppelt verkettete Listen $Grad[i]$, $i = 1, \ldots, \Delta(G)$, die jeweils die Knoten vom Grad i im Restgraphen führen. Ein weiteres Feld *Position* gibt für jeden Knoten v Auskunft über die Position von v in der Liste $Grad[d_{G_i}(v)]$. In jedem Schritt entfernt man nun einen Knoten aus der nicht-leeren Liste mit kleinstem Index und aktualisiert die Knotengrade im Restgraphen (ohne dabei den Restgraphen G_{i+1} zu berechnen!). Dann muß insgesamt höchstens $(\sum_{v \in V} d_G(v)) -$mal ein Knoten die Liste wechseln, was jeweils nur konstante Zeit kostet. Ebenso kostet die n-malige Berechnung des Index der ersten nicht-leeren Liste höchstens Zeit $\mathcal{O}(n + \sum_{v \in V} d_G(v)) = \mathcal{O}(n + m)$.

5.3.2 Angenommen, T_{k+1} wäre k-färbbar. Dann existiert nach dem Schubfachprinzip eine einfarbige n_k-elementige Teilmenge $U \subset R$. Die angehängte Kopie von T_k wäre mit $k - 1$ Farben gefärbt, Widerspruch.

5.4.2 Reduktion von 3-COLORABILITY durch Hinzufügen einer $(k-3)$-Clique.

5.4.4 Nach dem Satz 5.2.9 von BROOKS ist der einzige 4-chromatische Graph mit Maximalgrad 3 der K_4. Auch 1- oder 2-chromatische Graphen sind in polynomieller Zeit zu erkennen.

5.4.10 a) Zu einer Instanz G von 3-COLORABILITY konstruiere einen Graphen G' auf $n' := (k-2) \cdot n$ Knoten wie folgt. Setze $V_1 := V(G)$ und füge G (disjunkt) $n-3$ stabile Knotenmengen $V_2, \ldots, V_{k-2}$ jeweils der Kardinalität n hinzu. Verbinde die verschiedenen V_i's, $1 \leq i \leq k-2$, jeweils vollständig untereinander. Dann gilt $\delta(G') = (k-3)n = \frac{k-3}{k-2} n'$, und G' ist k-färbbar genau dann, wenn G 3-färbbar ist.
b) Verbinde jeden Knoten von G vollständig mit einem (eigenen) $K_{t,t}$ für ein $t \in \mathbb{N}$. Der entstehende Graph G' ist offenbar 3-färbbar genau dann, wenn es G ist. G' hat $n' = (1+2t)n$ Knoten und Minimalgrad $\delta(G') > t$. Um nun $t \geq n'^{1-\epsilon} = [(1+2t)n]^{1-\epsilon}$ zu gewährleisten, genügt es, $t \in \mathbb{N}$ mit

$$\begin{array}{rcl} t & \geq & [3tn]^{1-\epsilon} \\ \Leftrightarrow \quad t & \geq & \left([3n]^{1-\epsilon}\right)^{1/\epsilon} \end{array}$$

zu wählen. Für ϵ fest ist t und damit die Konstruktion von G' polynomiell in n.

5.6.16 Proposition 5.6.1.

5.6.17 Zusammen mit Gleichung (5.10) aus Satz 5.6.3.

5.6.18 Die von M überdeckten Knoten C bilden eine Knotenüberdeckung. Die Menge $V \setminus C$ ist also stabil. Überdeckt C alle Knoten vom Grad Δ in G, so besitzt $G - M$ eine Kantenfärbung mit $\Delta(G-M)+1 = \Delta(G)$ Farben, andernfalls bilden die Knoten vom Grad $\Delta(G)$ in $G - M$ eine stabile Menge und $G - M$ kann nach Korollar 5.6.10 mit $\Delta(G)$ Farben gefärbt werden. In jedem Fall verbleibt eine Farbe für M.

6.1.3 „$\Rightarrow$“: Angenommen, ein Gebiet eines ebenen Graphen wird nicht durch einen Kreis berandet. Dann trifft man beim Abfahren des Randes dieses Gebietes auf einen Knoten $a \in V$ ein zweites Mal. Die Knoten, denen man zwischen dem ersten und dem zweiten Auftreten von a begegnet ist, hat man dabei „umfahren“ - sie werden von dem abgefahrenen Gebietsrand in ein Gebiet der Ebene eingeschlossen, aus dem keine Kanten herausführen außer bei a. Also wäre der Knoten a eine Artikulation von G.
„$\Leftarrow$“: Dies folgt unmittelbar aus dem Satz über die Ohrenzerlegung zweifach zusammenhängender Graphen (Satz 2.3.1).

6.1.15 Korollar 5.2.6.

6.1.21 Sei S eine $\alpha(G)$-stabile Menge in G. Um Korollar 6.1.17 anwenden zu können, entferne alle Kanten in $G[V \setminus S]$. Der entstehende Graph ist dann bipartit. Für die Anzahl m' seiner Kanten gilt folglich $m' \leq 2n - 4$. Da keine mit S inzidierende Kanten entfernt wurden, gilt andererseits $m' \geq \alpha(G) \cdot \delta(G)$.

6.1.32 a) Ein beliebiger Knoten $v \in V$ einer Triangulation $G = (V, E)$ bildet mit seinen Nachbarn Dreiecke. Seine Nachbarn bilden also einen Kreis. $G \setminus v$ ist daher nach Proposition 6.1.3 stets ein 2-zusammenhängender, ebener Graph.
b) Jeder planare Graph ist Subgraph einer Triangulation, die nach Korollar 6.1.27 eine gradlinige Einbettung besitzt.

6.1.33 Der PETERSEN-Graph besitzt den K_5 als Minor, den $K_{3,3}$ als Unterteilung und hat nach Übung 6.1.20 zu viele Kanten für seine Taillenweite.

6.1.35 b) - d) Ein 2-zusammenhängender kreisartig planarer Graph ist wegen Übung 6.1.3 ein Kreis mit Sehnen. Also enthält er mindestens zwei Knoten vom Grad höchstens 2. Nach Korollar 5.2.6 ist er deshalb 3-färbbar. Da weiter jedes Innengebiet mindestens eine Kreiskante enthält, gibt es höchstens $n - 2$ Innengebiete oder $n - 3$ Sehen.
f) O.B.d.A. ist $\Delta(G) \geq 3$ und G zweifach zusammenhängend. Nach b) ist G Hamiltonsch. Falls G gerader Ordnung ist, besitzt G daher ein perfektes Matching M. Andernfalls - ein kreisartig planarer Graph enthält mindestens einen Knoten vom Grad ≤ 2 - besitzt G ein fast perfektes Matching M, das einen Knoten vom Grad $2 < \Delta(G)$ nicht überdeckt. In jedem Fall ist nach Annahme $G - M$ Klasse 1, also auch G, Widerspruch.

6.1.36 (FISK 1978) Trianguliere den Polygonzug und benutze Übung 6.1.35c.

6.4.15 Siehe [CL86], Theorem 4.28.

6.6.1 a) Übung 6.1.20 liefert für planare Graphen auf 10 Knoten mit Girth mindestens 5: $m \leq 13$. Also sind aus dem P_{10} mindestens 2 Kanten zu entfernen, um einen planaren Graphen zu erhalten.
b) Vervielfache im K_5 alle Kanten bis auf eine.

6.6.2 Einbettung: zwei Dreiecke ineinander;
$cr(K_6) \geq 3$: angenommen, es gäbe Einbettung mit nur 2 Kreuzungen.
(1) Dann gibt es einen Knoten, an dem zwei an Kreuzungen beteiligte Kanten inzidieren. Entferne diesen und erhalte Einbettung des K_5 in die Ebene, Widerspruch.
(2) Entferne die zwei überkreuzten Kanten und erhalten planaren Graphen. $E(K_6) - 2 = 13 > 3|V| - 6$, Widerspruch.
(3) Ersetze die Überkreuzungen durch Knoten und erhalte planaren Graphen. $19 = 15 + 2 + 2 \leq 3 \cdot (6 + 2) - 6 = 18$, Widerspruch.

6.6.10 a) Benutze Übung 6.1.35d; **b)** Benutze a) bzw. Übung 3.2.1a; **d)** Folgt mit c) aus Korollar 5.2.6.

6.3.6 a) Für $n = 3$, also $G \cong K_3$, ist die Behauptung trivial. Sei also $n \geq 4$. Falls der Kreis C eine Sehne hat, so läßt sich der Graph G entlang der Sehne in zwei kleinere Semi-Triangulationen zerlegen und deren Färbungen aus der Induktionsvoraussetzung zu einer von G zusammensetzen. Andernfalls ist $G - v_p$ eine Semi-Triangulation. Seien a und b zwei Farben aus $L(v_p) \setminus \{1\}$. Wende die Induktionsvoraussetzung auf $G - v_p$ an, wobei

die Farben a und b aus den Listen $L(u)$, $u \in \Gamma(v_p) \setminus C$, entfernt wurden. Die erhaltene Färbung läßt sich – je nach dem, welche Farbe v_{p-1} erhalten hat – mit a bzw. b auf v_p fortsetzen.
c) Vergleiche Übung 6.1.19b.

6.3.9 Siehe [Lov79], Übung 9.52, oder [Aig84], Satz 3.1.

6.3.20 Siehe [CL86], Abschnitt 6.2.

6.5.1 a) Nach Übung 5.1.5b bildet die Vereinigung zweier Farbklassen in einem eindeutig färbbaren Graphen einen zusammenhängenden Subgraphen. Daher zählt man in einem eindeutig 4-chromatischen planaren Graphen mit p_i, $1 \leq i \leq 4$, Knoten in der i-ten Farbklasse mindestens $\sum_{1 \leq i < j \leq 4}(p_i + p_j - 1) = 3n - 6$ Kanten.
b) Vier Farbklassen in dieser 5-Färbung induzieren einen eindeutig 4-färbbaren, planaren Graphen, dessen Einbettung nach a) eine Triangulation der Ebene ist. Es ist nun aber unmöglich, die Knoten der fünften Farbklasse der Einbettung hinzuzufügen, da jeder dieser Knoten aufgrund der Eindeutigkeit der Färbung mit mindestens einem Knoten aus jeder anderen Farbklassen verbunden sein muß.

6.5.2 Angenommen, G sei in die Ebene eingebettet und enthalte ein Gebiet mit Randkreis $v_1, \ldots, v_l$, $l \geq 4$, vgl. Proposition 6.1.3. Dann gibt es zwei nicht benachbarte Knoten v_i und v_j, $1 \leq i < j \leq l$. Identifiziere die beiden Knoten v_i und v_j im Graphen G. Der entstandene Graph G' ist ebenfalls planar und hat einen Knoten weniger. Aufgrund der Knotenminimalität von G ist G' 4-färbbar. Eine entsprechende Färbung Δ von G' ist aber auch eine 4-Färbung von G mit $\Delta(v_i) = \Delta(v_j)$, Widerspruch.

6.5.11 „$\Leftarrow$“: Nach Korollar 6.3.36 ist jede Eulersche Triangulation 3-färbbar. Also auch jeder ihrer Subgraphen. „$\Rightarrow$“: Sei G ein planarer Graph. G sei in die Ebene eingebettet und $c : \quad V \to \{1,2,3\}$ eine zulässige 3-Färbung seiner Knoten. Wieder nach Korollar 6.3.36 genügt es, zu zeigen, daß nun jedes Gebiet so trianguliert werden kann, daß die Färbung c zulässig bleibt. Sei G zunächst zweifach zusammenhängend. Dann ist der Rand eines jeden Gebietes R der Einbettung ein Kreis C in G. Wir induzieren nach $|C|$. Ohne Einschränkung enthalte C mehr als drei Knoten.
Fall 1. c ist eingeschränkt auf C eine Zweifärbung, d.h. die Knoten von C sind alternierend in zwei Farben gefärbt. Setze in das Innere des Gebietes R einen zusätzlichen Knoten v_C, verbinde alle Knoten auf C mit v_C und setze c auf $V + v_C$ fort mit der Farbe, die auf C nicht auftritt.
Fall 2. Es gibt auf C drei aufeinanderfolgende Knoten in den drei verschiedenen Farben. Verbinde den ersten und den dritten Knoten durch eine Kante. Der Randkreis des verbleibenden und noch zu triangulierenden Teils des Gebiets R enthält einen Knoten weniger als C und kann daher nach Induktionsvoraussetzung wie gewünscht trianguliert werden.
Angenommen nun, G ist nicht zweifach zusammenhängend. Falls G unzusammenhängend ist, kann man offenbar solange Kanten zwischen verschiedenen Komponenten hinzufügen (und ggf. die 3-Färbung anpassen), bis man einen zusammenhängenden 3-färbbaren planaren Supergraphen von G erhält. Sei G also o.B.d.A. 3-färbbar und planar mit Zusammenhangszahl $\kappa(G) = 1$. Falls G ein Baum ist, so verbinde man die Blätter durch einen Kreis (wobei nötigenfalls zwischen zwei Blättern ein neuer Knoten einzufügen ist), um einen zweifach zusammenhängenden 3-färbbaren planaren Supergraphen von G zu erhalten, der wie oben trianguliert werden kann. Andernfalls bette zunächst einen zweifach

zusammenhängenden Block B von G ein und trianguliere ihn. Nun bettet man schrittweise auch die anderen Blöcke von G in Reihenfolge einer Breitensuche auf dem Block-Artikulations-Baum $bc(G)$ mit Startknoten B ein, wobei die 3-Färbung der Blöcke jeweils am Artikulationsknoten aneinander anzupassen ist. Man sieht leicht ein, daß nach jedem Schritt trianguliert werden kann, so daß der eingebettete Graph stets eine Triangulation darstellt.

7.1.3 Für $k \geq 2$ ist $\chi(C_{2k+1}) = 3 > 2 = \omega(C_{2k+1})$ sowie $\chi(\overline{C_{2k+1}}) = \theta(C_{2k+1}) = k+1 > k = \alpha(C_{2k+1}) = \omega(\overline{C_{2k+1}})$.

7.1.4 $\chi(G) = \max\{\chi(G_1), \chi(G_2)\} = \max\{\omega(G_1), \omega(G_2)\} = \omega(G)$.

7.2.14 Das Lemma 7.2.5 von DIRAC garantiert für jeden chordalen Graphen, der nicht vollständig ist (in welchem Fall wir fertig wären), zwei nicht benachbarte Simplizialknoten. Von diesen kann nur einer in der Clique liegen. Ein solches PES läßt sich also sukzessive konstruieren. MAXIMUM-ADJAZENZ-SUCHE beispielsweise tut genau das, wenn sie nur mit der Clique startet.

7.2.18 Per Induktion nach n. Für $n = 2$ ist die Aussage trivial. Sei also $\mathcal{T} = \{T_i : i = 1, \ldots, n\}$, $n \geq 3$. Nach Induktionsannahme gibt es ein $u \in \bigcap_{i=1}^{n-1} T_i$. Falls nicht auch $u \in T_n$, so sei v der eindeutig bestimmte erste Knoten in T_n auf einem $u - T_n$–Pfad in T. Wir behaupten, daß auch $v \in T_i$ für alle $i = 1, \ldots, n-1$. Nach Voraussetzung gibt es zu $i \in \{1, \ldots, n-1\}$ einen Knoten $s_i \in T_i \cap T_n$. Da auch $u \in T_i$ ist, liegt der gesamte $u - s_i$–Pfad im Baum T_i. Wegen $s_i \in T_n$ liegt v auf diesem $u - s_i$–Pfad, und mithin gilt $v \in T_i$.

9.1.16 Benutze Proposition 9.1.2.

10.2.2 Da T durch eine Tiefensuche entsteht, ist jeder Nachbar eines Blattes b von T ein Vorfahr von b in T. Also ist die Menge der Blätter von T unabhängig in G und NB somit eine Knotenüberdeckung.
Sei r der Startknoten der Tiefensuche gewesen. Ein Knoten $u \in \Gamma(v)$ heißt ein Sohn von v, falls v der erste Knoten auf dem eindeutigen $u - r$–Pfad in T ist. Jedes Nicht-Blatt $v \in NB$ besitzt offenbar einen Sohn. Mithin bildet die Menge

$$M := \{\{u, v\} \in E(T) : (v \in S \cap NB) \wedge (u \text{ ist ein Sohn von } v)\}$$

ein Matching in T, und mit $\bar{S} := V \setminus S$ folgt schließlich

$$\tau(G) \geq |M| = |S \cap NB| = |NB \setminus \bar{S}| \geq |NB| - |\bar{S}| = |NB| - \tau(G).$$

10.3.3 Wähle s_1 als Blatt. Dann endet die Konstruktion von T_Z in einem benachbarten Blatt $s_{|S|}$. Wähle s_1 und $s_{|S|}$ so, daß der Abschnitt zwischen s_1 und $s_{|S|}$ der „schwerste“ unter allen b Abschnitten auf Z zwischen zwei Blättern ist (und damit Gewicht mindestens $c(Z)/b$ hat).

10.4.1 Sei G eine Instanz von HAMILTONIAN CIRCUIT. Zu G definiere eine ΔTSP-Instanz (K_n, c), $n = |V(G)|$, durch die folgende Gewichtung der Kanten $e \in \binom{V(G)}{2}$ des K_n:

$$c(e) := \begin{cases} 1 & \text{falls } e \in E(G), \\ 2 & \text{falls } e \notin E(G). \end{cases}$$

Dann gilt: G besitzt einen Hamiltonkreis genau dann, wenn (K_n, c) eine TSP-Tour der Länge $\leq n$ besitzt.

10.4.4 Sei M ein Matching minimalen Gewichts zwischen den ungeraden Knoten in T und C^* eine kürzeste TSP-Tour in (K_n, c). Ferner seien $v_{i_1}, \ldots, v_{i_{2k}}$ die Knoten ungeraden Grades in T in einer Reihenfolge, wie sie in C^* auftreten. Für die Matchings $M_1 := \{\{v_{i_1}, v_{i_2}\}, \{v_{i_3}, v_{i_4}\}, \ldots, \{v_{i_{2k-1}}, v_{i_{2k}}\}\}$ und $M_2 := \{\{v_{i_2}, v_{i_3}\}, \{v_{i_4}, v_{i_5}\}, \ldots, \{v_{i_{2k}}, v_{i_1}\}\}$ gilt dann offensichtlich

$$2c(M) \leq c(M_1) + c(M_2) \leq c(C^*),$$

so daß für die gemäß Lemma 10.4.2 aus $T \cup M$ erhaltenen TSP-Tour C in (K_n, c) folgt

$$c(C) \;\leq\; c(T) + c(M) \;<\; c(C^*) + \frac{1}{2}c(C^*) \;=\; \frac{3}{2}c(C^*).$$

10.7.2 Die Reduktion von MAX-2-SAT auf MAX CUT aus 10.6.1 zeigt, daß das Problem, eine maximum Bisektion zu finden, NP-vollständig ist. Ein Graph der Ordnung $n = 2\ell$ besitzt aber eine Bisektion der Größe $\geq k$ genau dann, wenn sein Komplement eine Bisektion der Größe $\leq \ell^2 - k$ besitzt.

10.9.20 Für k und r definiert durch $n = ka + r$, $0 \leq r < a$, hat $S(n,a)$ offenbar $m(S(n,a)) = r\binom{k+1}{2} + (a-r)\binom{k}{2}$ viele Kanten. Offenbar ist $S(n,a)$ kantenminimal mit der Eigenschaft, Stabilitätszahl a zu haben. Für feste Kantenzahl m minimiert die Gradsequenz von $S(n,a)$ den Term $\sum_{v\in V} \frac{1}{d(v)+1} = a$. Also hat jeder andere Graph mit derselben Anzahl von Kanten, aber einer anderen Gradsequenz als $S(n,a)$ nach Gleichung (i) aus Proposition 1.5.6 eine echt größere Stabilitätszahl und insbesondere hat jeder Graph auf weniger Kanten eine größere Stabilitätszahl; also ist $S(n,a)$ ein extremaler Graph.

Für die Eindeutigkeit bleibt zu zeigen, daß jeder Graph G mit derselben Gradsequenz wie $S(n,a)$ und $\alpha(G) = \sum_{v\in V} \frac{1}{d(v)+1} = a$ aus disjunkten Cliquen besteht, d.h. keinen $K_{1,2}$ enthält. Wieder nach Gleichung (i) aus Proposition 1.5.6 gilt, daß GREEDY_MIN auf *jeder* Anordnung der Knotenmenge eine maximum Clique liefert. Wenn ein Graph nun aber einen $K_{1,2}$ enthält, sagen wir mit dem Knoten x in der einen und den Knoten y und z in der anderen Partitionsklasse, so liefert GREEDY_MIN, angewandt auf die Ordnung $x, y, z, v_3, \ldots, v_n$ eine kleinere stabile Menge als angewandt auf $y, z, x, v_3, \ldots, v_n$.

12.2.2 Aus $G = (V, E)$ konstruiere einen Graphen H wie folgt: füge zu $V = \{v_1, \ldots, v_n\}$ neue Knoten $x, u_1, \ldots, u_n$ hinzu und verbinde x mit allen anderen Knoten.

14.1.2 Aus $n - 1 \geq \sum_{u\in\Gamma(v)} d(u)$ erhält man mit Hilfe der CAUCHY-SCHWARZ-schen Ungleichung

$$\begin{aligned} n(n-1) \;&\geq\; \sum_{v\in V} \sum_{u:\,\{u,v\}\in E} d(u) \;=\; \sum_{u\in V} d(u)^2 \\ &\geq\; \frac{1}{n}\left(\sum_u d(u)\right)^2 \;=\; \frac{4}{n}|E|^2. \end{aligned}$$

A.3 Die Grenze zwischen P und NP-vollständig

Die folgende kleine Auswahl von kombinatorischen Optimierungsproblemen mag illustrieren, welch unterschiedliche Phänomene Einfluß darauf haben, ob ein Problem schwer (vermutlich unzugänglich) oder polynomiell lösbar ist; seien es Einschränkungen an die Struktur der Problemeingaben, feste oder laufende Problemparameter, der Unterschied zwischen dem Maximierungs- und Minimierungsproblem oder der Unterschied zwischen analog formulierten Problemen einmal für die Kanten, das andere Mal für die Knoten eines Graphen.

polynomiell lösbar	**NP-schwer**
2-SAT	3-SAT
INDEPENDENT-SET ($\Delta(G) \leq 2$)	INDEPENDENT-SET ($\Delta(G) \leq 3$)
BIPARTITE-INDEPENDENT-SET	TRIANGLE-FREE-INDEPENDENT-SET
2-COLORABILITY	TRIANGLE-FREE-3-COLORABILITY
3-COLORABILITY ($\Delta(G) \leq 3$)	3-COLORABILITY ($\Delta(G) \leq 4$)
4-CONN.-PLANAR-HAMILTONIAN-CIRCUIT	3-CONN.-PLANAR-HAMILTONIAN-CIRCUIT
CLIQUE (k fest)	CLIQUE (k Teil der Eingabe)
MINIMUM k-CUT (k fest)	MINIMUM k-CUT (k Teil der Eingabe)
PLANARITY	GENUS
MIN CUT	MAX CUT
SHORTEST CIRCUIT (GIRTH)	LONGEST CIRCUIT (CIRCUMFERENCE)
MAXIMUM MATCHING	MINIMUM MAXIMAL MATCHING
MAXIMUM MATCHING	INDEPENDENT SET
EDGE-COVER	NODE-COVER
EULERIAN CYCLE	HAMILTONIAN CIRCUIT
CHINESE-POSTMAN-PROBLEM	TRAVELING-SALESMAN-PROBLEM

Bemerkung. Im Fall von SPANNING TREE bzw. BISECTION sind jedoch sowohl Minimierungs- wie Maximierungsproblem polynomiell lösbar bzw. NP-vollständig.

A.4 Übersicht über Graphenparameter

Knotenzahl	n	$\|V\|$
Kantenzahl	m	$\|E\|$
maximaler Grad	$\Delta(G)$	$\max_{v\in V}\ d(v)$
minimaler Grad	$\delta(G)$	$\min_{v\in V}\ d(v)$
# Zh.komponenten	$c(G)$	
# ungerader Komponenten	$q(G)$	
Taillenweite	$g(G)$	Länge eines kürzesten Kreises in G
Radius	$rad(G)$	$\min_{v\in V}\max_{u\in V-v} d(u,v)$
Durchmesser	$diam(G)$	$\max_{u,v\in V} d(u,v)$
Zusammenhangszahl	$\kappa(G)$	$\max\{k:\ \forall u\neq v\ \exists k$ kreuzungsfreie $u-v$−Pfade$\}$
Kanten-Zhs.zahl	$\lambda(G)$	$\max\{k:\ \forall u\neq v\ \exists k$ kantendisjunkte $u-v$−Pfade$\}$
Schnittdichte	$\beta(G)$	$\min\left\{\frac{\langle A,\bar{A}\rangle}{\|A\|\|\bar{A}\|}:\ A\subset V,\ \emptyset\neq A\neq V\right\}$
Matchingzahl	$\nu(G)$	$\max\{\|M\|:\ M\subseteq E\ \wedge\ \forall e,f\in M(e\cap f=\emptyset)\}$
Unabhängigkeitszahl	$\alpha(G)$	$\max\{\|S\|:\ S\subseteq V$ stabil in $G\}$
Cliquenzahl	$\omega(G)$	$\max\{\|C\|:\ C\subseteq V$ Clique in $G\}$
Knotenüberdeckungszahl	$\tau(G)$	$\min\{\|C\|:\ C\subseteq V\ \wedge\ \forall\{u,v\}\in E(u\in C\vee v\in C)\}$
Kantenüberdeckungszahl	$\rho(G)$	$\min\{\|F\|:\ F\subseteq E\ \wedge\ \forall v\in V\ \exists e\in F\ (v\in e)\}$
Dominanzzahl	$\sigma(G)$	$\min\{\|D\|:\ D\subseteq V\ \wedge\ \forall v\in V\setminus D\ (\Gamma(v)\cap D\neq\emptyset)\}$
chromatische Zahl	$\chi(G)$	$\min\{k\mid \exists f:V\to\{1,\ldots,k\}$ mit $f(u)\neq f(v)\forall\{u,v\}\in E\}$
chromatischer Index	$\chi'(G)$	$\chi(L(G))$
Cliquenpartitionszahl	$\theta(G)$	$\chi(\overline{G})$
Cliquenüberdeckungszahl	$\theta_e(G)$	$\min\{k:\ \exists C_1,\ldots,C_k$ mit $C_i\subseteq V$ Clique und $E\subseteq\bigcup_{i=1}^{k}E(G[C_i])\}$
Schnittzahl	$\iota(G)$	$\min\{\|\bigcup_{v\in V}S_v\|:\ (S_v)_{v\in V}$ Schnittgraph-Darstellung von $G\}$
(orientierbares) Geschlecht	$\gamma(G)$	$\min\{h:\ G$ ist in die S_h einbettbar$\}$
nicht-orient. Geschlecht	$\overline{\gamma}(G)$	$\min\{h:\ G$ ist in die N_h einbettbar$\}$
Kreuzungszahl	$cr(G)$	minimal mögliche Anzahl von Überkreuzungen, wenn man G in die Ebene zeichnet
Dicke	$th(G)$	$\min\{k:\ G=\bigcup_{i=1}^{k}P_i\ \wedge\ P_i$ planar$\}$
Arborizität	$a(G)$	$\min\{k:\ G=\bigcup_{i=1}^{k}W_i\ \wedge\ W_i$ Wald$\}$

A.5 Weiterführende Literatur

Wo wir Forschungsergebnisse nur erwähnen konnten, haben wir an Ort und Stelle auf die entsprechende Fachliteratur hingewiesen. Hier sind daher eher Monographien und Übersichtsartikel zusammengetragen, die als Ausgangspunkt für ein vertiefendes Studium der behandelten Themen bzw. als Orientierung in angrenzenden Forschungsgebieten dienen mögen.

Zur Einordnung der Graphentheorie in die Diskrete Mathematik siehe [Aig79, Jac83, Big85, GKP89, LW92a, Aig90, Aig93, Bry93, Cam94, GGL95].

Über die Graphentheorie im allgemeinen sind nun schon viele Bücher geschrieben worden. Besonders verweisen möchten wir hier nur auf [Die96] sowie [Bol78a, Bol79, Ber85, CL86, Hal89, Zyk90, Wes95] und [CM78]. Stärker auf die algorithmische Graphentheorie heben [Jun94, TS92] ab. Einen hervorragenden Überblick zu Graphenalgorithmen gibt [Lee90]. Eine ständig aktualisierte Übersicht in Tabellenform zu Algorithmen für verschiedene Probleme auf speziellen Graphenklassen findet man auf der WWW-Seite [SX96].

Die erste Adresse in Sachen Entwurf und Analyse effizienter Algorithmen und Datenstrukturen ist [CLR90]. Empfehlenswert sind des weiteren die z.T. bereits klassischen Texte [Knu73, AHU74, PS82, SDK83, Tar83, Meh84, Meh88, GK90, Wil90, Kin90, Kuc90, OW93, LW92b, Koz92, HSA93, Wei95]. Eine C++-Bibliothek für effiziente Datenstrukturen und (insbesondere Graphen-) Algorithmen, LEDA, erhält man über anonymous-ftp von „ftp.mpi-sb.mpg.de“, Verzeichnis „/pub/LEDA“; vergleiche ebenso [Ski90, Knu93].

Der Klassiker auf dem Gebiet der Komplexitätstheorie ist [GJ79], aktualisiert durch [Joh90a]. Eine gelungene modernere Darstellung der Komplexitätstheorie, wie sie für die kombinatorische Optimierung relevant ist, ist [Pap94]. Umfassende Behandlungen von Komplexitätstheorie bzw. Berechenbarkeit findet man in [Rog67, HU79, DW83, WW86, BDG88, Bör92, Weg93], siehe auch [Joh90b, Rei90, Sch92, BC94].

Zu speziellen Themenkreisen der Graphentheorie verweisen wir auf:

- graphische Darstellung [BETT94]
- Flüsse in Netzwerken [FF62, PS82, SDK83, TS92, GTT90, Ruh91, Gus92, AMO93, KMN93, KN93], Zusammenhang [LLW88, LSS89, Fra95, HK95], Network Design und Network Reliability [Sto92, GMS95, AKR95, WN87, Fra92, FIN93, GSS93b, HR93, KT93, Fra94b, KV94, Jor95, DH95, Col87, BCP95], disjunkte Pfade [Fra90, RS85b, RS95, RWW95] und Distanz [BCN89, BH90, COS91]
- Matching [LP86, Ger95, Pul95] sowie [Law76a, PS82, Sch83, SDK83, Avi83, Gal86, Der88, GGS89, GI89, Gab90, GH90, GT91, GW95, Plu93a, Blu90, DM91, Vaz94] und Transversaltheorie [Mir71b, Bru75, Jun82, Aig79, Rei84]
- Eulersche Graphen [Fle83, LO86, Fle90], Hamiltonsche Graphen [Ber78, Chv85, Gou91, Vol96] und Kreise [WV74, AG85, Vos91, Bon95]
- Knotenfärbung [SDK83, HW89a, Kuc89, WN90, Wer90, SS89, Tuz92, PS92b, HS94, Prö94, Tof95], probabilistische Methode [ASE92], Kantenfärbung [FW77, FW78, Yap86, HW89b, NZN95], Verallgemeinerungen [Che91, Rob91, Alo93, Yap96] sowie offene Probleme [JT95]

- Planare Graphen und Graphen höheren Geschlechts [NC88, Nis90, Rin74, WB78, Whi84, GT87, WB89, Tho94a, Arch95, Yan96, MT97] sowie Minoren [RS85a, RS85b, FL94, Tho95]
- 4-Farben-Vermutung und -Satz [BLW76, Ore67, RT88, SK77, Kau90, WW78, AH78, Bar83b, Aig84, AH86, AH89, FF94] HADWIGER Vermutung [Tof96] und nirgendsverschwindende Flüsse [Jae88, Sey95]
- Perfekte Graphen [Gol80, BC84, Lov83, GLS88, Knu94, BP93, Möh85, Fis85, GS93, Bog94, MP95]
- Baumweite [PS90, Bod93b, Klo94, BL95]
- randomisierte Algorithmen [Joh84b, Rag90, Kar91a, GSB94, MR95a, Lov95] und Derandomisierung [MNN94, MR95b, LW95a]
- Approximationsalgorithmen [Hoc96] sowie [PS82, Mot92, CK95, ACP95, Shm95, KMSV95, BG96, Blu94, KMS94, MNR96, KT94b, KT95, PT95b, Rad95, Shm96, GW96] und Nicht-Approximierbarkeit [Gol93, HPS94, Pap94, Aro94, Bab94, BGS95, Sud95, AL96, Bel96]
- Dominanz [HL90], Cliquenüberdeckung/Schnittgraphen [Pul83a, Rob85, Sch85, Gol88] und Linegraphen [HB78]
- Zufällige Graphen [Tin80, Bol85, Pal85, Bol91, ASE92, Kar95] und probabilistische Algorithmen [KLMR85, Rin87, Fri89, Fri90, KMN93, Mot94, FRS95, CL91b]
- Extremale Graphentheorie [Bol78a, Sim83, Sim84, Für91, Bol95, PA95, KS96b]

Stärker anwendungsbezogene Graphentheorie rechnet man meist der Operations Research [BM87, BM91, NM93, KL95, AMOR95, DD95] und der Kombinatorischen Optimierung [Law76a, PS82, SDK83, ELR85, LLRS85, LP86, Iba87, GLS88, PR88, HRW92, Lap92b, Ree93, GL95a, JRR95] zu.

Für Anwendungen der Graphentheorie in den Natur-, Ingenieur- und Sozialwissenschaften siehe [Kas67, Rob78, Rob93, CGM79, WB79, Wil82, MR84, Wal84, Vag85, JS89a, Len90, BR91a, Fou92, GSS93a].

Weiterführende Literatur zu nur angeschittenen oder gar nicht behandelten, aber verwandten Themen:

- parallele Graphenalgorithmen [GR88, KR90, May90, JaJ92, Cha92, Rei93, Smi93]
- gerichtete Graphen [TS92, Har74, CL86]
- (ganzzahlige) lineare Optimierung und polyedrische Kombinatorik [PS82, Pul83b, Sch86, NW88, GLS88, Kar91b, Zie95, Pad95, Pan95, JRT95, Sch95]
- algebraische Graphentheorie, Eigenwerte [LW92a, TS92, Big74, SW78, CDS80, CDGT88, BR91b, MP93b, BCN89], Expander [Lub94, Lub95, Kah95] sowie Graphen und Gruppen [Whi84, Bab95]
- Matroid-Theorie [Wel76, Aig79, Kun86, Whi87, Rec89, Ox92, BC95]

- Erzeugen und Zählen kombinatorischer Strukturen [Sin93, Wel93, AS95] und PÓLYA-Theorie [Rys63, HP73, Min78, Stan86, PR87, Slo91]
- unendliche Graphen [Wag70, Tho84a, Hal89, Zem96]
- RAMSEY-Theorie [GRS80, Nes95]
- Hypergraphen [Bol86, Ber89, Für88, Für91, Duch95, PA95]
- Codes und Designs [BJL85, Hal67a, CL91a]
- teilweise Ordnungen, Verbandstheorie [Bir67, Aig79, Riv85, Tro92]

Literaturverzeichnis

Referenzen sind (ohne Unterscheidung von Groß- und Kleinbuchstaben) alphabetisch sortiert nach einem Kürzel, das sich aus den ersten drei Buchstaben des Namens des Autors bzw. den Initialen bei mehreren Autoren sowie den letzten zwei Ziffern der Jahreszahl zusammensetzt. Es werden die folgenden gängigen Abkürzungen benutzt.

ACM	Association for Computing Machinery
AMS	American Mathematical Society
DMV	Deutsche Mathematiker Vereinigung
FoCS	(IEEE) Symposium on the Foundations of Computer Science
IEEE	The Institute of Electrical and Electronics Engineers
LMS	London Mathematical Society
LNCS	(Springer) Lecture Notes in Computer Science
LNM	(Springer) Lecture Notes in Mathematics
SIAM	Society of Industrial and Applied Mathematics
SoDA	(ACM-SIAM) Symposium on Discrete Algorithms
SToC	(ACM) Symposium on the Theory of Computing

[Abe64] Aberth, O., *On the sum of graphs,* Revue Fr. Rech. Opér. 33, 1964, 353-358.

[ABL95] Archdeacon, D., Bonnington, C.P., Little, C.H.C., *An algebraic characterization of planar graphs,* J. Graph Theory 19, 1995, 237-250.

[ABY63] Auslander, L., Brown, T.A., Youngs, J.W.T., *The imbedding of graphs in manifolds,* J. Math. Mech. 12, 1963, 629-634.

[AC81] Ainouche, A., Christofides, N., *Conditions for the existence of hamiltonian circuits in graphs based on vertex degree,* J. LMS 32, 1985, 385-391.

[ACP87] Arnborg, S., Corneil, D.G., Proskurowski, A., *Complexity of finding embeddings in a k-tree,* SIAM J. Alg. Disc. Meth. 8, 1987, 277-284.

[ACP95] Ausiello, G., Crescenzi, P., Protasi, M., *Approximate solution of NP optimization problems,* Theoret. Comput. Sci. 150, 1995, 1-55.

[ADFS95] Aronson, J., Dyer, M., Frieze, A., Suen, S., *Randomized greedy matching II,* Random Struct. Alg. 6, 1995, 55-73.

[ADLRY94] Alon, N., Duke, R.A., Lefmann, H., Roedl, V., Yuster, R., *The algorithmic aspects of the regularity lemma,* J. Alg. 16, 1994, 80-109.

[AFLM91] Abrahamson, K., Fellows, M.R., Langston, M.A., Moret, B.M.E., *Constructive complexity,* Disc. Appl. Math. 34, 1991, 3-16.

[AFW95] Alon, N., Frieze, A., Welsh, D., *Polynomial time randomized approximation schemes for Tutte-Gröthendieck invariants: the dense case,* Random Struct. Alg. 6, 1995, 459-478.

[AG85] Alspach, B.R., Godsil, C.D., (Hrsg.), *Cycles in graphs,* Ann. Disc. Math. 27, North-Holland, Amsterdam, 1985.

[AH77] Appel, K., Haken, W., *Every planar map is four colorable,* Part I: Discharging, Ill. J. Math. 21, 429-490; in [AH89] enthalten.

[AH78] Appel, K., Haken, W., *The four color problem*, in: Mathematics today, Hrsg. Steene, L.A., Springer, Berlin, 1978, 153-180.

[AH86] Appel, K., Haken, W., *The four color proof suffices*, Math. Intell. 8, 1986, 10-20.

[AH87] Adleman, L.M., Huang, M.A., *Recognizing primes in random polynomial time*, Proc. 19th ACM SToC, 1987, 462-469.

[AH89] Appel, K., Haken, W., *Every planar map is four colorable*, AMS, Providence RI, 1989.

[AHK77] Appel, K., Haken, W., Koch, J., *Every planar map is four colorable,* Part II: Reducibility, Ill. J. Math. 21, 491-567; in [AH89] enthalten.

[AHU74] Aho, A.V., Hopcroft, J.E., Ullman, J.D., *The design and analysis of computer algorithms*, Addison-Wesley, Reading MA, 1974.

[AI77] Aoshima, K., Iri, M., *Comments on F. Hadlock's paper: finding a maximum cut of a planar graph in polynomial time,* SIAM J. Comput. 6, 1977, 86-87.

[Aig79] Aigner, M., *Combinatorial Theory*, (Überarbeitung der deutschen zweibändigen Ausgabe *Kombinatorik* I/II, Springer, Berlin, 1975/1976), Springer-Verlag, Berlin, 1979.

[Aig84] Aigner, M., *Graphentheorie*, Teubner, Stuttgart, 1984; auch als: *Graph theory: a development from the 4-color-problem*, BCS Assocs., 1987.

[Aig90] Aigner, M., *Diskrete Mathematik*, in: Ein Jahrhundert Mathematik 1890-1990, Festschrift zum Jubiläum der DMV, Hrsg. Scharlau, W., u.a., Vieweg, Braunschweig, 1990, 83-112.

[Aig93] Aigner, M., *Diskrete Mathematik*, Vieweg, Braunschweig/Wiesbaden, 1993.

[AIS87] Awerbuch, B., Israeli, A., Shiloach, Y., *Finding Euler circuits in logarithmic parallel time,* in: Parallel and distributed computing, Hrsg. Preparata, F.P., Advances in computing research, Vol. 4, JAI Press, Greenwich, 1987, 69-78.

[AK89] Aarts, E., Korst, J., *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing,* Wiley & Sons, Chichester, 1989.

[AKK95] Arora, S., Karger, D., Karpinski, M., *Polynomial time approximation schemes for dense instances of $\mathcal{NP}$-hard problems,* Proc. 27th ACM SToC, 1995, 284-293.

[AKR95] Agrawal, A., Klein, P., Ravi, R., *When trees collide: an approximation algorithm for the generalized Steiner problem on networks,* SIAM J. Comput. 24, 1995, 440-456.

[AKS80] Ajtai, M., Komlós, J., Szemerédi, E., *A note on Ramsey numbers,* J. Combin. Theory A 29, 1980, 354-360.

[AL96] Arora, S., Lund, C., *Hardness of approximations,* in: Approximation algorithms for NP-hard problems, Hrsg. Hochbaum, D., PWS Publ., 1996.

[Ali95] Alizadeh, *Interior point methods in semidefinite programming with applications to combinatorial optimization,* SIAM J. Optim. 5, 1995, 13-51.

[ALMSS92] Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M., *Proof verification and hardness of approximation problems*, Proc. 33rd IEEE FoCS, 1992, 14-23.

[Alo86] Alon, N., *Eigenvalues and expanders*, Combin. 6, 1986, 83-96.

[Alo90] Alon, N., *Transversal numbers of uniform hypergraphs*, Graphs Combin. 6, 1990, 1-4.

[Alo93] Alon, N., *Restricted colorings of graphs*, in: Surveys in combinatorics 1993, Hrsg. Walker, K., LMS Lecture Notes Series Vol. 187, Cambridge Univ. Press, 1993.

[Alo96] Alon, N., *Bipartite subgraphs*, erscheint in Combinatorica.

[ALS91] Arnborg, S., Lagergren, J., Sleese, D., *Easy problems for tree-decomposable graphs*, J. Alg. 12, 1991, 308-340.

[AMO93] Ahuja, R.K., Magnanti, T.L., Orlin, J.B., *Network flows: theory, algorithms, and applications,* Prentice-Hall, Englewood Cliffs NJ, 1993.

[AMOR95] Ahuja, R.K., Magnanti, T.L., Orlin, J.B., Reddy, M.R., *Applications of network optimization,* in: Network models, Handbooks in OR & MS, Vol. 7, Hrsg. Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L., Elsevier, Amsterdam, 1995, 1-83.

[AMOT90] Ahuja, R.K., Mehlhorn, K., Orlin, J.B., Tarjan, R.E., *Faster algorithms for the shortest path problem,* J. ACM 37, 1990, 213-223.

[ANS80] Akiyama, T., Nishizeki, T., Saito, N., *NP-completeness of the Hamiltonian cycle problem for bipartite graphs,* J. Inform. Process. 3, 1980, 73-76.

[AP61] Auslander, L., Parter, S.V., *On imbedding graphs into the plane,* J. Math. Mech. 10, 1961, 517-523.

[AP86] Arnborg, S., Proskurowski, A., *Characterization and recognition of partial 3-trees,* SIAM J. Alg. Disc. Meth. 7, 1986, 305-314.

[AP89] Arnborg, S., Proskurowski, A., *Linear time algorithms for NP-hard problems restricted to partial k-trees,* Disc. Appl. Math. 23, 1989, 11-24.

[APC90] Arnborg, S., Proskurowski, A., Corneil, D.G., *Forbidden minors characterization of partial 3-trees,* Disc. Math. 80, 1990, 1-19.

[Arch90] Archdeacon, D., *The complexity of the graph embedding problem,* in: Topics in combinatorics and graph theory, Hrsg. Bodendiek, R., Henn, R., Physica-Verlag, Heidelberg, 1990, 59-64.

[Arch95] Archdeacon, D., (Hrsg.), *Problems in topological graph theory,* University of Vermont, August 1995, 67 Seiten, erhältlich über WWW als „http://www.emba.uvm.edu/∼archdeac/problems.ps.Z".

[Aro94] Arora, S., *Probabilistic checking of proofs and hardness of approximation problems,* Überarbeitung der Dissertation bei U.V. Vazirani, UC Berkeley, 1994; erhältlich über WWW an der URL „http://www.eccc.uni-trier.de/eccc-local/ECCC-Books/eccc-books.html".

[Aro96] Arora, S., *Polynomial time approximation schemes for Euclidean TSP and other geometric problems,* Proc. 37th IEEE FoCS, 1996.

[AS95] Alonso, L., Schott, R., *Random generation of trees,* Kluwer Acad. Publ., Boston, 1995.

[ASE92] Alon, N., Spencer, J.H., Erdös, P., *The probabilistic method,* Wiley & Sons, New-York, 1992.

[AST90a] Alon, N., Seymour, P., Thomas, R., *A separator theorem for nonplanar graphs,* J. AMS 3, 1990, 801-808.

[AST90b] Alon, N., Seymour, P., Thomas, R., *A separator theorem for graphs with an excluded minor and its applications,* Proc. 22nd ACM SToC 1990, 293-299.

[AST94] Alon, N., Seymour, P., Thomas, R., *Planar separators,* SIAM J. Disc. Math. 7, 1994, 184-193.

[Avi83] Avis, D., *A survey of heuristics for the weighted matching problem,* Networks 13, 1983, 475-493.

[AYZ95] Alon, N., Yuster, R., Zwick, U., *Color-coding,* J. ACM 42, 1995, 844-856.

[Azu67] Azuma, K., *Weighted sums of certain dependent random variables,* Tôhoku Math. J. 3, 1967, 357-367.

[Bab94] Babai, L., *Transparent proofs and limits to approximation,* Proc. 1st Europ. Congr. Math., Hrsg. Joseph, A., Mignot, F., Murat, F., Prum, B., Rentschler, R., Springer, Berlin, 1994.

[Bab95] Babai, L., *Automorphism groups, isomorphism, reconstruction,* in: [GGL95], 1447-1540.

[Bäb38] Bäbler, F., *Über die Zerlegung regulärer Streckenkomplexe ungerader Ordnung,* Comment. Math. Helv. 10, 1938, 275-287.

[Bäb53] Bäbler, F., *Über eine spezielle Klasse Eulerscher Graphen,* Comment. Math. Helv. 27, 1953, 81-100.

[Bak94] Baker, B.S., *Approximation algorithms for NP-complete problems on planar graphs,* J. ACM 41, 1994, 153-180; preliminary version in: Proc. 24th IEEE FoCS, 1983, 265-273.

[Bar75] Baranyai, Z., *On the factorisation of the complete uniform hypergraph,* in: Infinite and finite sets, Hrsg. Hajnal, A., Rado, T., Sós, V.T., North-Holland, Amsterdam, 1975, 91-108.

[Bar83a] Barahona, F., *The max cut problem in graphs not contractible to K_5,* Oper. Res. Lett. 2, 1983, 107-111.

[Bar83b] Barnette, D., *Map coloring, polyhedra, and the four-color problem,* The Math. Assoc. of America, 1983.

[Bar90a] Barahona, F., *Planar multicommodity flows, max cut, and the Chinese postman problem,* in: Polyhedral combinatorics, Hrsg. Cook, W., Seymour, P.D., DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 1, 1990, 189-202.

[Bar90b] Barahona, F.., *On some applications of the Chinese postman problem,* in: Paths, flows, and VLSI-layout, Hrsg. Korte, B., Lovász, L., Prömel, H.J., Schrijver, A., Springer, Berlin, 1990, 1-16.

[Bar95] Barnette, D.W., *A short proof of the d-connectedness of d-polytopes,* Disc. Math. 137, 1995, 351-352.

[BB96] Brassard, G., Bratley, P., *Fundamentals of algorithms,* Prentice Hall, Englewood Cliffs NJ, 1996.

[BBRV89] Bauer, D., Broersma, H.J., Rao, L., Veldman, H.J., *A generalization of a result of Häggkvist and Nicoghossian,* J. Combin. Theory B 47, 1989, 237-243.

[BC76] Bondy, J.A., Chvátal, V., *A method in graph theory,* Disc. Math. 15, 1976, 111-136.

[BC84] Berge, C., Chvátal, V., (Hrsg.), *Topics on perfect graphs,* Ann. Disc. Math. 21, North-Holland, Amsterdam, 1984.

[BC94] Bovet, D.P., Crescenzi, P., *Introduction to the theory of complexity,* Prentice - Hall, New York, 1994.

[BC95] Bixby, R.E., Cunningham, W.H., *Matroid optimization and algorithms,* in: [GGL95], 551-609.

[BCE80] Bollobás, B., Catlin, P.A., Erdős, P., *Hadwiger's conjecture is true for almost every graph,* Europ. J. Combin. 1, 1980, 195-199.

[BCLS87] Bui, T.N., Chaudhuri, S., Leighton, F.T., Sipser, M., *Graph bisection algorithms with good average case behavior,* Combin. 7, 1987, 171-191.

[BCN89] Brouwer, A.E., Cohen, A.M., Neumaier, A., *Distance-regular graphs,* Springer, Heidelberg, 1989.

[BCOL92] Ben-David, S., Chor, B., Goldreich, O., Luby, M., *On the theory of average case complexity,* J. Comput. System Sci. 44, 1992, 193-219.

[BCP95] Ball, M.O., Colbourn, C.J., Provan, J.S., *Network reliability,* in: Network models, Handbooks in OR & MS, Vol. 7, Hrsg. Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L., Elsevier, Amsterdam, 1995, 673-762.

[BD76] Borodin, A., Demers, A., *Some comments on functional self-reducibility and the* NP *hierarchy,* Tech. Report TR76-284, Dept. Comput. Sci., Cornell Univ., Ithaca NY, 1976.

[BD84] Berge, C., Duchet, P., *Strongly perfect graphs,* in [BC84], 57-61.

[BDG88] Balcázar, J.L., Díaz, J., Gabarró, J., *Structural complexity I/II,* Springer-Verlag, Berlin, 1988/1990.

[BE73] Bollobás, B., Erdős, P., *On the structure of edge graphs,* Bull. LMS 5, 1973, 317-321.

[BE76] Bollobás, B., Erdős, P., *Cliques in random graphs,* Math. Proc. Camb. Phil. Soc. 80, 1976, 419-427.

[BE81] Bar-Yehuda, R., Even, S., *A linear-time approximation algorithm for the weighted vertex cover problem,* J. Alg. 2, 1981, 198-203.

[BE85] Bar-Yehuda, R., Even, S., *A local-ratio theorem for approximating the weighted vertex cover problem,* in: Analysis and design of algorithms for combinatorial problems, Hrsg. Ausiello, G., Lucertini, M., Ann. Disc. Math. 25, 1985, 27-45.

[BE95] Beigel, R., Eppstein, D., *3-coloring in time $\mathcal{O}(1.3446^n)$: a no-MIS algorithm,* Electron. Colloq. Comput. Compl. TR95-033, 1995; erhältlich über WWW an der URL „http://www.eccc.uni-trier.de/eccc/".

[Bei67a] Beineke, L.W., *Complete bipartite graphs: decomposition into planar subgraphs,* in: A seminar in graph theory, Hrsg. Harary, F., Holt, Rinehart and Winston, New York, 1967, 42-53.

[Bei67b] Beineke, L.W., *The decomposition of complete graphs into planar subgraphs,* in: Graph theory and theoretical physisc, Hrsg. Harary, F., Academic Press, London, 1967, 139-154.

[Bel57] Bellman, R., *Dynamic programming,* Princeton Univ. Press, Princeton NJ, 1957.

[Bel96] Bellare, M., *Proof checking and approximation: towards tight results,* überarbeitete Fassung der Complexity Theory Column 12, SIGACT News 27 (1), 1996; erhältlich über WWW unter der URL „http://www-cse.ucsd.edu/users/mihir".

[Ber57] Berge, C., *Two theorems in graph theory,* Proc. Nat. Acad. Sci. US 43, 1957, 842-844.

[Ber58] Berge, C., *Sur le couplage maximum d'un graphe,* C.R. Acad. Sci. Paris (A) 247, 1958, 258-259.

[Ber60] Berge, C., *Les problèmes de colorations en théorie des graphs,* Publ. Inst. Statist. Univ. Paris 9, 1960, 123-160.

[Ber61] Berge, C., *Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind* (Zusammenfassung), Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg, Math.-Natur. Reihe, 1961, 114.

[Ber76] Bermond, J.C., *On Hamiltonian walks,* in: Proc. 5th British Combin. Conf., Hrsg. Nash-Williams, C.St.J.A., Sheehan, J., Utilitas Math. Publ., Winnipeg, 1976, 41-51.

[Ber78] Bermond, J.C., *Hamiltonian graphs,* in [BW78], 127-167.

[Ber81] Bertossi, A.A., *The edge Hamiltonian path problem is* NP*-complete,* Inform. Process. Lett. 13, 1981, 157-159.

[Ber84] Bertossi, A.A., *Dominating sets for split and bipartite graphs,* Inform. Process. Lett. 19, 1984, 37-40.

[Ber85] Berge, C., *Graphs*, North-Holland, Amsterdam, 1985; Überarbeitung des ersten Teils von „Graphs and hypergraphs" von 1973.

[Ber89] Berge, C., *Hypergraphs: combinatorics of finite sets*, North-Holland, Amsterdam, 1973, 1989.

[BES76] Bollobás, B., Erdős, P., Simonovits, M., *On the structure of edge graphs II,* J. LMS 12, 1976, 219-224.

[BET88] Blair, J., England, R., Thomason, M., *Cliques and their separators in triangulated graphs,* Tech. Rep. CS-78-88, Dept. Comput. Sci., Univ. of Tennessee, Knoxville, 1988.

[BETT94] Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G., *Algorithms for automatic graph drawing: an annotated bibliography,* Comp. Geometry Theory Appl. 4, 1994, 235-282.

[BF91] Berge, C., Fournier, J.C., *A short proof for a generalization of Vizing's theorem,* J. Graph Theory 15, 1991, 333-336.

[BFM89] Buss, S.R., Fellows, M.R., Monien, B., *Linear-time algorithms for some well-known fixed parameter problems,* Manuskript, Dept. Comput. Sci., Univ. Idaho, 1989.

[BFM+81] Beeri, C., Fagin, R., Maier, D., Mendelzon, A., Ullman, J.D., Yannakakis, M. *Properties of acyclic database schemes,* Proc. 13th ACM SToC 1981, 355-362.

[BG77] Boesch, F.T., Gimpel, J.F., *Covering the points of a digraph with point-disjoint paths and its application to code optimization,* J. ACM 24, 1977, 192-198.

[BG81] Bernstein, P.A., Goodman, N., *Power of natural semijoins,* SIAM J. Comput. 10, 1981, 751-771.

[BG91] Balinski, M.L., Gonzalez, J., *Maximum matchings in bipartite graphs via strong spanning trees,* Networks 21, 1991, 165-179.

[BG94] Bellare, M., Goldwasser, S., *The complexity of decision versus search,* SIAM J. Comput. 23, 1994, 97-119.

[BG95] Brönnimann, H., Goodrich, M.T., *Almost optimal set covers in finite VC-dimension,* Disc. Comput. Geometry 14, 1995, 463-479.

[BG96] Becker, A., Geiger, D., *Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem,* Artif. Intell. 83, 1996, 167-188.

[BGHK95] Bodlaender, H.L., Gilbert, J.R., Hafsteinsson, H., Kloks, T., *Approximating treewidth, pathwidth, frontsize, and shortest elimination tree,* J. Alg. 18, 1995, 238-255.

[BGJR88] Barahona, F., Grötschel, M., Jünger, M., Reinelt, G., *An application of combinatorial optimization to statistical physics and circuit layout design,* Oper. Res. 36, 1988, 493-513.

[BGLR93] Bellare, M., Goldwasser, S., Lund, C., Russell, A., *Efficient probabilistically checkable proofs and applications to approximation,* erhältlich über WWW unter der URL „http://www-cse.ucsd.edu/users/mihir"; vorläufige Version in Proc. 25th ACM SToC 1993, 294-304.

[BGS95] Bellare, M., Goldreich, O., Sudan, M., *Free bits, PCPs and non-approximability – towards tight results,* Electron. Colloq. Comput. Compl. TR95-024, Revision 02, Dezember 1995; erhältlich über WWW an der URL „http://www.eccc.uni-trier.de/eccc/".

[BH65] Beineke, L.W., Harary, F., *The genus of the n-cube,* Canad. J. Math. 17, 1965, 494-496.

[BH90] Buckley, F., Harary, F., *Distance in graphs,* Addison-Wesley, Redwood City CA, 1990.

[BH92] Boppana, R., Halldórsson, M.M., *Approximating maximum independent sets by excluding subgraphs,* BIT 32, 1992, 180-196.

[BHKY62] Battle, J., Harary, F., Kodama, Y., Youngs, J.W.T., *Additivity of the genus of a graph,* Bull. AMS 68, 1962, 565-571.

[BHT79] Bland, R.G., Huang, H.-C., Trotter Jr., L.E., *Graphical properties related to minimal imperfection,* Disc. Math. 27, 1979, 11-22.

[Big74] Biggs, N.L., *Algebraic graph theory,* Cambridge University Press, 1974, 2. Auflage 1993.

[Big85] Biggs, N.L., *Discrete Mathematics,* Clarendon Press, Oxford, 1985.

[Big90] Biggs, N.L., *Some heuristics for graph coloring,* in [WN90], 87-96.

[Bir67] Birkhoff, G. *Lattice theory,* AMS, Providence, 3. Auflage 1967.

[BJ70] Barnette, D., Jukovič, E., *Hamiltonian circuits on 3-polytopes,* J. Combin. Theory 2, 1970, 54-59.

[BJ82] Booth, K.S., Johnson, J.H., *Dominating sets in chordal graphs,* SIAM J. Comput. 11, 1982, 191-199.

[BJ95] Bodlaender, H.L., Jansen, K., *Restrictions of graph partition problems - part I,* Theoret. Comput. Sci. 148, 1995, 93-109.

[BJL85] Beth, T., Jungnickel, D., Lenz, H., *Design theory,* Bibliographisches Institut, Zürich, 1985; Cambridge Univ. Press, 1993.

[BK64] Blažek, J., Koman, M., *A minimal problem concerning complete plane graphs,* in: Theory of graphs and its applications, Hrsg. Fiedler, M., Academic Press, New York, 1964, 113-117.

[BK77] Borodin, O.V., Kostochka, A.V., *On an upper bound of a graph's chromatic number, depending on the graph's degree and density,* J. Combin. Theory B 23, 1977, 247-250.

[BK79] Bernhart, F., Kainen, P.C., *The book thickness of a graph,* J. Combin. Theory B 27, 1979, 320-331.

[BL76] Booth, K.S., Lueker, G.S., *Testing the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms*, J. Comput. System Sci. 13, 1976, 335-379.

[BL95] Bienstock, D., Langston, M.A., *Algorithmic implications of the graph minor theorem,* in: Network models, Handbooks in OR & MS, Vol. 7, Hrsg. Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L., Elsevier, Amsterdam, 1995, 481-502.

[Blu90] Blum, N., *A new approach to maximum matching in general graphs,* in: Automata, languages and programming – ICALP '90, Hrsg. Paterson, M.S., Springer LNCS 443, 1990, 586-597.

[Blu94] Blum, A., *New approximation algorithms for graph coloring,* J. ACM 41, 1994, 470-516.

[BLW76] Biggs, N.L., Lloyd, E.K., Wilson, R.J., *Graph theory 1736-1936,* Oxford University Press, 1976.

[BLW87] Bern, M.W., Lawler, E.L., Wong, A.L., *Linear time computation of optimal subgraphs of decomposable graphs,* J. Alg. 8, 1987, 216-235.

[BM76] Bondy, J.A., Murty, U.S.R., *Graph theory with applications,* MacMillan Press, London, 1976; Amer. Elsevier, New York, 1979.

[BM87] Beisel, E.-P., Mendel, M., *Optimierungsmethoden des Operations Research – Band 1: Lineare und ganzzahlige lineare Optimierung,* Vieweg, Braunschweig / Wiesbaden, 1987.

[BM91] Beisel, E.-P., Mendel, M., *Optimierungsmethoden des Operations Research – Band 2: Optimierung in Graphen,* Vieweg, Braunschweig / Wiesbaden, 1991.

[BN92] Bienstock, D., Dean, N., *New results on rectilinear crossing numbers and plane embeddings,* J. Graph Theory 16, 1992, 389-98.

[BN93] Bienstock, D., Dean, N., *Bounds for rectilinear crossing numbers,* J. Graph Theory 17, 1993, 333-348.

[BO91] Brylawski, T.H., Oxley, J.G., *The Tutte polynomial and its applications*, in: Matroid Applications, Hrsg. White, N., Encycl. of Math. and its Appl., Cambridge Univ.Press, 1991, 123-225.

[Bod88] Bodlaender, H.L., *Dynamic programming on graphs with bounded treewidth*, Proc. ICALP '88, Hrsg. Lepistö, T., Salomaa, A., Springer LNCS 317, 1988, 105-118.

[Bod93a] Bodlaender, H.L., *A linear time algorithm for finding tree-decompositions of small width*, Proc. 25th ACM SToC 1993, 226-234.

[Bod93b] Bodlaender, H.L., *A tourist guide through treewidth,* Acta Cybernet. 11, 1993, 1-21.

[Bog94] Bogart, K.P., *Intervals and orders: what comes after interval orders?,* in: Orders, algorithms, and applications, Hrsg. Bouchitté, V., Morvan, M., Springer LNCS 831, Berlin, 1994.

[Bol78a] Bollobás, B., *Extremal graph theory*, Academic Press, London, 1978.

[Bol78b] Bollobás, B., *Chromatic number, girth and maximal degree*, Disc. Math. 24, 1978, 311-314.

[Bol78c] Bollobás, B., *Cycles and semi-topological configurations*, in: Theory and applications of graphs, Springer LNM 642, Berlin, 1978, 66-74.

[Bol79] Bollobás, B., *Graph theory*, Springer-Verlag, New York, 1979.

[Bol85] Bollobás, B., *Random graphs*, Academic Press, London, 1985.

[Bol86] Bollobás, B., *Combinatorics*, Cambridge University Press, 1986.

[Bol88] Bollobás, B., *The chromatic number of random graphs*, Combinatorica 8, 1988, 49-55.

[Bol91] Bollobás, B., *Random graphs* und *Random graphs revisited*, Proc. Symp. Appl. Math. Vol. 44, Hrsg. Bollobás, B., AMS, 1991, 1-20 und 81-97.

[Bol95] Bollobás, B., *Extremal graph theory*, in: [GGL95], 1231-1292.

[Bon78] Bondy, J.A., *A remark on two sufficient conditions for Hamilton cycles*, Disc. Math. 22, 1978, 191-194.

[Bon95] Bondy, J.A., *Basic graph theory: paths and circuits,* in: [GGL95], 3-110.

[Bor87] Borgwardt, K.H., *The simplex method. A probabilistic analysis*, Algorithms Combin. 1, Springer, 1987.

[Bör92] Börger, E., *Berechenbarkeit, Komplexität, Logik*, Vieweg, Braunschweig/ Wiesbaden, 1992[3].

[Bor95] Borie, R.B., *Generation of polynomial-time algorithms for some optimization problems on tree-decomposable graphs,* Algorithmica 14, 1995, 123-137.

[BP89] Bern, M., Plassmann, P., *The Steiner problem with edge lengths 1 and 2,* Inform. Process. Lett. 32, 1989, 171-176.

[BP92] Bui, T.N., Peck, A., *Partitioning planar graphs,* SIAM J. Comput. 21, 1992, 203-215.

[BP93] Blair, J.R.S., Peyton, B., *An introduction to chordal graphs and clique trees,* in: Graph theory and sparse matrix computation, Hrsg. George, A., Gilbert, J.R., Liu, J.W.H., Springer, New York, 1993, 1-29.

[BR91a] Bonchev, D., Rouvray, D.H., (Hrsg.), *Chemical graph theory - introduction and fundamentals,* Gordon and Breach, New York, 1991.

[BR91b] Brualdi, R.A., Ryser, H.J., *Combinatorial matrix theory,* Cambridge Univ. Press, 1991.

[BR94] Berman, P., Ramaiyer, V., *Improved approximations for the Steiner tree problem,* J. Alg. 17, 1994, 381-408.

[Bra85] Brassard, G., *Crusade for a better notation,* SIGACT News 17, 1985, 60-64.

[Bra94a] Brandstädt, A., *Graphen und Algorithmen,* Teubner, Stuttgart, 1994.

[Bra94b] Brandt, S., *Subtrees and subforests of graphs,* J. Combin. Theory B 61, 1994, 63-70.

[Bré79] Brélaz, D., *New methods to color the vertices of a graph,* Comm. ACM 22, 1979, 251-256

[Bro41] Brooks, R.L., *On coloring the nodes of a network,* Proc. Cambridge Phil. Soc. 37, 1941, 194-197.

[Bro72] Brown, J.R., *Chromatic scheduling and the chromatic number problem,* Management Sci. 19, 1972, 456-463.

[Bru58] De Bruijn, N.B., *Asymptotic methods in analysis,* North-Holland, Amsterdam, 1958.

[Bru75] Brualdi, R.A., *Transversal theory and graphs,* in: Studies in graph theory II, MAA Press, Washington D.C., 1975, 23-88.

[BRW85] Bender, E.A., Richmond, L.B., Wormald, N.C., *Almost all chordal graphs split,* J. Austral. Math. Soc. (A) 38, 1985, 214-221.

[Bry93] Bryant, V., *Aspects of combinatorics,* Cambridge Univ. Press, 1993.

[BS76] Bollobás, B., Sauer, N., *Uniquely colourable graphs with large girth,* Canad. J. Math. 28, 1976, 1340-1344.

[BS93] Brightwell, G.R., Scheinerman, E.R., *Representations of planar graphs*, SIAM J. Disc. Math. 6, 1993, 214-229.

[BT86] Bollobàs, B., Thomason,A. *Threshold functions,* Combin. 7, 1986, 35-38.

[Bun74] Buneman, P., *A characterization of rigid circuit graphs,* Disc. Math. 9, 1974, 205-212.

[BW73] Beineke, L.W., Wilson, R.J., *On the edge-chromatic number of a graph,* Disc. Math. 5, 1973, 15-20.

[BW78] Beineke, L.W., Wilson, R.J., (Hrsg.), *Selected topics in graph theory,* Academic Press, London, 1978.

[BW83] Beineke, L.W., Wilson, R.J., (Hrsg.), *Selected topics in graph theory 2,* Academic Press, London, 1983.

[BW85] Bender, E., Wilf, H.S., *A theoretical analysis of backtracking in the graph coloring problem,* J. Alg. 6, 1985, 275-282.

[BW88] Beineke, L.W., Wilson, R.J., (Hrsg.), *Selected topics in graph theory 3,* Academic Press, London, 1988.

[Cae88] De Caen, D., *On a theorem of König on bipartite graphs,* J. Combin. Inform. System Sci. 13, 1988, 127.

[Cam94] Cameron, P.J., *Combinatorics: topics, techniques, algorithms,* Cambridge Univ. Press, 1994.

[Cat78] Catlin, P.A., *A bound on the chromatic number of a graph,* Disc. Math. 22, 1978, 81-83.

[Cat79] Catlin, P.A., *Hajós' graph-coloring conjecture: variations and counterexamples,* J. Combin. Theory B 26, 1979, 268-274.

[CC93] Chao, C.Y., Chen, Z., *On uniquely 3-colorable graphs,* Disc. Math. 112, 1993, 21-27.

[CDS80] Cvetković, D., Doob, M., Sachs, H., *Spectra of graphs,* VEB Deutscher Verlag der Wissenschaften, Berlin; Academic Press, London, New York, 1980.

[CDGT88] Cvetković, D., Doob, M., Gutman, I., Torgasev, A., (Hrsg.), *Recent results in the theory of graph spectra,* Ann. Disc. Math. 36, North-Holland, Amsterdam, 1988.

[CE72] Chvátal, V., Erdős, P., *A note on hamiltonian circuits,* Disc. Math. 2, 1972, 111-113.

[CG69] Chartrand, G., Geller, D., *On uniquely colorable planar graphs,* J. Combin. Theory 6, 1969, 271-278.

[CG96] Czumaj, A., Gibbons, A., *Guthrie's problem: new equivalences and rapid reductions,* Theoret. Comput. Sci. 154, 1996, 3-22.

[CGH75] Cockayne, E., Goodman, S., Hedetniemi, S., *A linear algorithm for the domination number of a tree,* Inform. Process. Lett. 4, 1975, 41-44.

[CGM79] Chachra, V., Ghare, P.M., Moore, J.M., *Applications of graph theory algorithms,* North-Holland, New York, 1979.

[CGP85] De Caen, D., Gregory, D.A., Pullman, N.J., *Clique coverings of complements of paths and cycles,* in [AG85], 257-268.

[CGR94] Cherkassky, B.V., Goldberg, A.V., Radzik, T., *Shortest paths algorithms: theory and experimental evaluation,* in: Proc. 5th ACM-SIAM Symp. Disc. Alg, New York / Philadelphia, 1994, 516-525.

[CH67] Chartrand, G., Harary, F., *Planar permutation graphs,* Ann. Inst. Henri Poincaré Sec. B 3, 1967, 433-438.

[CH68] Cartwright, D., Harary, F., *On colorings of signed graphs,* Elem. Math. 23, 1968, 85-89.

[CH82] Cole, R., Hopcroft, J., *On edge coloring bipartite graphs,* SIAM J. Comput. 11, 1982, 540-546.

[CH94] Clark, J., Holton, D.A., *Graphentheorie: Grundlagen und Anwendungen,* Spektrum Akadem. Verlag, Heidelberg, 1994.

[Cha66] Chartrand, G., *A graph theoretic approach to a communication problem,* SIAM J. Appl. Math. 14, 1966, 778-781.

[Cha82] Chaitin, G.J., *Register allocation and spilling via graph coloring,* Proc. SIGPLAN '82 Symp. Compiler Constr., ACM, New York, 98-105.

[Cha92] Chaudhuri, P., *Parallel algorithms – design and analysis,* Prentice-Hall, New York, 1992.

[Che91] Chetwynd, A.G., *Total colourings of graphs - a progress report,* in: Graph theory, combinatorics, and applications, Vol.1, Hrsg. Alavi, Y., Chartrand, G., Oellermann, O.R., Schwenk, A.J., Wiley & Sons, New York, 1991, 287-296.

[CHL88] Cockayne, E.J., Hedetniemi, S.T., Laskar, R., *Gallai theorems for graphs, hypergraphs, and set systems,* Disc. Math. 72, 1988, 35-47.

[Cho59] Chomsky, N., *On certain formal properties of grammars,* Inform. Control 2, 1959, 137-167.

[Chr71] Christofides, N., *An algorithm for the chromatic number of a graph,* Comput. J. 14, 1971, 38-39.

[Chr76] Christofides, N., *Worst-case analysis of a new heuristic for the traveling salesman problem,* Tech. Report, GSIA, Carnegie-Mellon Univ., 1976.

[Chu36] Church, A., *An unsolvable problem of elementary number theory,* Amer. J. Math. 58, 1936, 345-363.

[Chu88] Chung, F.R.K., *The average distance and the independence number,* J. Graph Theory 12, 1988, 229-235.

[Chu90] Chung, F.R.K., *Separator theorems and their applications,* in: Paths, flows, and VLSI-layout, Hrsg. Korte, B., Lovász, L., Prömel, H.J., Schrijver, A., Springer, Berlin, 1990, 17-34.

[Chu91] Chung, F.R.K., *Improved separators for planar graphs*, Graph Theory, Combin. Appl. 1, 1991, 265-270.

[Chv73] Chvátal, V., *The minimality of the Mycielski graph,* in: Graphs and combinatorics, Springer LNM 406, Berlin, 1973, 243-246.

[Chv75] Chvátal, V., *A combinatorial theorem in plane geometry,* J. Combin. Theory B 18, 1975, 39-41.

[Chv79] Chvátal, V., *A greedy heuristic for the set covering problem,* Math. Oper. Res. 4, 1979, 233-235.

[Chv83] Chvátal, V. *Linear Programming,* Freeman, New York, 1983.

[Chv85] Chvátal, V., *Hamiltonian cycles,* in [LLRS85], 403-429.

[CJ96] Chen, C.C., Jin, G.P., *Cycle partitions in graphs,* Combin. Probab. Comput. 5, 1996, 95-97.

[CK95] Crescenzi, P., Kann, V., *A compendium of NP optimization problems,* Oktober 1995; erhältlich über WWW unter der URL „http://www.nada.kth.se/∼viggo/problemlist/compendium.html“.

[CKK93] Chen, J., Kanchi, S.P., Kanevsky, A., *On the complexity of graph embeddings,* Proc. 3rd WADS, Hrsg. Dehne, F., u.a., Springer LNCS 709, 1993, 234-245.

[CL86] Chartrand, G., Lesniak, L., *Graphs and digraphs,* Wadsworth und Brooks/Cole, Pacific Grove CA, 2. Auflage, 1986; Überarbeitung von: Behzad, M., Chartrand, G., Lesniak-Foster, L.: *Graphs & digraphs*, Prindle, Weber & Schmidt, Boston, 1979.

[CL91a] Cameron, P.J., Van Lint, J.H., *Designs, graphs, codes and their links,* Cambridge Univ. Press, 1991.

[CL91b] Coffman, E.G., Lueker, G.S., *Probabilistic analysis of packing and partitioning algorithms,* Wiley & Sons, New York, 1991.

[Cla83] Clarkson, K.L., *A modification of the greedy algorithm for vertex cover,* Inform. Process. Lett. 16, 1983, 23-25.

[CLR87] Chung, F.R.K., Leighton, F.T., Rosenberg, A.L., *Embedding graphs in books: a layout problem with applications to VLSI design,* SIAM J. Alg. Disc. Meth. 8, 1987, 33-58.

[CLR90] Cormen, T.H., Leiserson, C.E., Rivest, R.L., *Introduction to algorithms*, McGraw-Hill, New York, 2. Auflage 1990.

[CLW94] Chen, B.-L., Lih, K.-W., Wu, P.-L., *Equitable coloring and the maximum degree,* Europ. J. Combin. 15, 1994, 443-447.

[CM78] Capobianco, M., Molluzzo, J.C., *Examples and counterexamples in graph theory,* North-Holland, New York, 1978.

[CM93] Courcelle, B., Mosbah, M., *Monadic second-order evaluations on tree-decomposable graphs,* Theoret. Comput. Sci. 109, 1993, 49-82.

[CMWZZ94] Chen, B., Matsumoto, M., Wang, J., Zhang, Z., Zhang, J., *A short proof of Nash-William's theorem for the arboricity of a graph,* Graphs Combin. 10, 1994, 27-28.

[CN82] Chang, G.J., Nemhauser, G.L., *The k-domination and k-stability problem of graphs,* Report No. 540, School of OR & IE, Cornell University, Ithaca NY, 1982.

[CN90] Chrobak, M., Nishizeki, T., *Improved edge-coloring algorithms for planar graphs,* J. Alg. 11, 1990, 102-116.

[CNS82] Chiba, N., Nishizeki, T., Saito, N., *An approximation algorithm for the maximum independent set problem on planar graphs*, SIAM J. Comput. 11, 1982, 663-675.

[Cob64] Cobham, A., *The intrinsic computational difficulty of functions*, in: Proc. 1964 Int. Congr. Log. Method. Phil. Sci., Hrsg. Bar-Hillel, Y., North-Holland, Amsterdam, 1964, 24-30.

[Col87] Colbourn, C.J., *The combinatorics of network reliability,* Oxford Univ. Press, New York, 1987.

[Coo71] Cook, S.A., *The complexity of theorem-proving procedures*, Proc. 3rd ACM SToC, 1971, 151-158.

[COS91] Chartrand, G., Oellermann, O.R., Schultz, M., *Distance: a graphical tour,* in: Graph theory, combinatorics, algorithms, and applications, Hrsg. Alavi, Y., Chung, F.R.K., Graham, R.L., Hsu, D.F., SIAM, Philadelphia, 1991, 441-458.

[CPR75] Choudum, S.A. Parthasarathy, K. R., Ravindra, G., *Line-clique cover number of a graph,* Proc. Indian Nat. Sci. Acad. 41A, 1975, 289-293.

[CPS73] Chartrand, G., Polimeni, A.D., Stewart, M.J., *The existence of 1-factors in line graphs, squares, and total graphs,* Indag. Math. 35, 1973, 228-232.

[CS81] Chvátal, V., Szemerédi, E., *On the Erdős-Stone theorem,* J. LMS 2, 1981, 207-214.

[CS83] Chvátal, V., Szemerédi, E., *Notes on the Erdős-Stone theorem,* Ann. Disc. Math. 17, 1983, 183-190.

[CS85] Colbourn, C.J., Stewart, L.K., *Dominating cycles in series-parallel graphs,* Ars Combin. 19A, 1985, 107-112.

[CS88] Chvátal, V., Sbihi, N., *Recognizing claw-free perfect graphs,* J. Comb. Theory B 44, 1988, 154-176.

[CS94] Chen, G., Schelp, R.H., *Hamiltonian graphs with neighborhood intersections,* J. Graph Theory 18, 1994, 497-513.

[CT91] Chen, Z., Toda, S., *On the complexity of computing optimal solutions,* Internat. J. Found. Comput. Sci. 2, 1991, 207-220.

[CY94] Chung, F.R.K., Yau, S.-T., *A near optimal algorithm for edge separators,* Proc. 26th ACM STOC, 1994, 1-8.

[DA93] Dolan, A., Aldous, J., *Networks and algorithms - an introductory approach,* Wiley & Sons, Chichester, 1993.

[Dan49] Danzig, G.B., *Linear programming and extensions*, Princeton Univ. Press, Princeton, NJ, 1963, erste Veröffentlichung in: Econometrics 17, 1949, 200-211.

[Dav73] Davis, M., *Hilbert's tenth problem is unsolvable*, Amer. Math. Monthly 80, 1973, 233-269.

[Dav82] Davis, M., *Why Gödel didn't have Church's thesis*, Inform. Control 54, 1982, 3-24.

[DB81] Dutton, R.D., Brigham, R.C., *A new graph coloring algorithm,* Comput. J. 24, 1981, 85-86.

[DD95] Domschke, W., Drexl, A., *Einführung in Operations Research,* Springer-Lehrbuch, Berlin, 2. Auflage, 1995.

[DDSV93] Diks, K., Djidjev, H.N., Sykora, O., Vrto, I., *Edge separators of planar and outerplanar graphs with applications*, J. Alg. 14, 1993, 258-279.

[Der88] Derigs, U., *Programming in networks and graphs,* Lecture Notes in Economics and Mathematical Systems 300, Springer Berlin, 1988.

[Des47] Descartes, B., (= Tutte, W.T.), *A three color problem*, Eureka 9, 1947.

[Des48] Descartes, B., (= Tutte, W.T.), *Solutions to problems in Eureka No. 9*, Eureka 10, 1948.

[Des54] Descartes, B., (= Tutte, W.T.), *Solution to advanced problem No. 4526*, Amer. Math. Monthly 61, 1954, 352.

[Dew83] Dewdney, A.K., *Fast Turing reductions between problems in NP,* Chapter 4: Reductions between NP-complete problems, Report Nr. 71, Dept. Comput. Sci. Univ. West. Ontario, London ON, 1983.

[DF85] Dyer, M.E., Frieze, A.M., *A simple heuristic for the p-centre problem,* Oper. Res. Lett. 3, 1985, 285-288.

[DF89] Dyer, M.E., Frieze, A.M., *The solution of some random NP-hard problems in polynomial expected time,* J. Alg. 10, 1989, 451-489.

[DF95a] Downey, R.G., Fellows, M.R., *Fixed-parameter tractability and completeness I: basic results,* SIAM J. Comput. 24, 1995, 873-921.

[DF95b] Downey, R.G., Fellows, M.R., *Fixed-parameter tractability and completeness II: on completeness for $W[1]$,* Theoret. Comput. Sci. 141, 1995, 109-131.

[DFJ94] Dyer, M., Frieze, A., Jerrum, M., *Approximately counting Hamilton cycles in dense graphs,* Proc. 5th ACM-SIAM SoDA, 1994, 336-343.

[DGP94] Demange, M., Grisoni, P., Paschos, V.T., *Approximation results for the minimum graph coloring problem,* Inform. Process. Lett. 50, 1994, 19-23.

[DH91] Dean, A.M., Hutchinson, J.P., *Relations among embedding parameters for graphs,* in: Graph theory, combinatorics, and applications, Vol.1, Hrsg. Alavi, Y., Chartrand, G., Oellermann, O.R., Schwenk, A.J., Wiley & Sons, New York, 1991, 287-296.

[DH95] Du, D.-Z., Hsu, D.F., *Combinatorial network theory,* Kluwer Acad. Publ., Dordrecht, 1995.

[Die96] Diestel, R., *Graphentheorie,* Springer, Heidelberg, erscheint voraussichtlich September 1996.

[Dij59] Dijkstra, E.W., *A note on two problems in connexion with graphs,* Numer. Math. 1, 1959, 269-271.

[Dil50] Dilworth, R.P., *A decomposition theorem for partially ordered sets,* Ann. Math. 51, 1950, 161-166.

[Din70] Dinic, E.A., *Algorithm for solution of a problem of maximal flow in a network with power estimation,* Sov. Math. Dokl. 11, 1970, 1277-1280.

[Din79] Dinitz, J., 10th south-eastern conference on combinatorics, graph theory, and computing; Boca Raton, 1979; zitiert in [ERT79].

[Dir52a] Dirac, G.A., *Some theorems on abstract graphs,* Proc. LMS 2, 1952, 69-81.

[Dir52b] Dirac, G.A., *A property of 4-chromatic graphs and some remarks on critical graphs,* J. LMS 27, 1952, 85-92.

[Dir53] Dirac, G.A., *The structure of k-chromatic graphs,* Fund. Math. 40, 1953, 42-55.

[Dir60] Dirac, G.A., *In abstrakten Graphen vorhandene vollständige 4-Graphen und ihre Unterteilungen,* Math. Nachr. 22, 1960, 61-85.

[Dir61] Dirac, G.A., *On rigid circuit graphs,* Abh. Math. Sem. Univ. Hamburg 25, 1961, 71-76.

[Dji82] Djidjev, H.N., *On the problem of partitioning planar graphs,* SIAM J. Alg. Disc. Meth. 3, 1982, 229-240.

[Dji85a] Djidjev, H.N., *A separator theorem for graphs of fixed genus,* Serdica (Bulg. math. pub.) 11, 1985, 319-329.

[Dji85b] Djidjev, H.N., *A linear algorithm for partitioning graphs of fixed genus,* Serdica (Bulg. math. pub.) 11, 1985, 369-387.

[Dji88] Djidjev, H.N., *Linear algorithms for graph separation problems*, Proc. 1st Scand. Workshop on Algorithm Theory, Halmstad, Sweden, Hrsg. Karlsson, R., Lingas, A., Springer LNCS 318, 1988, 216-222.

[DJPSY94] Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M., *The complexity of multiterminal cuts,* SIAM J. Comput. 23, 1994, 864-894.

[DKL87] Deo, N., Krishamoorty, M.S., Langston, M.A., *Exact and approximate solutions for the gate matrix layout problem,* IEEE Trans. Comput. Aid. Design 1987, 79-84.

[DL77] Dreyfus, S.E., Law, A.M., *The art and theory of dynamic programming,* Academic Press, New York, 1977.

[DLMS96] Diekmann, R., Lüling, R., Monien, B., Spräner, C., *Combining helpful sets and parallel simulated annealing for the graph-partitioning problem,* J. Parallel Alg. Appl. 8, 1996, 61-84.

[DM91] Derigs, U., Metz, A., *Solving (large scale) matching problems combinatorially,* Math. Progr. 50, 1991, 113-121.

[DMP64] Demoucron, G., Malgrange, Y., Pertuiset, R., *Graphes planaires, reconnaissance et construction des representations planaires topologiques,* Rev. Franc. Recher. Opérat. 8, 1964, 34-47.

[DP60] Davis, M., Putnam, H., *A computing procedure for quantification theory,* J. ACM 7, 1960, 201-215.

[DP93a] Delorme, C., Poljak, S., *Laplacian eigenvalues and the maximum cut problem,* Math. Progr. 62, 1993, 557-574.

[DP93b] Delorme, C., Poljak, S., *Combinatorial properties and the complexity of a max-cut approximation,* Europ. J. Combin. 14, 1993, 313-333.

[DS90] Dueck, G., Scheuer, T., *Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing,* J. Comput. Phys. 90, 1990, 161-175.

[Duch95] Duchet, P., *Hypergraphs,* in: [GGL95], 381-432.

[Duf65] Duffin, R.J., *Topology of series-parallel networks*, J. Math. Anal. Appl. 10, 1965, 303-318.

[DW83] Davis, M.D., Weyuker, E.J., *Computability, complexity, and languages,* Academic Press, San Diego, 1983.

[Edm65] Edmonds, J., *Paths, trees, and flowers*, Canad. J. Math. 17, 1965, 449-467.

[Edw86] Edwards, K., *The complexity of colouring problems on dense graphs,* Theoret. Comput. Sci. 43, 1986, 337-343.

[EF81] Erdős, P., Fajtlowicz, S., *On the conjecture of Hajós*, Combin. 1, 1981, 141-143.

[EFR86] Erdős, P., Frankl, P., Rödl, V., *The asymptotic number of graphs not containing a fixed subgraph and a problem for hypergraphs having no exponent,* Graphs Combin. 2, 1986, 113-121.

[EFS56] Elias, P., Feinstein, A., Shannon, C.E., *A note on the maximum flow through a network,* IRE Trans. Inform. Theory IT 2, 1956, 117-9.

[EG59] Erdős, P., Gallai, T., *On maximal paths and circuits,* Acta. Math. Sci. Hung. 10, 1959, 337-356.

[EG73] Erdős, P., Guy, R.K., *Crossing number problems,* Amer. Math. Monthly 80, 1973, 52-58.

[Ege31] Egerváry, E., *Matrixok kombinatorius tulajdonságairól,* Mat. Fiz. Lapok 38, 1931, 16-28.

[EGL91] Egawa, Y., Glas, R., Locke, S.C., *Cycles and paths through specified vertices in k-connected graphs,* J. Combin. Theory B 52, 1991, 20-29.

[EGL95] Eiselt, H.A., Gendreau, M., Laporte, G., *Arc routing problems, Part I: The Chinese postman problem, Part II: The rural postman problem,* Oper. Res. 43, 1995, 231-242 bzw. 399-414.

[EGP66] Erdős, P., Goodman, A.W., Pósa, L., *The representation of a graph by set intersections,* Canad. J. Math. 18, 1966, 106-112.

[EGS75] Erdős, P., Graham, R.L, Szemerédi, E., *On sparse graphs with dense long paths*, Comput. Math. Appl. 1, 1975, 365-369.

[EH85] Erdős, P., Hajnal, A., *Chromatic number of finite and infinite graphs and hypergraphs,* Disc. Math. 53, 281-285.

[EIS76] Even, S., Itai, A., Shamir, A., *On the complexity of timetable and multicommodity flow problems,* SIAM J. Comput. 5, 1976, 691-703.

[EJ73] Edmonds, J., Johnson, E.L., *Matching, Euler tours and the Chinese postman,* Math. Progr. 5, 1973, 88-124.

[EK72] Edmonds, J., Karp, R.M., *Theoretical improvements in algorithm efficiency for network flow problems,* J. ACM 19, 1972, 248-264.

[EKR76] Erdős, P., Kleitman, D.J., Rothschild, B.L., *Asymptotic enumeration of K_n-free graphs,* in: Internat. Colloq. Combin. Theory, Atti dei Convegni Lincei 17, Rom, 1976, 19-27.

[ELP72] Even, S., Lempel, A., Pnueli, A., *Permutation graphs and transitive graphs,* J. ACM 19, 1972, 400-410.

[ELR85] O'h Eigeartaigh, M., Lenstra, J.K., Rinnoy Kan, A.H.G., (Hrsg.), *Combinatorial optimization / Annotated bibiliographies,* Wiley & Sons, Chichester / Centre for Mathematics and Computer Science, Amsterdam, 1985.

[ER60] Erdős, P., Rényi, A. *On the evolution of random graphs,* Publ. Math. Inst. Hungar. Acad. Sci. 5, 1960, 17-61.

[Erd47] Erdős, P., *Some remarks on the theory of graphs,* Bull. AMS 53, 1947, 292-294.

[Erd59] Erdős, P., *Graph theory and probability,* Canad. J. Math. 11, 1959, 34-38.

[Erd61] Erdős, P., *Graph theory and probability II,* Canad. J. Math. 13, 1961, 346-352.

[Erd65] Erdős, P., *On some extremal problems in graph theory,* Israel J. Math. 3, 1965, 113-116.

[Erd67] Erdős, P., *On bipartite subgraphs of graphs,* Math. Lapok 18, 1967, 283-288.

[ERT79] Erdős, P., Rubin, A.L., Taylor, H., *Choosability in graphs,* Proc. west coast conf. combinatorics, graph theory and computing; Congr. Numer. XXVI, 1979, 125-157.

[ES35] Erdős, P., Szekeres, G., *A combinatorial problem in geometry,* Compositio Math. 2, 1935, 463-470.

[ES46] Erdős, P., Stone, A.H., *On the structure of linear graphs,* Bull. AMS 52, 1946, 1089-1091.

[ES63] Erdős, P., Sachs, H. *Reguläre Graphen gegebener Taillenweite mit minimaler Knotenzahl,* Wiss. Z. Univ. Halle-Wittenberg, Math.-Nat. R. 12, 1963, 251-258.

[ES66] Erdős, P., Simonovits, M. *A limit theorem in graph theory,* Studia Sci. Math. Hung. 1, 1966, 51-57.

[ET75] Even, S., Tarjan, R.E., *Network flow and testing graph connectivity,* SIAM J. Comput. 4, 1975, 507-518.

[ET76] Even, S., Tarjan, R.E., *Computing an s,t-numbering,* Theoret. Comput. Sci. 2, 1976, 339-344.

[Euler] Euler, L., *Solutio problematis ad geometriam situs pertinentis,* (in Latein), Comm. Acad. Sci. Imp. Petropolitanae 8, 1736, 128-140.

[EW77] Erdős, P., Wilson, R.J., *On the chromatic index of almost all graphs,* J. Combin. Theory B 23, 1977, 255-257.

[Fag74] Fagin, R., *Generalized first order spectra and polynomial-time recognizable sets,* in: Complexity of computations, Hrsg. Karp, R., AMS, 1974.

[Fan84] Fan, G.H., *New sufficient conditions for cycles in graphs,* J. Combin. Theory B 37, 1984, 221-227.

[Far48] Fáry, I., *On straight line representation of planar graphs,* Acta Sci. Math. Szeged 11, 1948, 229-233.

[Fei96] Feige, U., *A threshold of* $\ln n$ *for approximating set cover,* Proc. 28th ACM SToC 1996.

[FF56] Ford Jr., L.R., Fulkerson, D.R., *Maximum flow through a network,* Canad. J. Math. 8, 1956, 399-404.

[FF62] Ford Jr., L.R., Fulkerson, D.R., *Flows in networks,* Princeton Univ. Press, 1962.

[FF85] Fournier, I., Fraisse, P., *On a conjecture of Bondy,* J. Combin. Theory B 39, 1985, 17-26.

[FF94] Fritsch, R., Fritsch, G., *Der Vierfarbensatz,* BI Wissenschaftsverlag, Mannheim, 1994.

[FG65] Fulkerson, D.R., Gross, O.A., *Incidence matrices and interval graphs,* Pacific J. Math. 15, 1965, 835-855.

[FG85] Frieze, A.M., Grimmett, G.R., *The shortest path problem for graphs with random arclengths,* Disc. Appl. Math. 10, 1985, 57-77.

[FIN93] Frank, A., Ibaraki, T., Nagamochi, H., *On sparse subgraphs preserving connectivity properties,* J. Graph Theory 17, 1993, 275-281.

[Fio75] Fiorini, S., *On the chromatic index of outerplanar graphs,* J. Combin. Theory B 18, 1975, 35-38.

[Fis78] Fisk, S., *A short proof of Chvátal's watchman theorem,* J. Combin. Theory B 24, 1978, 374.

[Fis85] Fishburn, P.C., *Interval orders and interval graphs,* Wiley, New York, 1985.

[FJ95] Frieze, A., Jerrum, M., *Improved approximation algorithms for MAX k-CUT and MAX BISECTION,* in: Integer programming and combinatorial optimization, Hrsg. Balas, E., Clausen, J., Springer LNCS 920, 1995, 1-13; erscheint in Algorithmica.

[FJMR88] Frieze, A.M., Jackson, B., McDiarmid, C.J.H., Reed, B., *Edge-colouring of random graphs,* J. Combin. Theory B 45, 1988, 135-149.

[FK95] Feige, U., Kilian, J., *On limited versus polynomial nondeterminism,* Weizmann Institute of Science, Dept. Math., Technical Report CS95-34, Dezember 1995; erhältlich über WWW als „http://www.wisdom.weizmann.ac.il/Papers/CSreports/reps95/95-34.ps.Z".

[FK96] Feige, U., Kilian, J., *Zero knowledge and the chromatic number,* Preliminary version, Complexity Conference 1996.

[FL94] Fellows, M.R., Langston, M.A., *On search, decision and the efficiency of polynomial-time algorithms,* J. Comput. System Sci. 49, 1994, 769-779.

[Fle83] Fleischner, H., *Eulerian graphs*, in [BW83], 17-53.

[Fle90] Fleischner, H., *Eulerian graphs and related topics*, Part 1, Vol. 1 & 2, Ann. Disc. Math. 45 bzw. 50, North-Holland, Amsterdam, 1990 bzw. 1991.

[Fly94] Flye Sainte-Marie, C., *Solution to question nr. 48,* Interméd. Math. 1, 1894, 107-110.

[FM95] Feder, T., Motwani, R., *Clique partitions, graph compression and speeding-up algorithms,* J. Comput. System Sci. 51, 1995, 261-272.

[FMR79] Filotti, I.S., Miller, G.L., Reif J., *On determining the genus of a graph in $\mathcal{O}(V^{\mathcal{O}(g)})$ steps*, Proc. 11th ACM SToC, 1979, 27-37.

[FMR92] Frieze, A., McDiarmid, C., Reed, B., *On a conjecture of Bondy and Fan,* Ars Combin. 33, 1992, 329-336.

[For84] Forster, O., *Analysis 2,* Vieweg, Braunschweig / Wiesbaden, 5. Auflage, 1984.

[Fou73] Fournier, J.C., *Coloration des arêtes d'un graphe,* Cahiers Centre Études Rech. Opér. 15, 1973, 311-314.

[Fou92] Foulds, L.R., *Graph theory applications,* Springer, New York, 1992.

[FR74] Fischer, M.J., Rabin, M.O., *Super-exponential complexity of Presburger arithmetic*, in: Complexity of computation, Hrsg. Karp, R.M., SIAM-AMS Proc. 7, AMS, Providence RI, 1974, 27-41.

[FR94] Fürer, M., Raghavachari, B., *Approximating the minimum-degree Steiner tree to within one of optimal,* J. Alg. 17, 1994, 409-423.

[FR95] Frieze, A., Reed, B., *Covering the edges of a random graph by cliques,* Combin. 15, 1995, 489-497.

[Fra86] Fraisse, P., *A new sufficient condition for Hamiltonian graphs,* J. Graph Theory 10, 1986, 405-409.

[Fra90] Frank, A., *Packing paths, circuits, and cuts – a survey,* in: Paths, flows and VLSI-layout, Hrsg. Korte, B., Lovász, L., Prömel, H.J., Schrijver, A., Springer, Berlin, 1990, 47-100.

[Fra92] Frank, A., *Augmenting graphs to meet edge-connectivity requirements,* SIAM J. Disc. Math. 5, 1992, 25-53.

[Fra94a] Frank, A., *On the edge-connectivity algorithm of Nagamochi and Ibaraki,* Laboratoire Artemis, IMAG, Université J. Fourier, Grenoble, 1994.

[Fra94b] FRANK, A., *Connectivity augmentation problems in network design,* in: Mathematical programming – state of the art 1994, Hrsg. Birge, J.R., Murty, K.G., Univ. of Michigan, 1994, 34-63.

[Fra95] FRANK, A., *Connectivity and network flows,* in: [GGL95], 111-177.

[Fre87] FREDERICKSON, F.N., *Fast algorithms for shortest paths in planar graphs, with applications,* SIAM J. Comput. 16, 1987, 1004-1022.

[Fri89] FRIEZE, A.M., *On matchings and Hamilton cycles in random graphs*, in: Surveys in combinatorics 1989, Hrsg. Siemons, J., LMS Lecture Notes Series Vol. 141, Cambridge Univ. Press, 1989, 84-114.

[Fri90] FRIEZE, A.M., *Probabilistic analysis of graph algorithms*, in: Computational graph theory, Hrsg. Tinhofer, G., u.a., Computing Supplementum 7, Springer-Verlag, Wien, 1990, 209-233.

[Fro17] FROBENIUS, G., *Über zerlegbare Determinanten*, Sitzungsber. König. Preuss. Akad. Wiss. Berlin XVIII, 1917, 274-277.

[FRS95] FRIEZE, A., RADCLIFFE, A.J., SUEN, S., *Analysis of a simple greedy matching algorithm on random cubic graphs,* Combin. Probab. Comput. 4, 1995, 47-66.

[FS69] FINCK, H.J., SACHS, H., *Über eine von H.S. Wilf angegebene Schranke für die chromatische Zahl endlicher Graphen,* Math. Nachr. 39, 1969, 373-386.

[FT87] FREDMAN, M.L., TARJAN, R.E., *Fibonacci heaps and their uses in improved network optimization problems,* J. ACM 34, 1987, 596-615.

[Ful58] FULKERSON, D.R., *Note on Dilworth's decomposition theorem for partially ordered sets,* Proc. AMS 7, 1958, 78-85.

[Ful71] FULKERSON, D.R., *Blocking and anti-blocking pairs of polyhedra,* Math. Progr. 1, 1971, 168-194.

[Für88] FÜREDI, Z., *Matching and covers in hypergraphs,* Graphs Combin. 4, 1988, 115-206.

[Für91] FÜREDI, Z., *Turán type problems,* in: Surveys in combinatorics 1991, Hrsg. Keedwell, A.D., LMS Lecture Notes Series Vol. 166, Cambridge Univ. Press, 1991, 253-300.

[FW77] FIORINI, S., WILSON, R.J., *Edge-colourings of graphs,* Pitman, London, 1977.

[FW78] FIORINI, S., WILSON, R.J., *Edge colorings of graphs,* in: [BW78], 103-126.

[FW81] FRANKL, P., WILSON, R.M., *Intersection theorems with geometric consequences*, Combinatorica 1, 1981, 357-368.

[Gab76] GABOW, H.N., *Using Euler partitions to edge color bipartite mulitgraphs,* Internat. J. Comput. Inform. Sci. 5, 1976, 345-355.

[Gab90] GABOW, H.N., *Data structures for weighted matching and nearest common ancestors with linking,* Proc. 1st ACM-SIAM SoDA, 1990, 434-443.

[Gal59] GALLAI, T., *Über extreme Punkt- und Kantenmengen,* Ann. Univ. Sci. Budapest Eötvös Sect. Math. 2, 1959, 133-138.

[Gal64] GALLAI, T., *Elementare Relationen bezüglich der Glieder und trennenden Punkte von Graphen,* Magyar Tud. Akad. Mat. Kutató Int. Közl. 9, 1964, 235-236.

[Gal86] GALIL, Z., *Efficient algorithms for finding maximum matching in graphs,* ACM Computing Surveys 18, 1986, 23-38.

[Gal95] GALVIN, F., *The list chromatic index of a bipartite multigraph,* J. Combin. Theory B 63, 1995, 153-158.

[Gav65] GAVETT, J., *Three heuristic rules for sequencing jobs to a single production facility,* Management Sci. 11, 1965, B166-B176.

[Gav72] GAVRIL, F., *Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph,* SIAM J. Comput. 1, 1972, 180-187.

[Gav74a] GAVRIL, F., *The intersection graphs of subtrees in trees are exactly the chordal graphs,* J. Combin. Theory B 16, 1974, 47-56.

[Gav74b] Gavril, F., Zitiert in [GJ79] als persönliche Mitteilung.

[Gav77] Gavril, F., *Algorithms on clique separable graphs,* Disc. Math. 19, 1977, 159-165.

[GB93] Goemans, M.X., Bertsimas, D.J., *Survivable networks, linear programming relaxations and the parsimonious property,* Math. Progr. 60, 1993, 145-166.

[Ger95] Gerards, A.M.H., *Matching,* in: Network models, Handbooks in OR & MS, Vol. 7, Hrsg. Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L., Elsevier, Amsterdam, 1995, 135-224.

[GGL95] Graham, R.L., Grötschel, M., Lovász, L., (Hrsg.), *Handbook of combinatorics,* Vol. I (Seiten 1-1018) und II (Seiten 1019-2198), North-Holland, Amsterdam, 1995.

[GGP+94] Goemans, M.X., Goldberg, A.V., Plotkin, S., Shmoys, D.B., Tardos, E., Williamson, D.P., *Improved approximation algorithms for network design problems,* Proc. 5th ACM-SIAM SoDA, 1994, 223-232.

[GGS89] Gabow, H.N., Galil, Z., Spencer, T.H., *Efficient implementation of graph algorithms using contraction,* J. ACM 36, 1989, 540-572.

[GGST86] Gabow, H.N., Galil, Z., Spencer, T.H., Tarjan, R.E., *Efficient algorithms for finding minimum spanning trees in undirected and directed graphs,* Combin. 6, 1986, 109-122.

[GH64] Gilmore, P.C., Hoffman, A.J., *A characterization of comparability graphs and of interval graphs,* Canad. J. Math. 16, 1964, 539-548.

[GH85] Graham, R.L., Hell, P., *On the history of the minimum spanning tree problem,* Ann. History Comput. 7, 1985, 43-57.

[GH90] Goldschmidt, O., Hochbaum, D.S., *A fast perfect-matching algorithm in random graphs,* SIAM J. Disc. Math. 3, 1990, 48-57.

[GH94] Goldschmidt, O., Hochbaum, D.S., *A polynomial algorithm for the k-cut problem for fixed k,* Math. Oper. Res. 19, 1994, 24-37.

[GHT84] Gilbert, J.R., Hutchinson, J.P., Tarjan, R.E., *A separator theorem for graphs of bounded genus*, J. Alg. 5, 1984, 391-407.

[GHY93] Goldschmidt, O., Hochbaum, D.S., Yu, G., *A modified greedy heuristic for the set covering problem with improved worst case bound,* Inform. Process. Lett. 48, 1993, 305-310.

[GI89] Gusfield, D., Irving, R.W., *The stable marriage problem: structure and algorithms,* MIT Press, Cambridge MA, 1989.

[Gib85] Gibbons, A., *Algorithmic graph theory,* Cambridge University Press, 1985.

[Gil58] Gilbert, E.N., *Gray codes and paths on the n-cube,* Bell System Tech. J. 37, 1958, 815-826.

[GJ76] Garey, M.R., Johnson, D.S., *The complexity of near-optimal graph coloring,* J. ACM 23, 1976, 43-49.

[GJ77] Garey, M.R., Johnson, D.S., *The rectilinear Steiner tree problem is* NP*-complete,* SIAM J. Appl. Math. 32, 1977, 826-834.

[GJ78] Garey, M.R., Johnson, D.S., *Strong* NP*-completeness results: motivation, examples, and implications,* J. ACM 25, 1978, 499-508.

[GJ79] Garey, M.R., Johnson, D.S., *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.

[GJ83] Garey, M.R., Johnson, D.S., *Crossing number is* NP*-complete*, SIAM J. Alg. Disc. Meth. 4, 1983, 312-316.

[GJ85] Golumbic, M.C., Jamison, R.E., *Edge and vertex intersection of paths in trees,* Disc. Math. 55, 1985, 151-159.

[GJS76] Garey, M.R., Johnson, D.S., Stockmeyer, L., *Some simplified* NP*-complete graph problems*, Theoret. Comput. Sci. 1, 1976, 237-267.

[GJT76] Garey, M.R., Johnson, D.S., Tarjan, R.E., *The planar Hamiltonian circuit problem is NP-complete,* SIAM J. Comput. 5, 1976, 704-714.

[GK73] Greenwell, D., Kronk, H., *Uniquely line-colorable graphs,* Canad. Math. Bull. 16, 1973, 525-529.

[GK82] Gabow, H.N., Kariv, O., *Algorithms for edge coloring bipartite graphs and multigraphs,* SIAM J. Comput. 11, 1982, 117-129.

[GK90] Greene, D.H., Knuth, D.E., *Mathematics for the analysis of algorithms,* Birkhäuser, Boston, 3. Auflage, 1990.

[GKP89] Graham, R.L., Knuth, D.E., Patashnik, O., *Concrete mathematics - a foundation for computer science*, Addison-Wesley, Reading MA, 1989, 2. Auflage 1994.

[GKP95] Grigni, M., Koutsoupias, E., Papadimitriou, C.H., *An approximation scheme for planar graph TSP,* Proc. 36th IEEE FoCS, 1995, 640-645.

[GL83] Galil, Z., Leven, D., NP*-completeness of finding the chromatic index of regular graphs,* J. Alg. 4, 1983, 35-44.

[GL89] Golub, G.H., Van Loan, C.F., *Matrix computations,* Johns Hopkins University Press, Baltimore, 1989.

[GL95a] Grötschel, M., Lovász, L., *Combinatorial optimization,* in: [GGL95], 1541-1597.

[GL95b] González, J., Landaeta, O., *A competitive strong spanning tree algorithm for the maximum bipartite matching problem,* SIAM J. Disc. Math. 8, 1995, 186-195.

[GLS81] Grötschel, M., Lovász, L., Schrijver, A., *The ellipsoid method and its consequences in combinatorial optimization,* Combin. 1, 1981, 169-197; Corrigendum: Combin. 4, 1984, 291-295.

[GLS84] Grötschel, M., Lovász, L., Schrijver, A., *Polynomial algorithms for perfect graphs,* in: [BC84], 325-356.

[GLS88] Grötschel, M., Lovász, L., Schrijver, A., *Geometric algorithms and combinatorial optimization,* Springer, Berlin, 1988, 2. verb. Auflage 1993.

[GM60] Gallai, T., Milgram, A.N., *Verallgemeinerung eines graphentheoretischen Satzes von Rédei,* Acta Sci. Math. Szeged 21, 1960, 181-186.

[GM75] Grimmett, G.R., McDiarmid, C.J.H., *On colouring random graphs,* Math. Proc. Camb. Phil. Soc. 77, 1975, 313-324.

[GM90] Gazit, H., Miller, G.L., *Planar separators and the Euclidean norm,* in: Algorithms, Proc. SIGAL '90, Hrsg. Asano, T., Ibaraki, T., Imai, H., Nishizeki, T., Springer LNCS 450, Berlin, 1990, 338-347.

[GMS95] Grötschel, M., Monma, C.L., Stoer, M., *Design of survivable networks,* in: Network models, Handbooks in OR & MS, Vol. 7, Hrsg. Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L., Elsevier, Amsterdam, 1995, 617-672.

[God93] Godsil, C.D., *Algebraic combinatorics,* Chapman & Hall, New York, 1993.

[Göd31] Gödel, K., *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I,* Monatshefte Math. Phys. 38, 1931, 173-198.

[Gol80] Golumbic, M.C., *Algorithmic graph theory and perfect graphs,* Academic Press, New York, 1980.

[Gol88] Golumbic, M.C., *Algorithmic aspects of intersection graphs and representation hypergraphs,* Graphs Combin. 4, 1988, 307-321.

[Gol89] Goldberg, D.E., *Genetic algorithms in search, optimization, and machine learning,* Addison-Wesley, Reading MA, 1989

[Gol93] Goldreich, O., *A taxonomy of proof systems,* ACM SIGACT News 24 (4), 1993, 2-13, und 25 (1), 1994, 22-30.

[Gol95] Goldberg, A.V., *Scaling algorithms for the shortest paths problem,* SIAM J. Comput. 24, 1995, 494-504.

[Gon95] Gonzalez, T.F., *A simple LP-free approximation algorithm for the minimum weight vertex cover problem,* Inform. Process. Lett. 54, 1995, 129-131.

[Gou88] Gould, R., *Graph theory,* Benjamin / Cummings, Menlo Park CA, 1988.

[Gou91] Gould, R.J., *Updating the Hamiltonian problem,* J. Graph Theory 15, 1991, 121-157.

[GP82] Gregory, D.A., Pullman, N.J., *On a clique covering problem of Orlin,* Disc. Math. 41, 1982, 97-99.

[GP86] Gusfield, D., Pitt, L., *Equivalent approximation algorithms for node cover,* Inform. Process. Lett. 22, 1986, 291-294.

[GP88] Gallo, G., Pallottino, S., *Shortest paths algorithms,* Ann. Oper. Res. 13, 1988, 3-79.

[GP92] Gusfield, D., Pitt, L., *A bounded approximation for the minimum cost 2-sat problem,* Algorithmica 8, 1992, 103-117.

[GR88] Gibbons, A., Rytter, W., *Efficient parallel algorithms,* Cambridge Univ. Press, 1988.

[Gra66] Graham, R.L., *Bounds for certain multiprocessing anomalies,* Bell Systems Tech. J. 45, 1986, 1563-1581.

[GRE84] Gilbert, J.R., Rose, D.J., Edenbrandt, A., *A separator theorem for chordal graphs,* SIAM J. Alg. Disc. Meth. 5, 1984, 306-313.

[Gri68] Grinberg, E.J., *Plane homogeneous graphs of degree three without hamiltonian circuits,* Latvian Math. Yearbook 4, 1968, 51-58.

[Gri83a] Griggs, J.R., *Lower bounds on the independence number in terms of the degrees,* J. Combin. Theory B 34, 1983, 22-39.

[Gri83b] Griggs, J.R., *An upper bound on the Ramsey numbers $R(3,k)$,* J. Combin. Theory A 35, 1983, 145-153.

[Gri84] Grinstead, C.M., *The strong perfect graph conjecture for toroidal graphs,* in [BC84], 97-101.

[Grö59] Grötzsch, H., *Zur Theorie der diskreten Gebilde, 7. Mitteilung. Ein Dreifarbensatz für dreikreisfreie Netze auf der Kugel,* Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg, Math.-nat. Reihe 8, 1959, 109-120.

[GRS80] Graham, R.L., Rothschild, B.L., Spencer, J.H., *Ramsey Theory,* Wiley, New York, 1980, 1990[2].

[Grü67] Grünbaum, B., *Convex polytopes,* Wiley-Interscience, London, 1967.

[GS93] Golumbic, M.C., Shamir, R., *Complexity and algorithms for reasoning about time: a graph-theoretic approach,* J. ACM 40, 1993, 1108-1133.

[GSB94] Gupta, R., Smolka, S.A., Bhaskar, S., *On randomization in sequential and distributed algorithms,* ACM Computing Surveys 26, 1994, 7-86.

[GSS93a] Graver, J., Servatius, B., Servatius, H., *Combinatorial rigidity,* Graduate Studies in Mathematics Vol. 2, AMS, 1993.

[GSS93b] Garg, N., Santosh, V.S., Singla, A., *Improved approximation algorithms for biconnected subgraphs via better lower bounding techniques,* Proc. 4th ACM-SIAM SODA, 1993, 103-111.

[GSS93c] Gendreau, M., Soriano, P., Salvail, L., *Solving the maximum clique problem using a tabu search approach,* Ann. Oper. Res. 41, 1993, 385-403.

[GSV94] Garg, N., Saran, H., Vazirani, V.V., *Finding separator cuts in planar graphs within twice the optimal,* Proc. 35th IEEE FoCS, 1994, 14-23.

[GT83] Gabow, H.N., Tarjan, R.E., *A linear-time algorithm for a special case of disjoint set union,* Proc. 15th ACM SToC, 1983, 246-251.

[GT87] Gross, J.L., Tucker, T.W., *Topological graph theory,* Wiley-Interscience, New York, 1987.

[GT91] Gabow, H.N., Tarjan, R.E., *Faster scaling algorithms for general graph-matching problems,* J. ACM 38, 1991, 815-853.

[GTT90] Goldberg. A.V., Tardos, E., Tarjan, R.E., *Network flow algorithms,* in: Paths, flows, and VLSI-layout, Hrsg. Korte, B., Lovász, L., Prömel, H.J., Schrijver, A., Springer, Berlin, 1990, 101-164.

[GTW93] Glover, F., Taillard, E., De Werra, D., *A user's guide to tabu search,* Ann. Oper. Res. 41, 1993, 3-28.

[Gur91] Gurevich, Y., *Average case completeness*, J. Comput. System Sci. 42, 1991, 346-398.

[Gus83] Gusfield, D., *Connectivity and edge-disjoint spanning trees,* Inform. Process. Lett. 16, 1983, 87-89.

[Gus92] Gusfield, D., *Design (with analysis) of efficient algorithms,* in: Computing, Handbooks in OR & MS, Vol. 3, Hrsg. Coffman, E.G., Lenstra, J.K., Rinnooy Kan, A.H.G., North-Holland, Amsterdam, 1992, 375-453.

[Guy60] Guy, R.K., *A combinatorial problem,* Nabla (Proc. Malayan Math. Soc.) 7, 1960, 68-72.

[Guy71] Guy, R.K., *Latest results on crossing numbers,* in: Recent trends in graph theory, Springer-Verlag, New York, 1971, 143-156.

[Guy72] Guy, R.K., *Crossing number of graphs,* in: Graph theory and applications, Hrsg. Alavi, Y., Lick, D.R., White, A.T., Springer LNM 303, Berlin, 1972, 111-124.

[GVV96] Garg, N., Vazirani, V.V., Yannakakis, M., *Primal-dual approximation algorithms for integral flow and multicut in trees,* erscheint in Algorithmica, 1996; vorläufige Version in Proc. 20th ICALP 1993, LNCS 700.

[GW92] Gabow, H.N., Westermann, H.H., *Forests, frames, and games: algorithms for matroid sums and applications,* Algorithmica 7, 1992, 465-497.

[GW94a] Goemans, M.X., Williamson, D.P., *New $\frac{3}{4}$-approximation algorithms for the maximum satisfiability problem,* SIAM J. Disc. Math. 7, 1994, 656-666.

[GW94b] Goemans, M.X., Williamson, D.P., *.878-approximation algorithms for MAX CUT and MAX 2SAT,* Proc. 26th ACM SToC, 1994, 422-431; eine überarbeitete Fassung erscheint im J. ACM; erhältlich über WWW an der URL „http://theory.lcs.mit.edu/~goemans/".

[GW95] Goemans, M.X., Williamson, D.P., *A general approximation technique for constrained forest problems,* SIAM J. Comput. 24, 1995, 296-317.

[GW96] Goemans, M.X., Williamson, D.P., *The primal-dual method for approximation algorithms and its application to network design problems,* in: Approximation algorithms for NP-hard problems, Hrsg. Hochbaum, D., PWS Publishers, angekündigt für August 1996; erhältlich über WWW an der URL „http://theory.lcs.mit.edu/~goemans/".

[Gya90] Gyárfás, A., *A simple lower bound on edge coverings by cliques,* Disc. Math. 85, 1990, 103-104.

[Had43] Hadwiger, H., *Über eine Klassifikation der Streckenkomplexe,* Vierteljahresschr. Naturforsch. Ges. Zürich 88, 1943, 133-142.

[Had75] Hadlock, F.O., *Finding a maximum cut of a planar graph in polynomial time*, SIAM J. Comput. 4, 1975, 221-225.

[Haj57] Hajós, G., *Über eine Art von Graphen,* Internat. Math. Nachr. 11, 1957, Problem 65.

[Haj61] Hajós, G., *Über eine Konstruktion nicht n-färbbarer Graphen,* Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg, Math.-Nat. Reihe X/1, 1961, 113-4.

[Hal35] Hall, P., *On representatives of subsets,* J. LMS 10, 1935, 26-30.

[Hal67a] Hall, M., Jr., *Combinatorial theory,* Blaisdell, Waltham MA, 1967; Wiley & Sons, New York, 1986[2].

[Hal67b] Halin, R., *Unterteilungen vollständiger Graphen in Graphen mit unendlicher chromatischer Zahl,* Abh. Math. Sem. Univ. Hamburg 31, 1967, 156-165.

[Hal71] Halin, R., *Studies in minimally connected graphs,* in: Combinatorial mathematics and its applications, Hrsg. Welsh, D.J.A., Academic Press, New York, 1971, 129-136.

[Hal89] Halin, R., *Graphentheorie,* Wiss. Buchgesellschaft, Darmstadt, 2. Auflage 1989.

[Hal93a] Halldórsson, M.M., *A still better performance guarantee for approximate graph coloring,* Inform. Process. Lett. 45, 1993, 19-23.

[Hal93b] HALLDÓRSSON, M.M., *Approximating the minimum maximal independence number,* Inform. Process. Lett. 46, 1993, 169-172.

[Hal96a] HALLDÓRSSON, M.M., *Parallel and on-line graph coloring,* erscheint in J. Alg., vorläufige Version in ISAAC'92, Springer LNCS 650, 61-70.

[Hal96b] HALLDÓRSSON, M.M., *Approximating k-set cover and complementary graph coloring,* Proc. 5th IPCO'96, Hrsg. Cunningham, W.H., McCormick, S.T., Queyranne, M., Springer LNCS 1084, 1996, 118-131.

[Hamil] HAMILTON, W.R., *Account of the Icosian calculus,* Proc. Royal Irish Acad. 6, 1853-7, 415-416.

[Har57] HARARY, F., *On arbitrarily traceable graphs and directed graphs*, Scripta Math. 23, 1957, 37-41.

[Har62] HARARY, F., *The maximum connectivity of a graph,* Proc. Nat. Acad. Sci. U.S. 48, 1962, 1142-1146.

[Har74] HARARY, F., *Graphentheorie*, (Korrigierte Übersetzung der englischsprachigen Originalausgabe von 1969 bei Addison-Wesley) Oldenbourg Verlag, München, 1974.

[Hås96] HÅSTAD, J., *Clique is hard to approximate within* $n^{1-\epsilon}$, Manuskript, Royal Institute of Technology, March 1996.

[Hay85] HAYWARD, R.B., *Weakly triangulated graphs,* J. Combin. Theory B 39, 1985, 200-208.

[HB78] HEMMINGER, R.L., BEINEKE, L.W., *Line graphs and line digraphs,* in: [BW78], 271-305.

[Hea90] HEAWOOD, P.J., *Map colour theorem,* Quart. J. Pure Appl. Math. 24, 1890, 332-338.

[Hea98] HEAWOOD, P.J., *On the four-colour map theorem,* Quart. J. Pure Appl. Math. 29, 1898, 270-285.

[Hee69] HEESCH, H., *Untersuchungen zum Vierfarbenproblem,* B.I. Hochschulskripten 810/810a/-810b, Bibliographisches Institut, Mannheim, 1969.

[Hen90] HENDRY, G.R.T., *Extending cycles in graphs,* Disc. Math. 85, 1990, 59-72.

[Het64] HETYEI, G., *Rechteckige Konfigurationen, die durch* 2×1 *Rechtecke überdeckt werden können,* (auf Ungarisch), P'ecsi Tan. Főisk. Közl. 8, 1964, 351-367.

[HHR69] HARARY, F., HEDETNIEMI, S.T., ROBINSON, R.W., *Uniquely colorable graphs,* J. Combin. Theory 6, 1969, 264-270; Corrigendum: J. Combin. Theory B 9, 1970, 221.

[HHW88] HARARY, F., HAYES, J.P., WU, H.J., *A survey of the theory of hypercube graphs,* Comput. Math. Appl. 15, 1988, 277-289.

[Hie73] HIERHOLZER, C., *Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren*, Math. Ann. 6, 1873, 30-32.

[HJ63] HALIN, R., JUNG, H.A., *Über Minimalstrukturen von Graphen, insbesondere von n-fach zusammenhängenden Graphen,* Math. Ann. 152, 1963, 75-94.

[HJ85] HORN, R.A., JOHNSON, C.R., *Matrix analysis*, Cambridge University Press, 1985.

[HK73] HOPCROFT, J.E., KARP, R.M., *An* $n^{2.5}$ *algorithm for maximum matchings in bipartite graphs,* SIAM J. Comput. 2, 1973, 225-231.

[HK95] HENZINGER, M.R., KING, V., *Randomized dynamic graph algorithms with polylogarithmic time per operation,* Proc. 27th ACM SToC, 1995, 519-527.

[HL88] HAKEN, A., LUBY, M., *Steepest descent can take exponential time for symmetric connection networks,* Complex Systems 2, 1988, 191-196.

[HL89] HO, C.-W., LEE, R.C.T., *Counting clique trees and computing perfect elimination schemes in parallel,* Inform. Process. Lett. 31, 1989, 61-68.

[HL90] HEDETNIEMI, S.T., LASKAR, R.C., (Hrsg.), *Topics on domination,* Disc. Math. 86, 1990, und Ann. Disc. Math 48, North-Holland, Amsterdam, 1991.

[HL94] HASSIN, R., LAHAV, S., *Maximizing the number of unused colors in the vertex coloring problem,* Inform. Process. Lett. 52, 1994, 87-90.

[HL96a] Halldórsson, M.M., Lau, H.C., *Low-degree graph partitioning via local search with applications to constraint satisfaction, max cut, and 3-coloring,* Manuscript, February 1996.

[HL96b] Hofmeister, T., Lefmann, H., *A combinatorial design approach to MAXCUT,* in: Proc. STACS'96, Hrsg. Puech, C., Reischuk, R., Springer LNCS 1046, 1996, 441-452.

[HM88] Holton, D.A., McKay, B.D., *The smallest non-Hamiltonian 3-connected cubic planar graphs have 38 vertices*, J. Combin. Theory B 45, 1988, 305-319.

[HM93] Hanson, D., MacGillivray, G., *On small triangle-free graphs,* Ars Combin. 35, 1993, 257-263.

[HMS93] Helman, P., Moret, B.M.E., Shapiro, H.D., *An exact characterization of greedy structures,* SIAM J. Disc. Math. 6, 1993, 274-283.

[HN53] Harary, F., Norman, R.Z., *The dissimilarity characteristic of Husimi trees,* Ann. Math. 58, 1953, 134-141.

[HN79] Hsu, W.-L., Nemhauser, G.L., *Easy and hard bottleneck location problems,* Disc. Appl. Math. 1, 1979, 209-215.

[HO94] Hao, J., Orlin, J.B., *A faster algorithm for finding the minimum cut in a directed graph,* J. Alg. 17, 1994, 424-446.

[Hoa87] Hoáng, C.T., *On a conjecture of Meyniel,* J. Combin. Theory B 42, 1987, 302-312.

[Hoa94] Hoáng, C.T., *Efficient algorithms for minimum weighted colouring of some classes of perfect graphs,* Disc. Appl. Math. 55, 1994, 133-143.

[Hoc82] Hochbaum, D.S., *Approximation algorithms for the set covering and vertex cover problems,* SIAM J. Comput. 11, 1982, 555-556.

[Hoc96] Hochbaum, D.S., (Hrsg.), *Approximation algorithms for NP-hard problems,* PWS Publishers, Boston, 1996.

[Hof87] Hofri, M., *Probabilistic analysis of algorithms,* Springer, New York, 1987.

[Hol81a] Holyer, I., *The* NP*-completeness of some edge-partition problems,* SIAM J. Comput. 10, 1981, 713-717.

[Hol81b] Holyer, I., *The* NP*-completeness of edge-coloring,* SIAM J. Comput. 10, 1981, 718-720.

[HP66] Harary, F., Prins, G., *The block-cutpoint-tree of a graph,* Publ. Math. Debrecen 13, 1966, 103-107.

[HP73] Harary, F., Palmer, E.M., *Graphical enumeration*, Academic Press, New York, 1973.

[HPS94] Hougardy, S., Prömel, H.J., Steger, A., *Probabilistically checkable proofs and their consequences for approximation algorithms,* in: Trends in Discrete Mathematics, Hrsg. Deuber, W., Prömel, H.J., Voigt, B., Disc. Math. 136, 1994, 175-223; auch: Band 9 der Serie „Topics in discrete mathematics", North-Holland, Amsterdam, 1995.

[HR92a] Hoffman, D.G., Rodger, C.A., *The chromatic index of complete multipartite graphs,* J. Graph Theory 16, 1992, 159-163.

[HR92b] Hwang, F.K., Richards, D.S., *Steiner tree problems,* Networks 22, 1992, 55-89.

[HR93] Hsu, T., Ramachandran, V., *Finding a smallest augmentation to biconnect a graph,* SIAM J. Comput. 22, 1993, 889-912.

[HR94] Halldórsson, M.M., Radhakrishnan, J., *Greed is good: approximating independent sets in sparse and bounded-degree graphs,* Proc. 26th ACM SToC, 1994, 439-448; erscheint in Algorithmica, 1996.

[HR95] Halldórsson, M.M., Radhakrishnan, J., *Improved approximations of independent sets in bounded-degree graphs via subgraph removal,* Nordic J. Comput., Winter 1995, 275-92; vorläufige Version in: Algorithm theory - SWAT '94, Hrsg. Schmidt, E.M., Skyum, S., Springer LNCS 824, Berlin, 1994, 195-206.

[HRT86] Hanson, D., Robinson, G.C., Toft, B., *Remarks on the graph colour theorem of Hajós,* Proc. 17th southeastern conf. combinatorics, graph theory, and computing, Congr. Numer. 55, 1986, 69-76.

[HRW92] Hwang, F.K., Richards, D.S., Winter, P., *The Steiner tree problem,* Ann. Disc. Math. 53, North-Holland, Amsterdam, 1992.

[HS58] Hajnal, A., Surányi, J., *Über die Auflösung von Graphen in vollständige Teilgraphen,* Ann.Univ.Sci. Budapest Eötvös.Sect.Math.1, 1958, 113-121.

[HS65] Hartmanis, J., Stearns, R.E., *On the computational complexity of algorithms,* Trans. AMS 117, 1965, 285-306.

[HS70] Hajnal, A., Szemerédi, E., *Proof of a conjecture of Erdős,* in: Combinatorial theory and its applications, Vol. II, Hrsg. Erdős, P., Rényi, A., Sós, V.T., Colloq. Math. Soc. J. Bolyai 4, North Holland, Amsterdam, 1970, 601-623.

[HS78] Horowitz, E., Sahni, S., *Fundamentals of computer algorithms,* Computer Science Press, 1978.

[HS85] Hochbaum, D.S., Shmoys, D.B., *A best possible heuristic for the k-center problem,* Math. Oper. Res. 10, 1985, 180-184.

[HS93] Holton, D.A., Sheehan, J., *The Petersen graph,* Cambridge Univ. Press, 1993.

[HS94] Halldórsson, M.M., Szegedy, M., *Lower bounds for on-line graph coloring,* in: Dynamic and on-line algorithms, Hrsg. Ausiello, G., Haliano, G.F., Theoret. Comput. Sci. 130, 1994, 163-174.

[HSA93] Horowitz, E., Sahni, S., Anderson-Freed, S., *Fundamentals of data structures in C,* Freeman, New York, 1993.

[Hsu93] Hsu, W.-L., *A simple test for interval graphs,* in: Proc. 18th graph theoretic concepts in computer science, Hrsg. Mayr, E.W., Springer LNCS 657, 1993, 11-16.

[HT73] Hopcroft, J.E., Tarjan, R.E., *Dividing a graph into triconnnected components,* SIAM J. Comput. 2, 1973, 135-158.

[HT74] Hopcroft, J.E., Tarjan, R.E., *Efficient planarity testing,* J. ACM 21, 1974, 549-568.

[HT93] Hujter, M., Tuza, Z., *The number of maximal independent sets in triangle-free graphs,* SIAM J. Disc. Math. 6, 1993, 284-288.

[HU79] Hopcroft, J.E., Ullman, J.D., *Introduction to automata theory, languages, and computation,* Addison-Wesley, Reading MA, 1979.

[HV91] Haglin, D.J., Venkatesan, S.M., *Approximation and intractability results for the maximum cut problem and its variants,* IEEE Trans. Comput. 40, 1991, 110-113.

[HW74] Hopcroft, J.E., Wong, J.K., *Linear time algorithm for isomorphism of planar graphs,* Proc. 6th ACM SToC, 1974, 172-184.

[HW89a] Hertz, A., De Werra, D., (Hrsg.), *Graph coloring and variations,* Disc. Math. 74, 1989, und Ann. Disc. Math. 39, North-Holland, Amsterdam, 1989.

[HW89b] Hilton, A.J.W., Wilson, R.J., *Edge-colorings of graphs: a progress report,* in: Graph theory and its applications: east and west, Hrsg. Capobianco, M.F., Guan, M., Hsu, D.F., Tian, F., New York Acad. Sci., 1989, 241-249.

[Iba87] Ibaraki, T., *Enumerative approaches to combinatorial optimization,* Ann. Oper. Res. 10 + 11, 1987.

[IK75] Ibarra, O.H., Kim. C.E., *Fast approximation algorithms for the knapsack and sum of subset problems,* J. ACM 22, 1975, 463-468.

[IR78] Itai, A., Rodeh, M., *Finding a minimum circuit in a graph,* SIAM J. Comput. 7, 1978, 413-423.

[IT94] Ivanov, A.O., Tuzhilin, A.A., *Minimal Networks – The Steiner Problem and its generalizations,* CRC Press, Boca Raton, 1994.

[Jac83] Jacobs, K., *Einführung in die Kombinatorik,* de Gruyter, Berlin, 1983.

[Jac93] Jackson, B., *Hamilton cycles in regular 2-connected graphs,* in: Surveys in combinatorics, Hrsg. Walker, K., LMS Lecture Notes Series Vol. 187, Cambridge Univ. Press, 1993, 191-210.

[Jae88] Jaeger, F., *Nowhere-zero flow problems*, in: [BW88], 71-95.

[JaJ92] JáJa, J., *An introduction to parallel algorithms,* Addison-Wesley, Reading MA, 1992.

[JAMS89] Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C., *Optimization by simulated annealing: an experimental evaluation; Part I: graph partitioning,* Oper. Res. 37, 1989, 865-892.

[JAMS91] Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C., *Optimization by simulated annealing: an experimental evaluation; Part II: graph coloring and number partitioning,* Oper. Res. 39, 1991, 378-406.

[Jan96] Jansen, K., *Persönliche Mitteilung,* Mai 1996.

[Jar30] Jarník, V., *O jistém problému minimálním,* (in Tschechisch), Práca Moravské Přírodovědecké Společnosti 6, 1930, 57-63.

[Jen96] Jensen, T.R., *Persönliche Mitteilung,* 1996.

[Jer92] Jerrum, M., *Large cliques elude the Metropolis process,* Random Struct. Alg. 3, 1992, 347-359.

[Joh74a] Johnson, D.S., *Approximation algorithms for combinatorial problems,* J. Comput. System Sci. 9, 1974, 256-278.

[Joh74b] Johnson, D.S., *Worst case behavior of graph coloring algorithms,* Proc. 5th southeastern conf. combin., graph theory, and computing, Congr. Numer. X, 1974, 513-527.

[Joh84a] Johnson, D.S., *The* NP*-completeness column: an ongoing guide*, [Solving NP-hard problems quickly (on average)], J. Alg. 5, 1984, 284-299.

[Joh84b] Johnson, D.S., *The* NP*-completeness column: an ongoing guide*, [Randomized algorithms and complexity classes], J. Alg. 5, 1984, 433-447.

[Joh85] Johnson, D.S., *The* NP*-completeness column: an ongoing guide*, [Graph restrictions and their effect], J. Alg. 6, 1985, 434-451.

[Joh87] Johnson, D.S., *The* NP*-completeness column: an ongoing guide*, [The many faces of polynomial time], J. Alg. 8, 1987, 285-303.

[Joh90a] Johnson, D.S., *The* NP*-completeness column: an ongoing guide*, [Inhalt Folge 1-22], J. Alg. 11, 1990, 144-151.

[Joh90b] Johnson, D.S., *A catalog of complexity classes*, in: Handbook of theoretical computer science, Vol. A: Algorithms and complexity, Hrsg. Van Leeuwen, J., Elsevier Science Publ., Amsterdam, 1990, 68-161.

[Joh90c] Johnson, D.S., *Local optimization and the traveling salesman problem*, in: Automata, Languages, and Programming, Hrsg. Paterson, M.S., Springer LNCS 443, New York, 1990, 446-461.

[Joh92] Johnson, D.S., *The* NP*-completeness column: an ongoing guide*, [The tale of the second prover], J. Alg. 13, 1992, 502-524.

[Jor69] Jordan, C., *Sur les assemblages de lignes,* J. Reine Angew. Math. 70, 1869, 185-190.

[Jor95] Jordán, T., *On the optimal vertex-connectivity augmentation,* J. Combin. Theory B 63, 1995, 8-20.

[JPY88] Johnson, D.S., Papadimitriou, C.H., Yannakakis, M., *How easy is local search?,* J. Comput. System Sci. 37, 1988, 79-100.

[JRR95] Jünger, M., Reinelt, G., Rinaldi, G., *The travelling salesman problem,* in: Network models, Handbooks in OR & MS, Vol. 7, Hrsg. Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L., Elsevier, Amsterdam, 1995, 225-330.

[JRT95] Jünger, M., Reinelt, G., Thienel, S., *Practical problem solving with cutting plane algorithms in combinatorial optimization,* in: Combinatorial optimization, Hrsg. Cook, W., Lovász, L., Seymour, P.D., DIMACS series in discrete mathematics and theoretical computer science, Vol. 20, AMS, 1995, 111-152.

[JS89a] Jones, R.H., Steele, N.C., *Mathematics in communication theory,* Ellis Horwood, Chichester, 1989.

[JS89b] Jerrum, M., Sinclair, A., *Approximating the permanent,* SIAM J. Comput. 18, 1989, 1149-1178.

[JS93] Jerrum, M., Sorkin, G.B., *Simulated annealing for graph bisection,* Proc. 34th IEEE FoCS, 1993, 94-103.

[JT95] Jensen, T.R., Toft, B., *Graph coloring problems,* Wiley-Interscience, New York, 1995.

[Jun82] Jungnickel, D., *Transversaltheorie,* Akad. Verlagsgesell. Geest & Portig, Leipzig, 1982.

[Jun94] Jungnickel, D., *Graphen, Netzwerke und Algorithmen,* BI Wissenschaftsverlag, Mannheim, 1987, 3. Auflage 1994.

[Jur94] Jurkiewicz, S., *Planar graphs, Hamilton cycles and extreme independence number,* in: Applications of Combinatorial Optimization, Hrsg. Ribeiro, C.C, Maculan, N., Ann. Oper. Res. 50, 1994, 281-293.

[JV83] Johnson, D.B., Venkatesan, S.M., *Partition of planar flow networks*, Proc. 24th IEEE FoCS, 1983, 259-263.

[JVV86] Jerrum, M.R., Valiant, L.G., Vazirani, V.V., *Random generation of combinatorial structures from a uniform distribution,* Theoret. Comput. Sci. 43, 1986, 169-188.

[JYP88] Johnson, D.S., Yannakakis, M., Papadimitriou, C.H., *On generating all maximal independent sets,* Inform. Process. Lett. 27, 1988, 119-123.

[Kah95] Kahale, N., *Eigenvalues and expansion of regular graphs,* J. ACM 42, 1995, 1091-1106.

[Kam76] Kameda, T., *On maximally distant spanning trees of a graph,* Computing 17, 1976, 115-119.

[Kar72] Karp, R.M., *Reducibility among combinatorial problems*, in: Complexity of computer computations, Hrsg. Miller, R.E., Thatcher, J.W., Plenum Press, 1972, 85-103.

[Kar84] Karmarkar, N., *A new polynomial time algorithm for linear programming*, Combin. 4, 1984, 373-395.

[Kar91a] Karp, R.M., *An introduction to randomized algorithms*, Disc. Appl. Math. 34, 1991, 165-201.

[Kar91b] Karloff, H., *Linear programming,* Birkhäuser, Boston, 1991.

[Kar95] Karoński, M., *Random graphs,* in: [GGL95], 351-380.

[Kas67] Kasteleyn, P.W., *Graph theory and crystal physics*, in: Graph theory and theoretical physics, Hrsg. Harary, F., Academic Press, London, 1967, 43-110.

[Kau90] Kauffman, L.H., *Map coloring and the vector cross product*, J. Combin. Theory B 48, 1990, 145-154.

[Kei85] Keil, J.M., *Finding Hamiltonian circuits in interval graphs,* Inform. Process. Lett. 20, 1985, 201-206.

[Kel84] Kelmans, A.K., *A strengthening of the Kuratowski planarity criterion for 3-connected graphs*, Disc. Math. 51, 1984, 215-220.

[Kem79] Kempe, A.B., *On the geographical problem of the four colours,* Amer. J. Math. 2, 1879, 193-200.

[KGS95] Kaller, D., Gupta, A., Shermer, T., *The χ_t-coloring problem,* in: STACS'95, Hrsg. Mayr, W., Puech, C., Springer LNCS 900, 1995, 409-420.

[Kha79] Khachian, L.G., *A polynomial algorithm for linear programming*, aus dem Russischen in: Soviet Math. Doklady, 20, 1979, 191-194.

[Kim95] Kim, J.H., *On Brooks' theorem for sparse graphs,* Combin. Probab. Comput. 4, 1995, 97-132.

[Kin90] Kingston, J.H., *Algorithms and data structures – design, correctness, analysis,* Addison-Wesley, Sydney, 1990.

[Kir47] Kirchhoff, G.R., *Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Ströme geführt wird,* Annalen der Physik und Chemie 72, 1847, 497-508.

[KJ85] Kubale, M., Jackowski, B., *A generalized implicit enumeration algorithm for graph coloring,* Comm. ACM 28, 1985, 412-418.

[KK54] Kelly, J.B., Kelly, L.M., *Paths and circuits in critical graphs,* Amer. J. Math. 76, 1954, 786-792.

[KK82] Karmarkar, N., Karp, R.M., *An efficient approximation scheme for the one-dimensional bin packing problem,* Proc. 23rd IEEE FoCS, 1982, 312-320.

[KKT95] Karger, D.R., Klein, P.N., Tarjan, R.E., *A randomized linear-time algorithm to find minimum spanning trees,* J. ACM 42, 1995, 321-328.

[KL70] Kernighan, B., Lin, S., *An efficient heuristic procedure for partitioning graphs,* Bell Systems Tech. J. 49, 1970, 291-307.

[KL95] Kolen, A.W.J., Lenstra, J.K. *Combinatorics in operations research,* in: [GGL95], 1875-1910,

[KLC90] Ko, M.T., Lee, R.C.T., Chang, J.S., *An optimal approximation algorithm for the rectilinear m-center problem,* Algorithmica 5, 1990, 341-352.

[Kle36] Kleene, S.C., *General recursive functions of natural numbers*, Math. Ann. 112, 1936, 727-742.

[Kle43] Kleene, S.C., *Recursive predicates and quantifiers*, Trans. AMS 53, 1943, 41-73.

[Kle67] Kleinert, M., *Die Dicke des n-dimensionalen Würfel-Graphen,* J. Combin. Theory 3, 1967, 10-15.

[Kle70] Kleitman, D.J., *The crossing number of $K_{5,n}$,* J. Combin. Theory 9, 1970, 315-323.

[KLMR85] Karp, R.M., Lenstra, J.K., McDiarmid, C.J.H., Rinnooy Kan, A.H.G., *Probabilistic analysis,* in: Combinatorial optimization: annotated bibliographies, Hrsg. O'hEigeartaigh, M., Lenstra, J.K., Rinnooy Kan, A.H.G., Wiley, New York, 1985, 52-88.

[Klo94] Kloks, T., *Treewidth – Computations and approximations,* Springer LNCS 842, Berlin, 1994.

[KM72] Klee, V., Minty, G.J., *How good is the simplex algorithm?*, in: Inequalities III, Hrsg. Shisha, O., Academic Press, New York, 1972, 159-175.

[KM74] Kameda, T., Munro, I, *A $\mathcal{O}(|V|\cdot|E|)$ algorithm for maximum matching of graphs,* Comput. 12, 1974, 91-98.

[KM89] Korte, N., Möhring, R.H., *An incremental linear-time algorithm for recognizing interval graphs,* SIAM J. Comput. 18, 1989, 68-81.

[KMB81] Kou, L., Markowsky, G., Berman, L., *A fast algorithm for Steiner trees,* Acta Inform. 15, 1981, 141-145.

[KMN93] Karp, R.M., Motwani, R., Nisan, N., *Probabilistic analysis of network flow algorithms,* Math. Oper. Res. 18, 1993, 71-97.

[KMS94] Karger, D., Motwani, R., Sudan, M., *Approximate graph coloring by semidefinite programming,* Proc. 35th IEEE FoCS, 1994, 2-13.

[KMSV95] Khanna, S., Motwani, R., Sudan, M., Vazirani, U., *On syntactic versus computational views of approximability,* Electron. Colloq. Comput. Compl. TR95-023, 1995; erhältlich über WWW an der URL „http://www.eccc.uni-trier.de/eccc/".

[KN93] Khuller, S., Naor, J., *Flow in planar graphs: a survey of recent results,* in: Planar graphs, Hrsg. Trotter, W.T., DIMACS series in discrete mathematics and theoretical computer science, Vol. 9, AMS, 1993, 59-84.

[Kne55] Kneser, M., *Aufgabe 300,* Jahresber. DMV 58, 1955, 27.

[Knu73] Knuth, D.E., *The art of computer programming, Vol.1: Fundamental algorithms,* Addison-Wesley, Reading MA, 1968, 2. Auflage 1973.

[Knu81] Knuth, D.E., *The art of computer programming, Vol.2: Seminumerical algorithms,* Addison-Wesley, Reading MA, 2. Auflage 1981.

[Knu76] Knuth, D.E., *Big omicron and big omega and big theta*, SIGACT News 8, 1976, 18-24.

[Knu93] KNUTH, D.E., *The Stanford GraphBase: a platform for combinatorical computing*, Addison-Wesley, 1993.

[Knu94] KNUTH, D.E., *The sandwich theorem*, Electron. J. Combin. 1, 1994, # A1; erhältlich an der URL „http://ejc.math.gatech.edu:8080".

[Koe36] KOEBE, P., *Kontaktprobleme der konformen Abbildung,* Ber. Verhandl. Sächs. Akad. Wissensch. Leipzig, Math.-Phys. Klasse, 88, 1936, 141-164.

[Kön05] KÖNIG, D., *On map coloring*, (in Ungarisch), Math. Phys. Lapok 14, 1905, 193-200.

[Kön16] KÖNIG, D., *Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre*, Math. Ann. 77, 1916, 453-465.

[Kön31] KÖNIG, D., *Graphen und Matrizen,* in Ungarisch, Mat. Fiz. Lapok 38, 1931, 116-119.

[Kön32] KÖNIG, D., *Über trennende Knotenpunkte in Graphen (nebst Anwendungen auf Determinanten und Matrizen,* Acta. Lit. Sci. Sect. Sci. Math. (Szeged) 6, 1932-34, 155-179.

[Kön36] KÖNIG, D., *Theorie der endlichen und unendlichen Graphen*, Akad. Verlagsgesellschaft, Leipzig, 1936; Nachdruck: Teubner, Leipzig, 1986; Birkhäuser, Boston, 1989.

[Koh91] KOHAYAKAWA, Y., *A note on induced cycles in Kneser graphs,* Combin. 11, 1991, 245-251.

[Kor79] KORMAN, S.M., *The graph-colouring problem,* in: Combinatorial optimization, Hrsg. Christofides, N., Mingozzi, A., Toth, P., Sandi, C., Wiley, New York, 1979, 211-235.

[Kos84] KOSTOCHKA, A.V., *Lower bound on the Hadwiger number of graphs by their average degree,* Combinatorica 4, 1984, 307-316.

[Koz92] KOZEN, D.C., *The design and analysis of algorithms,* Springer-Verlag, New York, 1992.

[KP92] KOUTSOUPIAS, E., PAPADIMITRIOU, C.H., *On the greedy algorithm for satisfiability,* Inform. Process. Lett. 43, 1992, 53-55.

[KPR87] KOLAITIS, P.G., PRÖMEL, H.J., ROTHSCHILD, B.L., *K_{l+1}-free graphs: asymptotic structure and a 0-1-law,* Trans. AMS 303, 1987, 637-671.

[KPS90] KORTE, B., PRÖMEL, H.J., STEGER, A., *Steiner trees in VLSI-layout,* in: Paths, flows, and VLSI-layout, Hrsg. Korte, B., Lovász, L., Prömel, H.J., Schrijver, A., Springer, Berlin, 1990, 185-214.

[KR90] KARP, R.M., RAMACHANDRAN, V., *Parallel algorithms for shared-memory machines,* in: Handbook of theoretical computer science, Vol. A: Algorithms and complexity, Hrsg. Van Leeuwen, J., Elsevier Science Publ., Amsterdam, 1990, 869-941.

[KR95] KLEIN, P., RAVI, R., *A nearly best-possible approximation algorithm for node-weighted Steiner trees,* J. Alg. 19, 1995, 104-115.

[KR96] KHULLER, S., RAGHAVACHARI, B., *Improved approximation algorithms for uniform connectivity problems,* erscheint in J. Alg., 1996, vorläufige Version in Proc. 27th ACM SToC, 1995.

[Kra43] KRAUSZ, J., *Démonstration nouvell d'une théorème de Whitney sur les réseaux*, in Ungarisch, Mat.Fiz.Lapok. 50, 1943, 75-89.

[Kre88] KRENTEL, M.W., *The complexity of optimization problems,* J. Comput. System Sci. 36, 1988, 490-509.

[Kre90] KRENTEL, M.W., *On finding and verifying locally optimal solutions,* SIAM J. Comput. 19, 1990, 742-749.

[Kri75] KRISHNAMOORTHY, M.S., *An* NP*-hard problem in bipartite graphs,* SIGACT News 7, 1975, 26.

[Kri89] KRIZ, I., *A hypergraph-free construction of highly chromatic graphs without short cycles,* Combin. 9, 1989, 227-229.

[KRT94] KING, V., RAO, S., TARJAN, R.E., *A faster deterministic maximum flow algorithm,* J. Alg. 17, 1994, 447-474.

[Kru56] KRUSKAL, J.B., *On the shortest spanning subtree of a graph and the traveling salesman problem,* Proc. AMS 7, 1956, 48-50.

[KS83] Komlós, J., Szemerédi, E., *Limit distribution for the existence of hamiltonian cycles in a random graph,* Disc. Math. 43, 1983, 55-63.

[KS96a] Komlós, J., Szemerédi, E., *Topological cliques in graphs II,* Combin. Probab. Comput. 5, 1996, 79-90.

[KS96b] Komlós, J., Simonovits, M., *Szemerédi's regularity lemma and its applications in graph theory,* in: Combinatorics – Paul Erdős is eighty, Vol. 2, Hrsg. Miklós, D., Sós, V.T., Szönyi, T., János Bolyai Mathematical Society, Budapest, angekündigt 1996.

[KST54] Kővari, T., Sós, V.T., Turán, P., *On a problem of Zarankiewicz,* Colloq. Math. 3, 1954, 50-57.

[KSW78] Kou, L.T., Stockmeyer, L.J., Wong, C.K., *Covering edges by cliques with regard to keyword conflicts and intersection graphs,* Comm. ACM 21, 1978, 135-139.

[KT93] Khuller, S., Thurimella, R., *Approximation algorithms for graph augmentation,* J. Alg. 14, 1993, 214-225.

[KT94a] Kratochvil, J., Tuza, Z., *Algorithmic complexity of list colorings,* Disc. Appl. Math. 50, 1994, 297-302.

[KT94b] Kolaitis, P.G., Thakur, M.N., *Logical definability of NP optimization problems,* Inform. Comput. 115, 1994, 321-353.

[KT95] Kolaitis, P.G., Thakur, M.N., *Approximation properties of NP minimization classes,* J. Comput. System Sci. 50, 1995, 391-411.

[Kuc89] Kučera, L., *Graphs with small chromatic numbers are easy to color,* Inform. Process. Lett. 30, 1989, 233-236.

[Kuc90] Kučera, L., *Combinatorial algorithms,* Hilger, Bristol, 1990.

[Kuc91] Kučera, L., *The greedy algorithm is a bad probabilistic algorithm,* J. Alg. 12, 1991, 674-684.

[Kuh55] Kuhn, H.W., *The Hungarian method for the assignment problem,* Naval Res. Logist. Quart. 2, 1955, 83-97.

[Kuh56] Kuhn, H.W., *Variants of the Hungarian method for assignment problems,* Naval Res. Logist. Quart. 3, 1956, 253-258.

[Kun86] Kung, J.P.S., *A source book in matroid theory,* Birkhäuser, Boston, 1986.

[Kur30] Kuratowski, C., *Sur le problème des courbes gauches en topologie,* Fund. Math. 15, 1930, 271-283.

[KV94] Khuller, S., Vishkin, U., *Biconnectivity approximations and graph carvings,* J. ACM 41, 1994, 214-235.

[KZ95] Karpinski, M., Zelikovsky, A., *1.757 und 1.267-approximation algorithms for the network and rectilinear Steiner tree problems,* Electron. Colloq. Comput. Compl. TR95-003, 1995, erhältlich über WWW an der URL „http://www.eccc.uni-trier.de/eccc/".

[LA87] Van Laarhoven, P.J.M., Aarts, E.H.L., *Simulated annealing: theory and applications,* Reidel, Dordrecht, 1987 (verbesserter Nachdruck 1988).

[Lad75] Ladner, R.E., *On the structure of polynomial-time reducibility,* J. ACM 22, 1975, 155-171.

[Läu91] Läuchli, P., *Algorithmische Graphentheorie,* Birkhäuser, Basel, 1991.

[Lap92a] Laporte, G., *The traveling salesman problem: an overview of exact and approximate algorithms,* Europ. J. Oper. Res. 59, 1992, 231-247.

[Lap92b] Laporte, G., *The vehicle routing problem: an overview of exact and approximate algorithms,* Europ. J. Oper. Res. 59, 1992, 345-358.

[Law76a] Lawler, E.L., *Combinatorial optimization: networks and matroids,* Holt, Rinehart & Winston, New York, 1976.

[Law76b] Lawler, E.L., *A note on the complexity of the chromatic number problem,* Inform. Process. Lett. 5, 1976, 66-67.

[Law79] Lawler, E.L., *Graphical algorithms and their complexity,* in: Foundations of computer science II, Part 1, Hrsg. Apt, K.R., De Bakker, J.W., Mathematisch Centrum, Amsterdam, 1979, 25-32.

[LEC67] Lempel, A., Even, S., Cederbaum, I., *An algorithm for planarity testing of graphs*, in: Theory of graphs, Hrsg. Rosenstiehl, P., Gordon and Breach, New York, 1967, 215-232.

[Lee61] Lee, C.Y., *An algorithm for path connection and its applications,* IRE Trans. Electr. Comput. EC-10, 1961, 346-365.

[Lee90] Van Leeuwen, J., *Graph algorithms*, in: Handbook of theoretical computer science, Vol. A: Algorithms and complexity, Hrsg. Van Leeuwen, J., Elsevier Science Publ., Amsterdam, 1990, 526-631.

[Leh74] Lehot, P.G.H., *An optimal algorithm to detect a line graph and output its root graph,* J. ACM 21, 1974, 569-575.

[Lei79] Leighton, F.T., *A graph coloring algorithm for large scheduling problems*, J. Res. Nat. Bur. Stand. 84, 1979, 489-506.

[Lei80] Leiserson, C.E., *Area efficient graph layouts (for VLSI)*, Proc. 21st IEEE FoCS, 1980, 270-281.

[Lei83] Leighton, F.T., *Complexity issues in VLSI*, MIT Press, Cambridge MA, 1983.

[Lei92] Leighton, F.T., *Introduction to parallel algorithms and architectures: arrays · trees · hypercubes,* Morgan Kaufmann, San Mateo CA, 1992.

[Len90] Lengauer, T., *Combinatorial algorithms for integrated circuit layout,* Teubner, Stuttgart / Wiley & Sons, New York, 1990.

[Lev73] Levin, L.A., *Universal sorting problems*, Problemy Peredachi Informatsii 9, 115-116, englische Übersetzung: Problems Inform. Transmission 9, 1973, 265-266.

[Lich82] Lichtenstein, D., *Planar formulae and their uses,* SIAM J. Comput. 11, 1982, 329-393.

[Lin76] Linial, N., *A lower bound for the circumference of a graph,* Disc. Math. 15, 1976, 297-300.

[Lit74] Little, C.H.C., *An extension of Kasteleyn's method of enumerating the 1-factors of planar graphs*, in: Combinatorial Mathematics, Proc. 2nd Australian Conf., Hrsg. Holton, D., Springer LNM 403, Berlin, 1974, 63-72.

[Liu90] Liu, Y., *On Boolean characterizations of planarity and planar embeddings of graphs,* Ann. Oper. Res. 24, 1990, 165-174.

[Liu92] Liu, J.-H., *The multifrontal method for sparse matrix solution: theory and practice,* SIAM Rev. 34, 1992, 82-109.

[LK73] Lin, S., Kernighan, B.W., *An effective heuristic for the traveling salesman problem,* Oper. Res. 21, 1973, 489-516.

[LLRS85] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B., (Hrsg.), *The traveling salesman problem – A guided tour of combinatorial optimization,* Wiley & Sons, Chichester, 1985.

[LLS75] Ladner, R.E., Lynch, N.A., Selman, A.L., *A comparison of polynomial time reducibilities*, Theoret. Comput. Sci. 1, 1975, 103-123.

[LLW88] Linial, N., Lovász, L., Wigderson, A., *Rubber bands, convex embeddings and graph connectivity,* Combinatorica 8, 1988, 91-102.

[LO86] Lesniak, L., Oellermann, O.R., *An Eulerian exposition,* J. Graph Theory 10, 1986, 277-297.

[Lov68] Lovász, L., *On the chromatic number of finite set-systems,* Act. Math. Acad. Sci. Hung. 19, 1968, 59-67.

[Lov72a] Lovász, L., *Normal hypergraphs and the perfect graph conjecture,* Disc. Math. 2, 1972, 253-267; auch: [BC84].

[Lov72b] Lovász, L., *A characterization of perfect graphs,* J. Combin. Theory B 13, 1972, 95-98; auch: [BC84].

[Lov72c] Lovász, L., *A note on factor-critical graphs,* Studia Sci. Math. Hung. 7, 1972, 279-280.

[Lov73] Lovász, L., *Coverings and colorings of hypergraphs,* Proc. 4th Southeastern Conf. Combinatorics, Graph Theory, and Computing, 1973, 3-12.

[Lov75a] Lovász, L., *Three short proofs in graph theory,* J. Combin. Theory B 19, 1975, 269-271.

[Lov75b] Lovász, L., *On the ratio of optimal fractional and integral covers,* Disc. Math. 13, 1975, 383-390.

[Lov78] Lovász, L., *Kneser's conjecture, chromatic number, and homotopy,* J. Combin. Theory A 25, 1978, 319-324..

[Lov79] Lovász, L., *Combinatorial problems and exercises,* North-Holland, Amsterdam, 1979, 2. Auflage 1993.

[Lov82] Lovász, L., *Bounding the independence number of a graph,* Ann. Disc. Math. 16, 1982, 213-223.

[Lov83] Lovász, L., *Perfect graphs,* in: [BW83], 55-87.

[Lov95] Lovász, L., *Randomized algorithms in combinatorial optimization,* in: Combinatorial optimization, Hrsg. Cook, W., Lovász, L., Seymour, P.D., DIMACS series in discrete mathematics and theoretical computer science, Vol. 20, AMS, 1995, 153-179.

[LP86] Lovász, L., Plummer, M.D., *Matching theory,* Ann. Disc. Math. 29, North-Holland, Amsterdam, 1986.

[LPP89] Lewis, J., Peyton, B., Pothen, A., *A fast algorithm for reordering sparse matrices for parallel factorization,* SIAM J. Sci. Statist. Comput. 10, 1989, 1156-1173.

[LPS88] Lubotzky, A., Phillips, R., Sarnak, P., *Ramanujan graphs,* Combin. 8, 1988, 261-277.

[LPV81] Lev, G.F., Pippenger, N., Valiant, L.G., *A fast parallel algorithm for routing in permutation networks,* IEEE Trans. Comput. C-30, 1981, 93-100.

[LR79] Lenstra, J.K., Rinnooy Kan, A.H.G., *Complexity of packing, covering and partitioning problems,* in: Packing and covering in combinatorics, Hrsg. Schrijver, A., Math. Centre Tracts 106, Amsterdam, 1979, 275-291.

[LRT79] Lipton, R.J., Rose, D.J., Tarjan, R.E., *Generalized nested dissection,* SIAM J. Numer. Anal. 16, 1979, 346-358.

[LSS89] Lovász, L., Saks, M., Schrijver, A., *Orthogonal representations and connectivity of graphs,* Lin. Alg. Appl. 114/115, 1989, 439-454.

[LT79] Lipton, R.J., Tarjan, R.E., *A separator theorem for planar graphs,* SIAM J. Appl. Math. 36, 1979, 177-189.

[LT80] Lipton, R.J., Tarjan, R.E., *Applications of a planar separator theorem,* SIAM J. Comput. 9, 1980, 615-627.

[Lu92] Lu, X., *Hamiltonian games,* J. Combin. Theory B 55, 1992, 18-32.

[Lu94] Lu, X., *A Chvátal-Erdős condition for Hamiltonian graphs,* J. Graph Theory 18, 1994, 791-800.

[Lub94] Lubotzky, A., *Discrete groups, expanding graphs and invariant measures,* Birkhäuser, Basel, 1994.

[Lub95] Lubotzky, A., *Cayley graphs: eigenvalues, expanders and random walks,* in: Surveys in combinatorics, Hrsg. Rowlinson, P., LMS Lecture Notes Series Vol. 218, Cambridge Univ. Press, 1995, 155-189.

[Luz95] Luz, C.J., *An upper bound on the independence number of a graph computable in polynomial time,* Oper. Res. Lett. 18, 1995, 139-145.

[LW92a] Van Lint, J.H., Wilson, R.M., *A course in combinatorics,* Cambridge University Press, 1992.

[LW92b] Van Leeuwen, J., Widmayer, P., *Fundamental algorithms and data structures,* in: Computing, Handbooks in OR & MS, Vol. 3, Hrsg. Coffman, E.G., Lenstra, J.K., Rinnooy Kan, A.H.G., North-Holland, Amsterdam, 1992, 323-373.

[LW95a] Luby, M., Wigderson, A., *Pairwise independence and derandomization,* ICSI TR-95-035, July 1995, 62 Seiten; erhältlich über WWW an der URL „http://www.icsi.berkeley.edu/~luby/pair_sur.html".

[LW95b] Loebbing, M., Wegener, I., *The number of knight's tours equals 33,439,123,484,294 – counting with binary decision diagrams,* Electron. Colloq. Comput. Compl. TR95-047, 1995.

[MacL37a] MacLane, S., *A structural characterization of planar combinatorial graphs,* Duke Math. J. 3, 1937, 340-372.

[MacL37b] MacLane, S., *A combinatorial condition for planar graphs,* Fund. Math. 28, 1937, 22-32.

[Mad67] Mader, W., *Homomorphieeigenschaften und mittlere Kantendichte von Graphen,* Math. Ann. 174, 1967, 265-268.

[Mad72] Mader, W., *Über minimal n-fach zusammenhängende, unendliche Graphen und ein Extremalproblem,* Arch. Math. 23, 1972, 553-560.

[Mad73] Mader, W., *1-Faktoren von Graphen,* Math. Ann. 201, 1973, 269-282.

[Mah92] Mahmoud, H.M., *Evolution of Random Search Trees,* Wiley & Sons, New York, 1992.

[Mal88] Malkevitch, J., *Polytopal graphs,* in: [BW88], 169-188.

[Mal94a] Malitz, S.M., *Graphs with E edges have pagenumber $\mathcal{O}(\sqrt{E})$,* J. Alg. 17, 1994, 71-84.

[Mal94b] Malitz, S.M., *Genus g graphs have pagenumber $\mathcal{O}(\sqrt{g})$,* J. Alg. 17, 1994, 85-109.

[Man07] Mantel, W., *Problem 28,* Wiskundige Opgaven 10, 60-61.

[Man83] Mansfield, A., *Determining the thickness of graphs is* NP*-hard,* Math. Proc. Cambr. Phil. Soc. 93, 1983, 9-23.

[Mar51] Markov, A.A., *Theory of algorithms,* Israel Program for Scientific Translations, Jerusalem, 1961 (aus dem Russischen).

[Mat68] Matula, D.W., *A min-max theorem for graphs with application to graph coloring,* SIAM Rev. 10, 1968, 481-2.

[Mat76] Matula, D.W., *The largest clique size in a random graph,* Tech. Rep. Dept. Comput. Sci., Southern Methodist University, Dallas, 1976.

[Mat87] Matula, D.W., *Determining edge connectivity in $\mathcal{O}(nm)$,* Proc. 28th FoCS, 1987, 249-251.

[Mat93] Matula, D.W., *A linear time $2+\epsilon$ approximation algorithm for edge connectivity,* Proc. 4th Ann. ACM-SIAM SoDA, New York/Philadelphia, 1993, 500-504.

[May75] Mayer, J., *Inégalités nouvelles dans le problème des quatre couleurs,* J. Combin. Theory B 19, 1975, 119-149.

[May90] Mayr, E.W., *Basic parallel algorithms in graph theory,* in: Computational graph theory, Hrsg. Tinhofer, G., Mayr, E., Noltemeier, H., Syslo, M.M., Computing Suppl. 7, 1990, 69-91.

[MB83] Matula, D.W., Beck, L.L., *Smallest-last ordering and clustering and graph coloring algorithms,* J. ACM 30, 1983, 417-427.

[McD72] McDiarmid, C.J.H., *The solution of a timetabling problem,* J. Inst. Math. Appl. 9, 1972, 23-34.

[McD79] McDiarmid, C.J.H., *Determining the chromatic number of a graph,* SIAM J. Comput. 8, 1979, 1-14.

[McG94] McGuinness, S., *The greedy clique decomposition of a graph,* J. Graph Theory 18, 1994, 427-430.

[McK84] McKee, T.A., *Recharacterizing Eulerian: intimations of new duality,* Disc. Math. 51, 1984, 237-242.

[MD58] Mendelsohn, N.S., Dulmage, A.L., *Some generalizations of the problem of distinct representatives,* Canad. J. Math. 10, 1958, 230-241.

[Meh84] Mehlhorn, K., *Data structures and algorithms 2: Graph algorithms and* NP*-completeness,* Springer, Berlin, 1984.

[Meh88] MEHLHORN, K., *Datenstrukturen und effiziente Algorithmen, Bd.1: Sortieren und Suchen,* Teubner, Stuttgart, 1988.

[Mei62] MEI-KO, K., *Graphic programming using odd or even points,* Chinese Math. 1, 1962, 273-277.

[Men27] MENGER, K., *Zur allgemeinen Kurventheorie*, Fund. Math. 10, 1927, 95-115.

[Mey76] MEYNIEL, H., *On the perfect graph conjecture,* Disc. Math. 16, 1976, 339-342; auch: [BC84].

[Mic96] MICHALEWICZ, Z., *Genetic algorithms + data structures = evolution programs,* Springer, Berlin, 3. Auflage 1996.

[Mih92] MIHÓK, P., *An extension of Brook's theorem,* in: 4th Czechosl. Symp. Combin., Graphs, and Complexity, Hrsg. Nešetřil, J., Fiedler, M., Ann. Disc. Math. 51, Elsevier/North-Holland, Amsterdam, 1992.

[Mil80] MILLER, G.L., *Isomorphism testing for graphs of bounded genus,* Proc. 12th ACM SToC, 1980, 225-235.

[Mil86] MILLER, G.L., *Finding small simple cycle separators for 2-connected planar graphs*, J. Comput. System Sci. 32, 1986, 265-279.

[Min78] MINC, H., *Permanents*, Encycl.Math.Appl.6, Hrsg. Rota, G.-C., Addison-Wesley, Reading MA, 1978.

[Mir71a] MIRSKY, L., *A dual of Dilworth's decomposition theorem,* Amer. Math. Monthly 78, 1971, 876-877.

[Mir71b] MIRSKY, L., *Transversal theory,* Academic Press, New York, 1971.

[Mit76] MITCHEM, J., *On various algorithms for estimating the chromatic number of a graph,* Comput. J. 19, 1976, 182-183.

[MM65] MOON, J.W., MOSER, L., *On cliques in graphs,* Israel J. Math. 3, 1965, 23-28.

[MMI72] MATULA, D.W., MARBLE, G., ISAACSON, J.D., *Graph coloring algorithms,* in: Graph theory and computing, Hrsg. Read, R.C., Academic Press, New York, 1972.

[MNN94] MOTWANI, R., NAOR, J., NAOR, M., *The probablistic method yields deterministic parallel algorithms*, J. Comput. System Sci. 49, 1994, 478-516.

[MNR96] MOTWANI, R., NAOR, J., RAGHAVAN, P., *Randomized approximation algorithms in combinatorial optimization,* in: Approximation algorithms, Hrsg. Hochbaum, D., PWS Publishers, angekündigt Dezember 1995.

[MNW90] MORAN, S., NEWMAN, I., WOLFSTAHL, Y., *Approximation algorithms for covering a graph by vertex-disjoint paths of maximum total weight,* Networks 20, 1990, 55-64.

[Möh85] MÖHRING, R.H., *Algorithmic aspects of comparability graphs and interval graphs,* in: Graphs and order, Hrsg. Rival, I., Reidel, Dordrecht, 1985, 41-101.

[Moo59] MOORE, E.F., *The shortest path through a maze,* Proc. Internat. Symp. Theory Switching, Part II, Harvard Univ. Press, Cambridge MA, 1959, 285-292.

[Mot92] MOTWANI, R., *Lecture notes on approximation algorithms*, Vol. 1, Report No. STAN-CS-92-1435, Stanford University, 1992; erhältlich über WWW an der URL „http://theory.stanford.edu/people/motwani/books.html".

[Mot94] MOTWANI, R., *Average-case analysis of algorithms for matchings and related probelems*, J. ACM 41, 1994, 1329-1356.

[MP90] MIDDENDORF, M., PFEIFFER, F., *On the complexity of recognizing perfectly orderable graphs,* Disc. Math. 80, 1990, 327-333.

[MP93a] MIDDENDORF, M., PFEIFFER, F., *On the complexity of the disjoint paths problem,* Combin. 13, 1993, 97-107.

[MP93b] MOHAR, B., POLJAK, S., *Eigenvalues in combinatorial optimization,* in: Combinatorial and graph-theoretical problems in linear algebra, Hrsg. Brualdi, R.A., Friedland, S., Klee, V., Springer, New York, 1993, 107-151.

[MP95] Mahadev, N. V. R., Peled, U. N., *Threshold graphs and related topics,* Annals of Discrete Mathematics 56, North-Holland, Amsterdam, 1995.

[MR84] Mirkin, B.G., Rodin, S.N., *Graphs and genes,* Springer-Verlag, Berlin, 1984.

[MR95a] Motwani, R., Raghavan, P., *Randomized algorithms,* Cambridge Univ. Press, 1995.

[MR95b] Mahajan, S., Ramesh, H., *Derandomizing semidefinite programming based approximation algorithms,* Proc. 36th IEEE FoCS, 1995, 162-169.

[MR96] Marathe, M.V., Ravi, S.S., *On approximation algorithms for the minimum satisfiability problem,* Inform. Process. Lett. 58, 1996, 23-29.

[MS72] Meyer, A.R., Stockmeyer, L.J., *The equivalence problem for regular expressions with squaring requires exponential space*, Proc. 13th IEEE Ann. Symp. Switch. Autom. Theory, 1972, 125-129.

[MS85] Monien, B., Speckenmeyer, E., *Ramsey numbers and an approximation algorithm for the vertex cover problem,* Acta Inform. 22, 1985, 115-123.

[MS90] Matula, D.W., Shahrokhi, F., *Sparsest cuts and bottlenecks in graphs,* Disc. Appl. Math. 27, 1990, 113-123.

[MS95] McConnell, R.M., Spinrad, J.P., *Modular decomposition and transitive orientation,* Technical Report 475/1995, Technische Universität Berlin, FB Mathematik, 1995; erhältlich über WWW an der URL „ftp://ftp.math.tu-berlin.de/pub/Preprints/combi".

[MT87] Moffat, A., Takaoka, T., *An all pairs shortest path algorithm with expected time* $\mathcal{O}(n^2 \log n)$, SIAM J. Comput. 16, 1987, 1023-1031.

[MT97] Mohar, B., Thomassen, C., *Graphs on surfaces,* Johns Hopkins University Press, angekündigt für 1997.

[MTV91] Miller, G.L., Teng,S.-H., Vavasis,S.A., A unified geometric approach to graph separators, Proc. 32nd IEEE FOCS, 1991, 538-547.

[Mun57] Munkres, J., *Algorithms for the assignment and transportation problems,* J. SIAM 5, 1957, 32-38.

[Mur92] Murphy, O.J., *Computing independent sets in graphs with large girth,* Disc. Appl. Math. 35, 1992, 167-170.

[MV80] Micali, S., Vazirani, V.V., *An* $O(\sqrt{n}m)$ *algorithm for finding maximum matching in general graphs*, Proc. 21st IEEE FoCS, 1980, 17-27.

[MVV87] Mulmuley, K., Vazirani, U.V., Vazirani, V.V., *Matching is as easy as matrix inversion*, Combin. 7, 1987, 105-113.

[Myc55] Mycielski, J., *Sur le coloriage des graphes,* Colloq. Math. 3, 1955, 161-162.

[Nash61] Nash-Williams, C.St.J.A., *Edge-disjoint spanning trees of finite graphs,* J. LMS 36, 1961, 445-450.

[Nash64] Nash-Williams, C.St.J.A., *Decomposition of finite graphs into forests,* J. LMS 39, 1964, 12.

[Nash71] Nash-Williams, C.St.J.A., *Hamilton arcs and circuits,* in: Recent trends in graph theory, Hrsg. Capobianco, M., Frechen, J.B., Krolik, M., LNM 186, Springer, Berlin, 1971, 197-210.

[NC88] Nishizeki, T., Chiba, N., *Planar graphs: theory and algorithms*, North-Holland, Amsterdam, 1988.

[Nes66] Nešetřil, J., *k-chromatic graphs without cycles of length at most 7,* (auf Tschechisch), Comment. Math. Univ. Carolinae 7, 1966, 373-376.

[Nes95] Nešetřil, J., *Ramsey theory,* in: [GGL95], 1331-1403.

[Neu53] von Neumann, J., *A certain zero-sum two-person game equivalent to the optimal assignment problem*, in: Contributions to the theory of games II, Hrsg. Kuhn, H.W., Tucker, A.W., Princeton Univ. Press, 1953.

[NG56] Nordhaus, E.A., Gaddum, J.W., *On complementary graphs,* Amer. Math. Monthly 63, 1956, 175-177.

[NI92a] Nagamochi, H., Ibaraki, T., *Linear time algorithms for finding a sparse k-connected spanning subgraph of a k-connected graph,* Algorithmica 7, 1992, 583-596.

[NI92b] Nagamochi, H., Ibaraki, T., *Computing edge-connectivity in multigraphs and capacitated graphs,* SIAM J. Disc. Math. 5, 1992, 54-66.

[Nic66] Nicholson, T., *Finding the shortest route between two points in a network,* Comput. J. 9, 1966, 275-280.

[Nis90] Nishizeki, T., *Planar graph problems,* in: Computational graph theory, Hrsg. Tinhofer, G., Mayr, E., Noltemeier, H., Syslo, M.M., Computing Suppl. 7, Springer, Wien, 1990, 53-68.

[NM93] Neumann, K., Morlock, M., *Operations research,* Hanser Verlag, München, 1993.

[NN94] Nesterov, Y., Nemirovskii, A., *Interior point polynomial methods in convex programming,* SIAM, Philadelphia, 1994.

[NOI94] Nagamochi, H., Ono, T., Ibaraki, T., *Implementing an efficient minimum capacity cut algorithm,* Math. Progr. 67, 1994, 325-341.

[NP94] Nishizeki, T., Poljak, S., *k-connectivity and decomposition of graphs into forests,* Disc. Appl. Math. 55, 1994, 295-301.

[NR79] Nešetřil, J., Rödl, V., *A short proof of the existence of highly chromatic hypergraphs without short cycles,* J. Combin. Theory B 27, 1979, 225-227.

[NT93] Van Ngoc, N., Tuza, Z., *Linear-time approximation algorithms for the Max Cut problem,* Combin. Probab. Comput. 2, 1993, 201-210.

[NW88] Nemhauser, G.L., Wolsey, L.A., *Integer and combinatorial optimization,* Wiley, New York, 1988.

[NZN95] Nakano, S., Zhou, X., Nishizeki, T., *Edge-coloring algorithms,* in: Computer science today, Hrsg. Van Leeuwen, J., Springer LNCS 1000, 1995, 172-183.

[OCF76] Ohtsuki, T., Cheung, L.K., Fujisawa, T., *Minimal triangulation of a graph and optimal pivoting order in a sparse matrix,* J. Math. Anal. Appl. 54, 1976, 622-633.

[OD72] Orlova, G.I., Dorfman, Y.G., *Finding the maximum cut in a graph,* Engrg. Cybernetics 10, 1972, 502-506.

[OG89] Otten, R., Van Ginneken, L., *The annealing algorithm,* Kluwer Acad. Publ., Boston, 1989.

[OKSW94] Orponen, P., Ko, K., Schöning, U., Watanabe, O., *Instance complexity,* J. ACM 41, 1994, 96-121.

[OR81] Opsut, R.J., Roberts, F.S., *On the fleet maintenance, mobile radio frequency, task assignment, and traffic phasing problems,* in: The theory and applications of graphs, Hrsg. Chartrand, G., et al., Wiley & Sons, New York, 1981, 479-492.

[Ore51] Ore, O., *A problem regarding the tracing of graphs*, Elem. Math. 6, 1951, 49-53.

[Ore55] Ore, O., *Graphs and matching theorems*, Duke Math. J. 22, 1955, 625-639.

[Ore60] Ore, O., *Note on Hamilton circuits*, Amer. Math. Monthly 67, 1960, 55.

[Ore63] Ore, O., *Hamilton connected graphs*, J. Math. Pure Appl. 42, 1963, 21-27.

[Ore67] Ore, O., *The four color problem*, Academic Press, New York, 1967.

[OW93] Ottmann, T., Widmayer, P., *Algorithmen und Datenstrukturen,* BI Wissenschaftsverlag, Mannheim, 2. Auflage, 1993.

[Ox92] Oxley, J.G., *Matroid theory*, Oxford University Press, 1992.

[PA95] Pach, J., Agarwal, P.K., *Combinatorial Geometry,* Wiley Interscience, New York, 1995.

[Pad74] Padberg, M.W., *Perfect zero-one matrices,* Math. Progr. 6, 1974, 180-196.

[Pad95] Padberg, M., *Linear optimization and extensions,* Springer, Berlin, 1995.

[Pal85] PALMER, E.M., *Graphical evolution*, Wiley & Sons, New York, 1985.

[Pan95] PANIK, M.J., *Linear programming: mathematics, theory and algorithms,* Kluwer Acad. Publ., Dordrecht, 1995.

[Pap92] PAPADIMITRIOU, C.H., *The complexity of the Lin-Kernighan heuristic for the traveling salesman problem,* SIAM J. Comput. 21, 1992, 450-465.

[Pap94] PAPADIMITRIOU, C.H., *Computational complexity,* Addison-Wesley, Reading MA, 1994.

[Pet91] PETERSEN, J., *Die Theorie der regulären Graphen,* Acta Math. 15, 1891, 193-220.

[Ple75] PLESNÍK, J., *Critical graphs of given diameter,* Acta Fac. Rerum Natur. Univ. Comenian. Math. 30, 1975, 71-93.

[Ple87] PLESNÍK, J., *A heuristic for the p-center problem in graphs,* Disc. Appl. Math. 17, 1987, 263-268.

[Plu93a] PLUMMER, M.D., *Matching and vertex packing: how "hard" are they?,* in: Quo vadis, graph theory?, Hrsg. Gimbel, J., Kennedy, J.W., Quintas, L.V., Ann. Disc. Math. 55, North-Holland, Amsterdam, 1993, 275-312.

[Plu93b] PLUMMER, M.D., *Well-covered graphs: a survey,* Quaestiones Math. 16, 1993, 253-287.

[PM81] PAZ, A., MORAN, S., *Non deterministic polynomial optimization problems and their approximations,* Theoret. Comput. Sci. 15, 1981, 251-277.

[Pol74] POLJAK, S., *A note on stable sets and colorings of graphs,* Comm. Math. Univ. Carolinae 15, 1974, 307-309.

[Pot93] POTVIN, J.-Y., *The traveling salesman problem: a neural network perspective,* ORSA J. Comput. 5, 1993, 328-348.

[PR76] PARTHASARATHY, K.R., RAVINDRA, G., *The strong perfect-graph conjecture is true for $K_{1,3}$-free graphs,* J. Combin. Theory B 21, 1976, 212-223.

[PR79] PARTHASARATHY, K.R., RAVINDRA, G., *The validity of the strong perfect-graph conjecture for $(K_4 - e)$-free graphs,* J. Combin. Theory B 26, 1979, 98-100.

[PR87] PÓLYA, G., READ, R.C., *Combinatorial enumeration of groups, graphs, and chemical compounds,* Springer-Verlag, New York, 1987.

[PR88] PARKER, R.G., RARDIN, R.L., *Discrete optimization,* Academic Press, Boston, 1988.

[Pra75] PRATT, V., *Every prime has a succinct certificate*, SIAM J. Comput. 4, 1975, 214-220.

[Pri57] PRIM, R.C., *Shortest connection networks and some generalizations,* Bell System Tech. J. 36, 1957, 1389-1401.

[Prö94] PRÖMEL, H.J., *Probabilistische Argumente und algorithmische Probleme in der Diskreten Mathematik,* in: Jahrbuch Überblicke Mathematik 1994, Hrsg. Chatterji, S.D., Fuchssteiner, B., Kulisch, U., Liedl, R., Vieweg & Sohn, Braunschweig Wiesbaden, 1994, 28-46.

[PRT81] POLJAK, S., RÖDL, V., TURZÍK, D., *Complexity of representation of graphs by set systems,* Disc. Appl. Math. 3, 1981, 301-312.

[PS77] PAPADIMITRIOU, C.H., STEIGLITZ, K., *On the complexity of local search for the traveling salesman problem*, SIAM J. Comput. 6, 1977, 76-83.

[PS82] PAPADIMITRIOU, C.H., STEIGLITZ, K., *Combinatorial optimization: algorithms and complexity*, Prentice-Hall, Englewood Cliffs, 1982.

[PS90] PROSKUROWSKI, A., SYSLO, M.M., *Efficient computations in tree-like graphs,* in: Computational graph theory, Hrsg. Tinhofer, G., Mayr, E., Noltemeier, H., Syslo, M.M., Computing Suppl. 7, Springer, Wien, 1990, 1-15.

[PS92a] PRÖMEL, H.J., STEGER, A., *Almost all graphs are perfect*, Combin. Probab. Comput. 1, 1992, 53-79.

[PS92b] PRÖMEL, H.J., STEGER, A., *Coloring clique-free graphs in linear expected time*, Random Struct. Alg. 3, 1992, 375-402.

[PT82] Poljak, S., Turzik, D., *A polynomial algorithm for constructing a large bipartite subgraph, with an application to a satisfiability problem,* Canad. J. Math. 24, 1982, 519-524.

[PT95a] Poljak, S., Tuza, Z., *Maximum cuts and largest bipartite subgraphs,* in: Combinatorial optimization, Hrsg. Cook, W., Lovász, L., Seymour, P., DIMACS series in discrete mathematics and theoretical computer science, Vol. 20, AMS, 1995, 181-244.

[PT95b] Plotkin, S., Tardos, E., *Improved bounds on the max-flow min-cut ratio for multicommodity flows,* Combinatorica 15, 1995, 425-434.

[PU95] Pitassi, T., Urquhart, A., *The complexity of the Hajós calculus,* SIAM J. Disc. Math. 8, 1995, 464-483.

[Pul83a] Pullman, N.J., *Clique coverings of graphs - a survey,* Proc. conf. combin. math. X, Hrsg. Casse, L.R.A., Springer LNM 1036, Berlin, 1983, 72-85.

[Pul83b] Pulleyblank, W.R., *Polyhedral combinatorics,* in: Mathematical programming – the state of the art – Bonn 1982, Hrsg. Bachem, A., Grötschel, M., Korte, B., Springer-Verlag, Berlin, 1983, 312-345.

[Pul95] Pulleyblank, W.R., *Matchings and extensions,* in: [GGL95], 179-232.

[PV90] Plehn, J., Voigt, B., *Finding minimally weighted subgraphs,* in: Proc. 16th workshop on graph theoretic concepts in computer science, Springer LNCS 484, 1990, 18-29.

[PW89] Petford, A.D., Welsh, D.J.A., *A randomised 3-colouring algorithm,* Disc. Math. 74, 1989, 253-261.

[PY81] Papadimitriou, C.H., Yannakakis, M., *The clique problem for planar graphs,* Inform. Process. Lett. 13, 1981, 131-133.

[PY91] Papadimitriou, C.H., Yannakakis, M., *Optimization, approximation, and complexity classes,* J. Comput. System Sci. 43, 1991, 425-440.

[PY93] Papadimitriou, C.H., Yannakakis, M., *The traveling salesman problem with distances one ore two,* Math. Oper. Res. 18, 1993, 1-11.

[Rab60] Rabin, M.O., *Degree of difficulty of computing a function and a partial ordering of recursive sets*, Tech. Report 2, Hebrew Univ., Jerusalem, 1960.

[Rab76] Rabin, M.O., *Probabilistic algorithms,* in: Algorithms and complexity: New directions and recent results, Hrsg. Traub, J.F., Academic Press, New York, 1976, 21-39.

[Rad95] Radzik, T., *Fast deterministic approximation for the multicommodity flow problem,* Proc. 6th ACM-SIAM SODA, 1995, 486-492.

[Rag88] Raghavan, P., *Probabilistic construction of deterministic algorithms: approximating packing integer programs*, J. Comput. System Sci. 37, 1988, 130-143.

[Rag90] Raghavan, P., *Lecture notes on randomized algorithms*, RC 15340 (#68237), IBM Research Division, Yorktown Heights, NY, 1990.

[Ram30] Ramsey, F.P., *On a problem of formal logic,* Proc. LMS 30, 1930, 264-286.

[Rao87] Rao, S., *Finding near optimal separators in planar graphs,* Proc. 28th IEEE FoCS, 1987, 225-237.

[Rav82] Ravindra, G., *Meyniel's graphs are strongly perfect,* J. Combin. Theory B 33, 1982, 187-190; auch: [BC84].

[Rea69] Read, R.C., *Teaching graph theory to a computer,* in: Recent progress in combinatorics, Hrsg. Tutte, W.T., Academic Press, New York, 1969, 161-173.

[Rec89] Recski, A., *Matroid theory and its applications,* Springer, Berlin, 1989.

[Réd34] Rédei, L., *Ein kombinatorischer Satz,* Acta Litt. Szeged 7, 1934, 39-43.

[Ree93] Reeves, C.R., (Hrsg.), *Modern heuristic techniques for combinatorial problems,* Blackwell Scientific Publ., Oxford, 1993.

[Rei84] Reichmeider, P.F., *The equivalence of some combinatorial matching theorems,* Polygonal Publ. House, Washington NJ, 1984.

[Rei90] Reischuk, K.R., *Einführung in die Komplexitätstheorie*, Teubner, Stuttgart, 1990.

[Rei93] Reif, R.H., (Hrsg.), *Synthesis of parallel algorithms,* Morgan Kaufmann Publ., San Mateo CA, 1993.

[Rei94] Reinelt, G., *The traveling salesman - Computational solutions for TSP applications,* Springer LNCS 840, Berlin, 1994.

[Rey77] Reyner, S.W., *An analysis of a good algorithm for the subtree problem,* SIAM J. Comput. 6, 1977, 730-733.

[RH87] Ravi, S.S., Hunt, H.B., *An application of the planar separator theorem to counting problems*, Inform. Process. Lett. 25, 1987, 317-321.

[Rin54] Ringel, G., *Bestimmung der Maximalzahl der Nachbargebiete auf nichtorientierbaren Flächen*, Math. Ann. 127, 1954, 181-214.

[Rin55] Ringel, G., *Über drei kombinatorische Probleme am n-dimensionalen Würfel und Würfelgitter*, Abh. Math. Sem. Univ. Hamburg 20, 1955, 10-19.

[Rin65] Ringel, G., *Das Geschlecht des vollständigen paaren Graphen*, Abh. Math. Sem. Univ. Hamburg 28, 1965, 139-150.

[Rin74] Ringel, G., *Map color theorem*, Springer, Berlin, 1974.

[Rin87] Rinooy Kan, A.H.G., *Probabilistic analysis of algorithms,* in: Surveys in combinatorial optimization, Hrsg. Martello, S., Laporte, G., Minoux, M., Ribeiro, C., Ann. Disc. Math. 31, North-Holland, Amsterdam, 1987, 365-384.

[Riv85] Rival, I., (Hrsg.), *Graphs and order: the role of graphs in the theory of ordered sets and its applications,* Reidel, Dordrecht, 1985.

[Rob39] Robbins, H.E., *A theorem on graphs, with an application to a problem in traffic control,* Amer. Math. Monthly 46, 1939, 281-283.

[Rob78] Roberts, F.S., *Graph theory and its applications to problems of society,* SIAM, Philadelphia, 1978.

[Rob85] Roberts, F.S., *Applications of edge coverings by cliques,* Disc. Appl. Math. 10, 1985, 93-109.

[Rob86] Robson, J.M., *Algorithms for maximum independent sets,* J. Alg. 7, 1986, 425-440.

[Rob91] Roberts, F.S., *From garbage to rainbows: generalizations of graph coloring and their applications,* in: Graph theory, combinatorics, and applications, Vol.2, Hrsg. Alavi, Y., Chartrand, G., Oellermann, O.R., Schwenk, A.J., Wiley & Sons, New York, 1991, 1031-1052.

[Rob93] Roberts, F.S., *New directions in graph theory (with an emphasis on the role of applications,* in: Quo vadis, graph theory?, Hrsg. Gimbel, J., Kennedy, J.W., Quintas, L.V., Ann. Disc. Math. 55, North-Holland, Amsterdam, 1993, 13-43.

[Rog67] Rogers, H., Jr., *Theory of recursive functions and effective computability*, McGraw-Hill, New York, 1967, Neudruck: MIT Press, Cambridge MA, 1987.

[Ros70] Rose, D.J., *Triangulated graphs and the elimination process*, J. Math. Anal. Appl. 32, 1970, 597-609.

[Ros72] Rose, D.J., *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in: Graph theory and computing, Hrsg. Read, R.C., Academic Press, 1972, 183-217.

[Ros74] Rose, D.J., *On simple characterizations of k-trees*, Disc. Math. 7, 1974, 317-322.

[Rou73] Roussopoulos, N.D., *A* $\max\{m,n\}$ *algorithm for determining the graph H from its line graph G,* Inform. Process. Lett. 2, 1973, 108-112.

[Rou87] O'Rourke, J., *Art gallery theorems and algorithms,* Oxford Univ. Press, New York, 1987.

[RS83] Robertson, N., Seymour, P.D., *Graph minors I: excluding a forest*, J. Combin. Theory B 35, 1983, 39-61.

[RS84] Robertson, N., Seymour, P.D., *Graph minors III: planar tree-width*, J. Combin. Theory B 36, 1984, 49-64.

[RS85a] Robertson, N., Seymour, P.D., *Graph minors - a survey*, in: Surveys in Combinatorics, Hrsg. Anderson, I., LMS Lecture Notes Series Vol. 103, Cambridge Univ. Press, 1985, 153-171.

[RS85b] Robertson, N., Seymour, P.D., *Disjoint paths - a survey*, SIAM J. Alg. Disc. Meth. 6, 1985, 300-305.

[RS86] Robertson, N., Seymour, P.D., *Graph minors II: algorithmic aspects of tree-width*, J. Alg. 7, 1986, 309-322.

[RS90] Robertson, N., Seymour, P.D., *Graph minors VIII: a Kuratowski theorem for general surfaces*, J. Combin. Theory B 48, 1990, 255-288; *Graph minors XVI: Wagner's conjecture*, erscheint in J. Combin. Theory B.

[RS95] Robertson, N., Seymour, P.D., *Graph minors XIII: the disjoint paths problem*, J. Combin. Theory B 63, 1995, 65-110.

[RSL77] Rosenkrantz, D.J., Stearns, R.E., Lewis, P.M., *An analysis of several heuristics for the traveling salesman problem,* SIAM J. Comput. 6, 1977, 563-581.

[RST93] Robertson, N., Seymour, P.D., Thomas, R., *Hadwiger's conjecture for K_6-free graphs*, Combin. 13, 1993, 279-361.

[RSST94] Robertson, N., Sanders, D.P., Seymour, P.D., Thomas, R., *The four-colour theorem*, Manuskript, August 1994; erhältlich über das WWW an der URL „http://www.math.ohio-state.edu/ dsanders".

[RSST96] Robertson, N., Sanders, D.P., Seymour, P.D., Thomas, R., *Efficiently four-coloring planar graphs*, Proc. 28th ACM SToC, 1996, 571-575.

[RT81] Reingold, E.M., Tarjan, R.E., *On a greedy heuristic for complete matching,* SIAM J. Comput. 10, 1981, 676-681.

[RT85] Roskind, J., Tarjan, R.E., *A note on finding minimum-cost edge-disjoint spanning trees,* Math. Oper. Res. 10, 1985, 701-708.

[RT87] Raghavan, P., Thompson, C.D., *Randomized rounding: a technique for provably good algorithms and algorithmic proofs,* Combin. 7, 1987, 365-374.

[RT88] Read, R.C., Tutte, W.T. *Chromatic polynomials,* in: [BW88], 15-42.

[RTL76] Rose, D.J., Tarjan, R.E., Lueker, G.S., *Algorithmic aspects of vertex elimination on graphs,* SIAM J. Comput. 5, 266-283.

[Ruh91] Ruhe, G., *Algorithmic aspects of flows in networks,* Kluwer Acad. Publ., Dordrecht, 1991.

[RW65] Van Rooij, A., Wilf, H., *The interchange graph of a finite graph,* Act. Math. Acad. Sci. Hung. 16, 1965, 263-269.

[RW96] Ravi, R., Williamson, D.P., *An approximation algorithm for minimum-cost vertex-connectivity problems,* Proc. 6th ACM-SIAM SoDA 1995, 332-341; erscheint in Algorithmica, 1996.

[RWW95] Ripphausen-Lipa, H., Wagner, D., Weihe, K., *Efficient algorithms for disjoint paths in planar graphs,* in: Combinatorial optimization, Hrsg. Cook, W., Lovász, L., Seymour, P.D., DIMACS series in discrete mathematics and theoretical computer science, Vol. 20, AMS, 1995.

[RY68] Ringel, G., Youngs, J.W.T., *Solution of the Heawood map coloring problem,* Proc. Nat. Acad. Sci. USA 60, 1968, 438-445.

[Rys63] Ryser, H.J., *Combinatorial mathematics*, Carus math. monograph no. 14, Math. Assoc. America, Wiley and Sons, 1963.

[Sav80] Savage, C., *Maximum matchings and trees,* Inform. Process. Lett. 10, 1980, 202-205.

[Sav82] Savage, C., *Depth-first search and the vertex cover problem,* Inform. Process. Lett. 14, 1982, 233-235.

[SB94] Szwarcfiter, J.L., Bornstein, C.F., *Clique graphs of chordal and path graphs,* SIAM J. Disc. Math. 7, 1994, 331-336.

[SBW90] Siran, J., Bodendiek, R., Wagner, K., *Planarität – ein einführender Überblick,* in [WB89], 9. Kapitel, Band II, 99-171.

[Sch83] Schrijver, A., *Min-max results in combinatorial optimization,* in: Mathematical programming – the state of the art, Hsrg. Bachem, A., Grötschel, M., Korte, B., Springer, Berlin, 1983, 439-500.

[Sch84] Schwarz, H.R., *Methode der finiten Elemente,* Teubner, Stuttgart, 1984.

[Sch85] Scheinerman, E.R., *Characterizing intersection classes of graphs,* Disc. Math. 55, 1985, 185-193.

[Sch86] Schrijver, A., *Theory of linear and integer programming,* Wiley & Sons, Chichester, 1986.

[Sch87a] Scheinerman, E., *The maximum interval number of graphs with given genus,* J. Graph Theory 11, 1987, 441-446.

[Sch87b] Scheffler, P., *Linear-time algorithms for NP-complete problems restricted to partial k-trees*, Report R-MATH-03/87, K.-Weierstraß-Institut für Math., Berlin, 1987.

[Sch89a] Scheffler, P., *Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme*, Dissertation, Report R-MATH-04/89, K.-Weierstraß-Institut für Math., Berlin, 1989.

[Sch89b] Schnyder, W., *Planar graphs and poset dimension*, Order 5, 1989, 323-343.

[Sch92] Schöning, U., *Theoretische Informatik kurz gefaßt,* BI Wissenschaftsverlag, Mannheim, 1992.

[Sch95] Schrijver, A., *Polyhedral combinatorics,* in: [GGL95], 1649-1704.

[SDK83] Syslo, M.M., Deo, N., Kowalik, J.S., *Discrete optimization algorithms,* Prentice-Hall, Englewood Cliffs NJ, 1983.

[Seb93] Sebö, A., *The connectivity of minimal imperfect graphs,* Report No. 93799, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, 1993.

[Sei74] Seinsche, D., *On a property of the class of n-colorable graphs,* J. Combin. Theory B 16, 1974, 191-193.

[Sey81] Seymour, P.D., *On odd cuts and plane multicommodity flows,* Proc. LMS 42, 1981, 178-192.

[Sey95] Seymour, P.D., *Nowhere-zero flows,* in: [GGL95], 289-299.

[SG76] Sahni, S., Gonzalez, T., *P-complete approximation problems,* J. ACM 23, 1976, 555-565.

[Sha56] Shannon, C.E., *The zero-error capacity of a noisy channel,* IRE Trans. Inf. Theo. IT 3, 1956, 3-15.

[She83] Shearer, J.B., *A note on the independence number of triangle-free graphs,* Disc. Math. 46, 1983, 83-87.

[SHF94] Schöneburg, E., Heinzmann, F., Feddersen, S., *Genetische Algorithmen und Evolutionsstrategien,* Addison-Wesley, Bonn, 1994.

[Shi81] Shiloach, Y., *Another look at the degree constrained subgraph problem,* Inform. Process. Lett. 12, 1981, 89-92.

[Shi88] Shibata, Y., *On the tree representation of chordal graphs,* J. Graph Theory 12, 1988, 421-428.

[Shm95] Shmoys, D.B., *Computing near-optimal solutions to combinatorial optimization problems,* in: Combinatorial optimization, Hrsg. Cook, W., Lovász, L., Seymour, P.D., DIMACS series in discrete mathematics and theoretical computer science, Vol. 20, AMS, 1995, 355-397.

[Shm96] Shmoys, D.B., *Cut problems and their application to divide-and-conquer,* in: Approximation algorithms for NP-hard problems, Hrsg. Hochbaum, D., PWS Publishers, angekündigt 1996; auch erhältlich über's WWW unter der URL „http://www.orie.cornell.edu/ shmoys".

[Sim75] Simon, J., *On some central problems in computational complexity,* PhD Thesis, TR 75-224, Dept.Comput.Sci., Cornell Univ., Ithaca NY, 1975.

[Sim83] Simonovits, M., *Extremal graph theory,* in: [BW83], 161-200.

[Sim84] SIMONOVITS, M., *Extremal graph problems, degenerate extremal problems, and supersaturated graphs*, in: Progress in graph theory, Hrsg. Bondy, J.A., Murty, U.S.R., Academic Press, Toronto, 1984, 419-437.

[Sim91] SIMON, K., *A new simple linear algorithm to recognize interval graphs,* in: Proc. computational geometry, Hrsg. Bieri, H., Noltemeier, H., Springer LNCS 553, 1991, 289-308.

[Sim92] SIMON, K., *Effiziente Algorithmen für perfekte Graphen,* Teubner, Stuttgart, 1992.

[Sin93] SINCLAIR, A.J., *Algorithms for random generation & counting*, Birkhäuser, Boston, 1993.

[Sip92] SIPSER, M., *The history and status of the* P *versus* NP *question*, Proc. 24th ACM SToC, 1992, 603-618.

[SK77] SAATY, T.L., KAINEN, P.C., *The four-color problem: assaults and conquest*, McGraw-Hill, New York, 1977.

[Ski90] SKIENA, S., *Implementing discrete mathematics: combinatorics and graph theory with Mathematica,* Addison-Wesley, Redwood City CA, 1990.

[Sla95] SLAVÍK, P., *Improved performance of the greedy algorithm for the minimum set cover and minimum partial cover problems,* Electron. Colloq. Comput. Compl. TR95-053, 1995, erhältlich über WWW an der URL „http://www.eccc.uni-trier.de/eccc/".

[Sli96] SLIVNIK, T., *A short proof of Galvin's theorem on the list-chromatic index of a bipartite mulitgraph,* Combin. Probab. Comput. 5, 1996, 91-94.

[Slo91] SLOMSON, A., *An introduction to combinatorics,* Chapman and Hall, London, 1991.

[Smi93] SMITH, J.R., *The design and analysis of parallel algorithms,* Oxford Univ. Press, 1993.

[Spe28] SPERNER, E., *Ein Satz über Untermengen einer endlichen Menge,* Math. Zeitschrift 27, 1928, 544-548.

[Spi92] SPINRAD, J.P., *P_4 trees and substitution decomposition,* Disc. Appl. Math. 39, 1992, 263-291.

[SR34] STEINITZ, E., RADEMACHER, H., *Vorlesungen über die Theorie der Polyeder,* Springer-Verlag, Berlin, 1934; Nachdruck 1976.

[SS77] SOLOVAY, R., STRASSEN, V., *A fast Monte-Carlo test for primality*, SIAM J. Comput. 6, 1977, 84-85; Erratum 7, 1978, 118.

[SS87] SHAMIR, E., SPENCER, J., *Sharp concentration of the chromatic number in random graphs $G_{n,p}$*, Combinatorica 7, 1987, 121-129.

[SS89] SACHS, H., STIEBITZ, M., *On constructive methods in the theory of colour-critical graphs,* Disc. Math. 74, 1989, 201-226.

[SS92] SANKARANARAYANA, R.S., STEWART, L.K., *Complexity results for well-covered graphs,* Networks 22, 1992, 247-262.

[SS95] SPINRAD, J., SRITHARAN, R., *Algorithms for weakly triangulated graphs,* Disc. Appl. Math. 59, 1995, 181-191.

[SSV95] SHAHROKHI, F., SZÉKELY, L.A., VRT'O, I., *Crossing numbers of graphs, lower bound techniques and algorithms: a survey,* in: Graph drawing - GD '94, Hrsg. Tamassia, R., Tollis, I.G., Springer LNCS 894, Berlin, 1995, 131-142.

[ST80] SEIFERT, H., THRELFALL, W., *A textbook of topology,* Academic Press, 1980.

[ST90] SATYANARAYANA, A., TUNG, L., *A characterization of partial 3-trees*, Networks 20, 1990, 883-898.

[Stan86] STANLEY, R.P., *Enumerative combinatorics,* Vol.1, Wadsworth & Brooks/Cole, Monterey CA, 1986.

[Ste22] STEINITZ, E., *Polyeder und Raumeinteilungen,* Enzykl. math. Wiss., Bd.3, 1922, 1-139.

[Ste51] STEIN, S.K., *Convex maps,* Proc. AMS 2, 1951, 464-466.

[Ste74] STEIN, S.K., *Two combinatorial covering theorems,* J. Combin. Theory A 16, 1974, 391-397.

[Ste93] Steinberg, R., *The state of the three color problem,* in: Quo vadis, graph theory?, Hrsg. Gimbel, J., Kennedy, J.W., Quintas, L.V., Ann. Disc. Math. 55, North-Holland, Amsterdam, 1993, 211-248.

[Sto73] Stockmeyer, L.J., *Planar 3-colorability is polynomial complete,* (ACM) SIGACT News 5, 1973, 19-25.

[Sto87] Stockmeyer, L.J., *Classifying the computational complexity of problems,* J. Symb. Logic 52, 1987, 1-43.

[Sto92] Stoer, M., *Design of survivable networks,* Springer-Verlag, Berlin, 1992.

[Su95] Su, X.-Y., *An algorithm for the decomposition of graphs into cliques,* J. Graph Theory 20, 1995, 195-202.

[Sud95] Sudan, M., *Efficient checking of polynomials and proofs, and the hardness of approximation problems,* Ph.D. Thesis, UC Berkeley, Springer, LNCS 1001, 1995.

[SV95] Saran, H., Vazirani, V.V., *Finding k cuts within twice the optimal,* SIAM J. Comput. 24, 1995, 101-108.

[SW68] Szekeres, G., Wilf, H.S., *An inequality for the chromatic number of a graph,* J. Combin. Theory 4, 1968, 1-3.

[SW78] Schwenk, A.J., Wilson, R.J., *On the eigenvalues of a graph,* in: [BW78], 307-336.

[SW89] Seese, D.G., Wessel, W., *Graphenminoren und Gitter – zu einigen Arbeiten von N. Robertson und P.D. Seymour,* 17. Kapitel in [WB89], Band III, 148-209.

[SW94] Stoer, M., Wagner, F., *A simple min cut algorithm,* in: Algorithms – ESA '94, Hrsg. Van Leeuwen, J., Springer LNCS 855, Berlin, 1994, 141-147.

[SX96] Stewart, L., Xu, B. *Graph families,* WWW-Seite „http://web.cs.ualberta.ca/~bing/GRAPH/index.html", Januar 1996.

[SY91] Schäffer, A.A., Yannakakis, M., *Simple local search problems that are hard to solve,* SIAM J. Comput. 20, 1991, 56-87.

[Sys89] Syslo, M.M., *Sequential coloring versus Welsh-Powell bound,* Disc. Math. 74, 1989, 241-243.

[Szp45] Szpilrajn-Marczewski, E., *Sur deux propriétés des classes d'ensembles,* Fund. Math. 33, 1945, 303-307.

[Tai80a] Tait, P.G., *Note on a theorem in the geometry of position*, Trans. Royal Soc. Edinb. 29, 1880, 657-660.

[Tai80b] Tait, P.G., *Remarks on the colouring of maps*, Proc. Royal Soc. Edinb. A 10, 1880, 501-503, 729.

[Tai84] Tait, P.G., *Listing's Topologie*, Phil. Mag. 17, 1884, 30-46.

[Tar72] Tarjan, R.E., *Depth-first search and linear graph algorithms*, SIAM J. Comput. 1, 1972, 146-160.

[Tar74] Tarjan, R.E., *Finding dominators in directed graphs*, SIAM J. Comput. 3, 1974, 62-89.

[Tar83] Tarjan, R.E., *Data structures and network algorithms*, CBMS-NSF Reg. Conf. Ser. Appl. Math. 44, SIAM, Philadelphia, 1983.

[Tar85] Tarjan, R.E., *Decomposition by clique separators,* Disc. Math. 55, 1985, 221-232.

[Tar95] Tarry, G., *Le problème des labyrinthes,* Nouv. Ann. Math. 14, 1895, 187.

[Tho77] Thomassen, C., *Straight line representations of infinite planar graphs*, J. LMS 16, 1977, 411-423.

[Tho80] Thomassen, C., *Planarity and duality of finite and infinite graphs*, J. Combin. Theory B 29, 1980, 244-271.

[Tho81] Thomassen, C., *Kuratowski's theorem*, J. Graph Theory 5, 1981, 225-241.

[Tho83] Thomassen, C., *A theorem on paths in planar graphs,* J. Graph Theory 7, 1983, 169-176.

[Tho84a] Thomassen, C., *Infinite graphs*, in: [BW84], 129-160.

[Tho84b] Thomassen, C., *A refinement of Kuratowski's theorem*, J. Combin. Theory B 37, 1984, 245-253.

[Tho84c] Thomason, A.G., *An extremal function for contractions of graphs,* Math. Proc. Camb. Phil. Soc. 95, 1984, 261-265.

[Tho89] Thomassen, C., *The graph genus problem is* NP*-complete,* J. Alg. 10, 1989, 568-576.

[Tho92] Thomassen, C., *The Jordan-Schönflies curve theorem and the classification of surfaces,* Amer. Math. Monthly 99, 1992, 116-130.

[Tho94a] Thomassen, C., *Embeddings of graphs,* Disc. Math. 124, 1994, 217-228.

[Tho94b] Thomassen, C., *Every planar graph is 5-choosable,* J. Comb. Theory B 62, 1994, 180f.

[Tho94c] Thomassen, C., *Grötzsch's 3-color theorem and its counterparts for the torus and the projective plane,* J. Comb. Theory B 62, 1994, 268-279.

[Tho95] Thomassen, C., *Embeddings and minors,* in: [GGL95], 301-349.

[TIAS77] Tsukiyama, S., Ide, M., Ariyoshi, H., Shirakawa, I., *A new algorithm for generating all maximal independent sets,* SIAM J. Comput. 6, 1977, 505-517.

[Tin80] Tinhofer, G., *Zufallsgraphen,* Hanser, München, 1980.

[Tin84] Tinhofer, G., *A probabilistic analysis of some greedy cardinality matching algorithms,* in: Stochastics and optimization, Hrsg. Archetti, F., Maffioli, F., Ann. Oper. Res. 1, 1984, 239-254.

[TM80] Takahashi, H., Matsuyama, A., *An approximate solution for the Steiner problem in graphs,* Math. Jap. 24, 1980, 573-577.

[Tof95] Toft, B., *Colouring, stable sets, and perfect graphs,* in: [GGL95], 233-288.

[Tof96] Toft, B., *A survey of Hadwiger's conjecture,* Institut for Matematik og Datalogi, Odense Universitet, Preprint No. 1, 1996.

[Toi73] Toida, S., *Properties of an Euler graph,* J. Franklin Inst. 295, 1973, 343-345.

[Tov84] Tovey, C.A., *A simplified NP-complete satisfiability problem,* Disc. Appl. Math. 8, 1984, 85-89.

[Tro92] Trotter, W.T., *Combinatorics and partially ordered sets: dimension theory,* Johns Hopkins University Press, Baltimore, 1992.

[TS92] Thulasiraman, K., Swamy, M.N.S., *Graphs: theory and algorithms,* Wiley & Sons, New York, 1992.

[TT77] Tarjan, R.E., Trojanowski, A.E., *Finding a maximum independent set,* SIAM J. Comput. 6, 1977, 537-546.

[Tuc73] Tucker, A., *The strong perfect graph conjecture for planar graphs,* Canad. J. Math. 25, 1973, 103-114. .

[Tuc84] Tucker, A., *The validity of the perfect graph conjecture for K_4-free graphs,* in [BC84], 149-158.

[Tur36] Turing, A., *On computable numbers, with an application to the Entscheidungsproblem*, Proc. Lond. Math. Soc., Ser.2, 1936, 42, 230-265, Corrigendum 43, 544-546.

[Tur41] Turán, P., *Egy gráfelméleti szélsőért'ekfeladatról* [On an extremal problem in graph theory], Mat. Fiz. Lapok 48, 1941, 436-452.

[Tur54] Turán, P., *On the theory of graphs,* Colloq. Math. 3, 1954, 19-30.

[Tur88] Turner, J.S., *Almost all k-colorable graphs are easy to color,* J. Alg. 9, 1988, 63-82.

[Tur96] Turau, V., *Algorithmische Graphentheorie,* Addison Wesley, Bonn, 1996.

[Tut47] Tutte, W.T., *The factorization of linear graphs,* J. LMS 22, 1947, 107-111.

[Tut52] Tutte, W.T., *The factors of graphs,* Canad. J. Math. 4, 1952, 314-328.

[Tut54a] Tutte, W.T., *A contribution to the theory of chromatic polynomials*, Canad. J. Math. 6, 1954, 80-91.

[Tut54b] TUTTE, W.T., *A short proof of the factor theorem for finite graphs*, Canad. J. Math. 6, 1954, 347-352.

[Tut56] TUTTE, W.T., *A theorem on planar graphs,* Trans. AMS 82, 1956, 99-116.

[Tut60] TUTTE, W.T., *Convex representations of graphs,* Proc. LMS 10, 1960, 304-320.

[Tut61] TUTTE, W.T., *On the problem of decomposing a graph into n connected factors,* J. LMS 36, 1961, 221-230.

[Tut63] TUTTE, W.T., *How to draw a graph,* Proc. LMS 13, 1963, 743-768.

[Tuz92] TUZA, Z., *Graph coloring in linear time,* J. Combin. Theory B 55, 1992, 236-243.

[Tve67] TVERBERG, H., *On Dilworth's decomposition theorem for partially ordered sets,* J. Combin. Theory 3, 1967, 305-306.

[TY84] TARJAN, R.E., YANNAKAKIS, M., *Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs,* SIAM J. Comput. 13, 1984, 566-579; Addendum 14, 1985, 254-255.

[Usp94] USPENSKY, V.A., *Gödel's incompleteness theorem,* Theoret. Comput. Sci. 130, 1994, 239-319.

[Vag85] VAGO, I., *Graph theory: applications to the calculation of electrical networks,* Elsevier, Amsterdam, 1985.

[Val79] VALIANT, L.G., *The complexity of computing the permanent*, Theoret. Comput. Sci. 8, 1979, 189-201.

[Vaz94] VAZIRANI, V.V., *A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ general graph maximum matching algorithm*, Combin. 14, 1994, 71-109.

[VB95] VANDENBERGHE, L., BOYD, S., *Semidefinite programming,* eingereicht bei SIAM Rev. 1995.

[Veb12] VEBLEN, O., *An application of modular equations in analysis situs*, Ann. Math. 14, 1912-13, 86-94.

[Veg83] DE LA VEGA, W.F., *On the maximum density of graphs which have no subcontraction to K^s,* Disc. Math. 46, 1983, 109-110.

[Ven87] VENKATESAN, S.M., *Improved constants for some separator theorems*, J. Alg. 8, 1987, 572-578.

[Vin93] VINCE, A., *The classification of closed surfaces using colored graphs,* Graphs Combin. 9, 1993, 75-84.

[Vit81] VITÁNYI, P.M.B., *How well can a graph be n-colored?,* Disc. Math. 34, 1981, 69-80.

[Viz63] VIZING, V.G., *The Cartesian product of graphs,* Vyč. Sis. 9, 1963, 30-43.

[Viz64] VIZING, V.G., *On an estimate of the chromatic class of a p-graph,* (in Russisch), Diskret. Anal. 3, 1964, 25-30.

[Viz65a] VIZING, V.G., *Critical graphs with a given chromatic class,* (in Russisch), Diskret. Anal. 5, 1965, 9-17.

[Viz65b] VIZING, V.G., *The chromatic class of a multigraph,* (in Russisch), Cybernetics 3, 1965, 32-41.

[Viz76] VIZING, V.G., *Coloring the vertices of a graph in prescribed colors,* (in Russisch), Metody Diskret. Analiz. 29, 1976, 3-10.

[Voi93] VOIGT, M., *List colourings of planar graphs,* Disc. Math. 120, 1993, 215-219.

[Vol68] VOLLMERHAUS, H., *Über die Einbettung von Graphen in zweidimensionale orientierbare Mannigfaltigkeiten kleinsten Geschlechts,* in: Beiträge zur Graphentheorie, Hrsg. Sachs, H., Voss, H., Walter, H.; Teubner, Leipzig, 1968, 163-168.

[Vol96] VOLKMANN, L., *Fundamente der Graphentheorie,* Springer, Wien, 1996.

[Vos91] VOSS, H.-J., *Cycles and bridges in graphs,* Kluwer Acad. Publ., Dordrecht / Deutscher Verlag der Wissenschaften, Berlin, 1991.

[Wag36] WAGNER, K., *Bemerkungen zum Vierfarbenproblem,* Jahresber. DMV 46, 1936, 26-32.

[Wag37] WAGNER, K., *Über eine Eigenschaft der ebenen Komplexe,* Math. Ann. 114, 1937, 570-590.

[Wag60] Wagner, K., *Bemerkungen zu Hadwigers Vermutung,* Math. Ann. 141, 1960, 433-451.

[Wag64] Wagner, K., *Beweis einer Abschwächung der Hadwiger Vermutung,* Math. Ann. 153, 1964, 139-141.

[Wag70] Wagner, K., *Graphentheorie,* Bibliographisches Institut, Mannheim, 1970.

[Wal78] Walter, J.R., *Representation of chordal graphs as subtrees of a tree,* J. Graph Theory 2, 1978, 265-267.

[Wal84] Walther, H., *Ten applications of graph theory,* Reidel, Dordrecht, 1984 (überarbeitete Übersetzung von: *Anwendungen der Graphentheorie*, Vieweg & Sohn, Braunschweig, 1979, in Lizenz des VEB Deutscher Verlag der Wissenschaften, Berlin, 1978).

[WB78] White, A.T., Beineke, L.W., *Topological graph theory,* in: [BW78], 15-49.

[WB79] Wilson, R.J., Beineke, L.W., (Hrsg.), *Applications of graph theory,* Academic Press, London, 1979.

[WB89] Wagner, K., Bodendiek, R., *Graphentheorie I/II/III,* Bibliographisches Institut, Mannheim, 1989/1990/1992.

[WC83] Wald, J.A., Colbourn, C.J., *Steiner trees, partial 2-trees, and minimum IFI networks,* Networks 13, 1983, 159-167.

[Weg93] Wegener, I., *Theoretische Informatik – eine algorithmenorientierte Einführung,* Teubner, Stuttgart, 1993.

[Wei63] Weinstein, J.M., *On the number of disjoint edges in a graph,* Canad. J. Math. 15, 1963, 106-111.

[Wei81] Wei, V.K., *A lower bound on the stability number of a simple graph,* Bell Lab. Tech. Memor. 81-11217-9, 1981.

[Wei95] Weiss, M.A., *Data structures and algorithm analysis,* Benjamin/Cummings, Redwood City CA, 2. Auflage, 1995.

[Wel76] Welsh, D.J.A., *Matroid theory*, Academic Press, London, 1976.

[Wel93] Welsh, D.J.A., *Complexity: knots, colourings, and counting*, LMS Lecture Notes Series Vol. 186, Cambridge Univ. Press, 1993.

[Wer04] Wernicke, P., *Über den kartographischen Vierfarbensatz,* Math. Ann. 58, 1904, 413-426.

[Wer70] De Werra, D., *On some combinatorial problems arising in scheduling,* Canad. Oper. Res. Soc. J. 8, 1970, 165-175.

[Wer90] De Werra, D., *Heuristics for graph coloring*, in: Computational graph theory, Hrsg. Tinhofer, G., Mayr, E., Noltemeier, H., Syslo, M.M., Computing Suppl. 7, Springer, Wien, 1990, 191-208.

[Wes95] West, D.B., *Introduction to graph theory,* Prentice Hall, Upper Saddle River NJ, 1995.

[WGMV95] Williamson, D.P., Goemans, M.X., Mihail, M., Vazirani, V.V., *A primal-dual approximation algorithm for generalized Steiner network problems,* Combinatorica 15, 1995, 435-454.

[Whi31] Whitney, H., *A theorem on graphs,* Ann. Math. 32, 1931, 378-390.

[Whi32a] Whitney, H., *Congruent graphs and the connectivity of graphs,* Amer. J. Math. 54, 1932, 150-168.

[Whi32b] Whitney, H., *Non-separable and planar graphs,* Trans. AMS 34, 1932, 339-362.

[Whi33] Whitney, H., *Planar graphs,* Fund. Math. 21, 1933, 73-84.

[Whi84] White, A.T., *Graphs, groups, and surfaces*, North-Holland, Amsterdam, 2. Auflage, 1984.

[Whi87] White, N., *Combinatorial geometries,* Cambridge Univ. Press, 1987.

[Wig82] Wigderson, A., *The complexity of the hamiltonian circuit problem for maximal planar graphs,* EECS Dept. Report 298, Princeton Univ., 1982.

[Wil67] Wilf, H.S., *The eigenvalues of a graph and its chromatic number,* J. LMS 42, 1967, 330-332.

[Wil82] Wilson, R.J., *Applications of combinatorics,* Shiva Publ., Cheshire, 1982.

[Wil84] WILF, H.S., *Backtrack: an $\mathcal{O}(1)$ average time algorithm for the graph coloring problem,* Inform. Process. Lett. 18, 1984, 119-122.

[Wil86] WILF, H.S., *Algorithms and complexity,* Prentice-Hall, 1986; erhältlich über anonymous ftp als Datei „pub/wilf/AlgComp.ps.Z“ von „ftp.cis.upenn.edu“.

[Wil90] WILF, H.S., *Generating functionology,* Academic Press, 1990.

[Wil93] WILLIAMSON, D.P., *On the design of approximation algorithms for a class of graph problems,* Ph.D.-thesis, MIT, 1993; erhältlich über WWW an der URL „http://www.research.ibm.com/people/w/williamson/“.

[Wim87] WIMER, T.V., *Linear algorithms on k-terminal graphs,* Ph.D. thesis URI-030, Clemson University, 1987.

[Win87] WINTER, P., *Steiner problem in networks: a survey,* Networks 17, 1987, 129-167.

[Wir83] WIRTH, N., *Algorithmen und Datenstrukturen,* (PASCAL-Version), Teubner, Stuttgart, 1975, 1983[3].

[WKC88] WIMER, S., KOREN, J., CEDERBAUM, I., *Floor plans, planar graphs and layouts,* IEEE Trans. Circuits and Systems CAS-35, 1988, 267-278.

[WN87] WATANABE, T., NAKAMURA, A., *Edge-connectivity augmentation problems,* J. Comput. System Sci. 35, 1987, 96-144.

[WN90] WILSON, R., NELSON, R., *Graph colorings,* Longman, Essex / Wiley & Sons, New York, 1990.

[Woo92] WOODALL, D.R., *A short proof of a theorem of Dirac's about Hadwiger's conjecture,* J. Graph Theory 16, 1992, 79-80.

[WP67] WELSH, D.J.A., POWELL, M.B., *An upper bound for the chromatic number of a graph and its application to timetabling problems,* Comput. J. 10, 1967, 85-87.

[WT92] WATANABE, O., TANG, S., *On polynomial-time Turing and many-one completeness in* PSPACE, Theoret. Comput. Sci. 97, 1992, 199-215.

[WV74] WALTHER, H., VOSS, H.-J., *Über Kreise in Graphen,* VEB Deutscher Verlag der Wissenschaften, Berlin, 1974.

[WW78] WOODALL, D.R., WILSON, R.J., *The Appel-Haken proof of the four-color theorem,* in: [BW78], 83-101.

[WW86] WAGNER, K., WECHSUNG, G., *Computational complexity,* Reidel, Dordrecht, 1986.

[WW90] WILSON, R.J., WATKINS, J.J., *Graphs - an introductory approach,* Wiley & Sons, New York, 1990.

[Xu89] XU, W., *An improved algorithm for planarity testing based on Wu-Liu's criterion,* in : Graph theory and its applications: east and west, Hrsg. Capobianco, M.F., Guan, M., Hsu, D.F., Tian, F., New York Acad. Sci., 1989, 641-652.

[Yan81a] YANNAKAKIS, M., *Edge-deletion problems* SIAM J. Comput. 10, 1981, 297-309.

[Yan81b] YANNAKAKIS, M., *Computing the minimum fill-in is* NP*-complete,* SIAM J. Alg. Disc. Meth. 2, 1981, 77-79.

[Yan86] YANNAKAKIS, M., *Four pages are necessary and sufficient for planar graphs,* Proc. 18th ACM SToC, 1986, 104-108.

[Yan89] YANNAKAKIS, M., *Embedding planar graphs in four pages,* J. Comput. System Sci. 38, 1989, 36-67.

[Yan94] YANNAKAKIS, M., *On the approximation of maximum satisfiability,* J. Alg. 17, 1994, 475-502.

[Yan96] YANPEI, L., *Embeddability in graphs,* Kluwer Acad. Publ., Dordrecht, 1996.

[Yap86] YAP, H.P., *Some topics in graph theory,* LMS Lecture Notes Series Vol. 108, Cambridge Univ. Press, 1986.

[Yap96] YAP, H.P., *Total colourings of graphs,* Springer LNM 1623, Berlin, 1996.

[YG80] YANNAKAKIS, M., GAVRIL, F., *Edge dominating sets in graphs,* SIAM J. Appl. Math. 38, 1980, 364-372.

[You63] YOUNGS, J.W.T., *Minimal imbeddings and the genus of a graph*, J. Math. Mech. 12, 1963, 303-315.

[You71] YOUNG, H.P., *A quick proof of Wagner's equivalence theorem,* J. LMS 3, 1971, 661-664.

[Zar54] ZARANKIEWICZ, K., *On a problem of P. Turán concerning graphs,* Fund. Math. 41, 1954, 137-145.

[Zar91] ZAREMBA, L.S., *Perfect graphs and norms,* Math. Progr. 51, 1991, 269-272.

[Zel93a] ZELIKOVSKY, A.Z., *An 11/6-approximation algorithm for the network Steiner problem,* Algorithmica 9, 1993, 463-470.

[Zel93b] ZELIKOVSKY, A.Z., *A faster approximation algorithm for the Steiner tree problem in graphs,* Inform. Process. Lett. 46, 1993, 79-83.

[Zem96] ZEMANIAN, A., *Transfiniteness – for graphs, electrical networks, and random walks,* Birkhäuser, Basel, 1996.

[Zie95] ZIEGLER, G.M., *Lectures on polytopes,* Springer, New York, 1995.

[Zwi95] ZWICK, U., *The smallest networks on which the Ford-Fulkerson maximum flow procedure may fail to terminate,* Theoret. Comput. Sci. 148, 1995, 165-170.

[Zyk49] ZYKOV, A.A., *On some properties of linear complexes,* (auf Russisch), Math. Sbornik 24, 1949, 163-188; AMS Translations No. 79, 1952.

[Zyk90] ZYKOV, A.A., *Fundamentals of graph theory,* Übersetzung der russischen Ausgabe von 1987, Hrsg. Boron, L., u.a., BCS Associates, Moscow, Idaho USA, 1990.