Combinatorial Optimization

Stephan Held

Research Institute for Discrete Mathematics
University of Bonn
Lennéstr. 2
53113 Bonn

held@or.uni-bonn.de

February 5, 2015

Announcements

Required Qualifications

- **graph algorithms**
- **linear optimization**
- **programming skills in C/C++!**

Examination

Oral exams will be held in the first two weeks of the spring break. Successful participation in the problem classes is a requirement for examination.

Problem classes

The participation at the problem classes is obligatory. For a successful participation at the problem classes a regular participation, as well as at least 50% of the points in the theorical exercises and 50% of the points of the programming exercises are required.

Literature

- B. Korte, J. Vygen: Combinatorial Optimization: Theory and Algorithms. Springer, Fifth edition 2012.
  (most parts are based on this book)
- A. Schrijver: Combinatorial Optimization: Polyhedra and Efficiency. Springer, 2003.
  (comprehensive reference of combinatorial optimization)
- W. Cook, W. Cunningham, W. Pullyblank, A. Schrijver: Combinatorial Optimization, John Wiley & Sons, 1997.
  (very descriptive but slightly less comprehensive)
- A. Frank: Connections in Combinatorial Optimization. Oxford University Press, 2011.
- J. Lee: A First Course in Combinatorial Optimization, Cambridge Texts in Applied Mathematics, 2004.
- C.H. Papadimitriou and K. Steiglitz: Combinatorial Optimization: Algorithms and Complexity, Prentince Hall, 1982, Dover edition form 1998).
  (classical and affordable text book compromising the state-of-the-art in 1982)
- E. Lawler: Combinatorial Optimization: Networks and Matroids, Holt, Rinehart and Winston 1976, Dover edition form 2001).
  (the other classical and affordable text book compromising the state-of-the-art in 1976)

If need to learn linear programming basics in a self-study you will find a chapter in most of the books above. Apart from that, I recommend:

- B. Gärtner, J. Matousek: Understanding and Using Linear Programming, Springer, Berlin, 2006.

This list of definitions and theorems is continuously updated here:
http://www.or.uni-bonn.de/ held/combo/1213/lecturenotes/CombOpt.pdf. For download you need following access data : user: *combo* password: *COMBO1213* .

Notice

These lecture notes are a plain collection of definitions and theorems presented in the lecture. Details and proofs can be found in the books listed above.

Contents

1 Matching

1.1 Matchings and Alternating Paths

Definition 1.1 (Matchings and Edge Covers)**.**

1. *A* **matching** *M of a graph $G = (V, E)$ is a set of pairwise disjoint edges, i.e. no two edges in M have a common endpoint. By $\nu(G)$ we denote the maximum cardinality of a matching of G.*

2. *An* **edge cover** *C of a graph $G = (V, E)$ is a subset of E s.t.*

$$V = \bigcup_{\{v,w\} \in C} \{v, w\}.$$

   By $\zeta(G)$ we denote the minimum cardinality of an edge cover of G.

3. *A matching M is called* **perfect** *if it is an edge-cover. A perfect matching is also called* **1-factor**.

4. *Vertices $v \in V$ with $v \in e \in M$ are called* **M-covered**.

5. *Vertices not covered by M are called* **M-exposed**.

Definition 1.2 (Stable Sets and Vertex Covers)**.**

1. *A* **stable set** *S of a graph $G = (V, E)$ is a set of pairwise non-adjacent vertices, i.e. no edge connects vertices in S. By $\alpha(G)$ we denote the maximum cardinality of a stable set of G.*

2. *An* **vertex cover** *C of a graph $G = (V, E)$ is a subset of V s.t.*

$$E = \bigcup_{\{v,w\} \in E : v \in C} \{v, w\}.$$

   By $\tau(G)$ we denote the minimum cardinality of a vertex cover of G.

Lemma 1.3 (Equalities for α, τ, ν, and ζ)**.**

1. *$\alpha(G) + \tau(G) = |V(G)|$ for any graph G.*

2. *$\nu(G) + \zeta(G) = |V(G)|$ for any graph G with no isolated vertices.*

3. $\zeta(G) = \alpha(G)$ *for any bipartite graph* G *with no isolated vertices.*

Proof. Homework exercise. □

In this lecture, we are interested in the following optimization problems.

CARDINALITY MATCHING PROBLEM	
Input:	an undirected graph G.
Task:	find a maximum cardinality matching M in G.

MAXIMUM WEIGHT MATCHING PROBLEM	
Input:	an undirected graph G and weights $c : E(G) \to \mathbb{R}$.
Task:	find a maximum weight matching M in G.

MINIMUM WEIGHT PERFECT MATCHING PROBLEM	
Input:	an undirected graph G and weights $c : E(G) \to \mathbb{R}$.
Task:	find a minimum weight perfect matching M in G or decide that G has no perfect matching.

Lemma 1.4. *The* MAXIMUM WEIGHT MATCHING PROBLEM *and the* MINIMUM WEIGHT PERFECT MATCHING PROBLEM *are equivalent, i.e. there exist linear size transformations between them.*

Proof. □

Definition 1.5. *(Alternating Path) Given a matching* M *of* G*, a path* P *is* M**-alternating** *if its edges are alternatingly in and not in* M*. If in addition both endpoints in* P *are* M*-exposed,* P *is an* M**-augmenting** *path.*

Lemma 1.6. *Given a matching* M *of* G *and an (inclusionwise) maximal alternating path* P*, then the symmetric difference*

$$M \triangle P := M \setminus P \cup P \setminus M \tag{1.1}$$

is a matching. If P *is* M*-augmenting then* $|M \triangle P| = |M| + 1$*.*

Proof. simple exercise. □

A central observation for solving maximum matching problems is the following characterization.

Theorem 1.7. *(Augmenting Path Theorem of Matchings, Petersen [1891], Berge [1957]) A matching* M *in* G *is maximum* $\Leftrightarrow$ *there is no* M*-augmenting path.*

1.2 Bipartite Matching

In this section we recapitulate known facts about bipartite matching. r

Theorem 1.8. *(König [1931]). If G is bipartite, then $\nu(G) = \tau(G)$.*

Theorem 1.9. *(Hall [1935]) Let $G = (A \dot\cup B, E)$ be a bipartite graph. Then G has a matching of A into B if and only if it satisfies the* **Hall condition**

$$|\Gamma(X)| \geq |X| \quad \textit{for all } X \subseteq A \tag{1.2}$$

Proof. It can be directly deduced from Theorem 1.8, or prooved from scratch by induction. □

Corollary 1.10. *(The Marriage Theorem, Frobenius [1917]). A bipartite graph $G = (A \dot\cup B, E)$ has a perfect matching if and only if $|A| = |B|$ and (1.2) holds.*

Theorem 1.11. *The* CARDINALITY MATCHING PROBLEM *for bipartite graphs can be solved in $O(nm)$ time, where $n = |V|$ and $m = |E|$.*

Proof. Construct a directed graph form G, with a new source s connected to each $a \in A$ and new sink t connected to each $b \in B$, directed from s to t and with unit capacities. Then apply Ford-Folkerson's maximum flow algorithm. □

Definition 1.12. *The* MINIMUM WEIGHT PERFECT MATCHING PROBLEM *for bipartite graphs is called* ASSIGNMENT PROBLEM.

Theorem 1.13. *The* ASSIGNMENT PROBLEM *can be solved in $O(nm + n^2 \log n)$ time, where $n = |V|$ and $m = |E|$.*

Proof. Construct a directed graph G' form G as in the proof of Theorem 1.11 and set the edge costs of artificial edges to zero. A minimum-cost integral s-t-flow of value n in G' corresponds to a minimum weight perfect matching in G and vice versa. A minimum-cost flow can be found by the SUCCESSIVE SHORTEST PATH ALGORITHM in $O(nm + n^2 \log n)$. As all edge capacities are one the flow is on each edge is either zero or one. □

1.3 The Tutte Matrix and Randomized Matching

Definition 1.14. *(Tutte Matrix). Let G be a simple undirected graph and let G' be an arbitrary orientation of G. For a vector $x = (x_e)_{e \in E(G)}$ of variables, we define the* **Tutte matrix** *by*

$$T_G(x) = (t^x_{vw})_{v,w \in V(G)}$$
$$t^x_{vw} := \begin{cases} x_{\{v,w\}} & \textit{if } (v,w) \in E(G') \\ -x_{\{v,w\}} & \textit{if } (w,v) \in E(G') \\ 0 & \textit{otherwise} \end{cases}. \tag{1.3}$$

Remark 1.15. *The Tutte matrix is skew-symmetric, i.e.* $T_G(x) = -T_G(x)^\mathsf{T}$ *and its rank is independent of the chosen orientation. Its determinant* $\det T_G(x)$ *is a polynomial in* x*.*

Theorem 1.16. *(Tutte [1947]). A simple graph* G *has a perfect matching if and only if* $\det T_G(x) \not\equiv 0$*.*

Remark 1.17. *This means that if we pick a vector* $x' \in [0,1]^{E(G)}$ *of independent and uniformly distributed variables, then* $\det T_G(x') = 0$ *almost surely if and only if* G *does not have a perfect matching, because if* G *has a perfect matching, the set of roots of* T_G *has measure* 0*.*

This idea can be extended to guess the size of a maximum matching:

Theorem 1.18. *(Lovász [1979]) Let* G *be a simple graph and* $x = (x_e)_{e \in E(G)}$ *a random vector, where the coordinates are independent and uniformly distributed in* $[0,1]$*. Then almost surely*

$$\operatorname{rank}(T_G(x)) = 2\nu(G).$$

1.4 Tutte's Matching Theorem

The main goal of this section is to generalize the "good characterizations" of Theorems 1.8 and 1.9 from bipartite to arbitrary graphs.

First we characterize the existence of perfect matchings. Assume we have a set of points $X \subseteq V(G)$ such that the number $q_G(X)$ of odd connected components in $G - X$ exceeds $|X|$. Then G cannot have a perfect matching. Tutte's Theorem states that this necessary condition for perfect matchings is also sufficient!

Definition 1.19. *A graph* G *satisfies the* **Tutte Condition** *if*

$$q_G(X) \leq |X| \quad \textit{for all } X \subseteq V(G). \tag{1.4}$$

A nonempty vertex set $X \subseteq V(G)$ *is called a* **barrier** *if* $q_G(X) = |X|$*.*

Proposition 1.20. *For any graph* G *and any* $X \subseteq V(G)$ *we have*

$$q_G(X) - |X| = |V(G)| \pmod 2.$$

Definition 1.21. *A graph G is called* **factor-critical** *if $G - v$ has a perfect matching for each $v \in V(G)$. A matching is called* **near-perfect** *if it covers all vertices but one.*

Theorem 1.22. *(Tutte's Theorem [1947]). A graph G has a perfect matching if and only if it satisfies the Tutte condition (1.4).*

Tutte's theorem gives us a good characterization: "yes" (a perfect matching M) and "no" (a vertex set X violating the Tutte condition) certificates for the existence of perfect matchings.

Theorem 1.23. *(The Berge Formula, Berge [1958]).*

$$2\nu(G) + \max_{X \subseteq V(G)} (q_G(X) - |X|) = |V(G)|. \tag{1.5}$$

1.5 Ear-Decompositions of Factor-Critical Graphs

Definition 1.24. *Let G be a (undirected or directed) graph. An* **ear-decomposition** *if G is a sequence $r, P_1, \dots, P_k$ with $G = (\{r\}, \emptyset) + P_1 + \dots, P_k$, such that each P_i is either a path where exactly the endpoints belong to $\{r\} \cup V(P_1) \cup \dots V(P_{i-1})$, or a circuit where exactly one of its vertices belongs to $\{r\} \cup V(P_1) \cup \dots V(P_{i-1})$ ($i \in \{1, \dots, k\}$).*

$P_1, \dots, P_k$ are called **ears**. *If $k \geq 1$, P_1 is a circuit of length at least three, and $P_2, \dots, P_k$ are paths, then the ear-decomposition is called* **proper**.

Definition 1.25. *Let $k \geq 2$. A graph with more than k vertices that remains connected after deleting any $k-1$ vertices is called* **k-connected**.

A graph with at least 2 vertices that remains connected after deleting any $k-1$ edges is called **k-edge-connected**.

2-connected graphs are characterized by proper ear-decompositions:

Theorem 1.26. *(Whitney [1932]) An undirected graph is 2-connected if and only if it has a proper ear-decomposition.*

Definition 1.27. *An ear-decomposition is called* **odd** *if every ear has odd length.*

Theorem 1.28. *(Lovász [1972]) A graph is factor-critical if and only if it has an odd ear-decomposition. Furthermore, the initial vertex of the ear-decomposition can be chosen arbitrarily.*

Definition 1.29. *Given a factor-critical graph G and a near-perfect matching M, am* **M-alternating ear-decomposition** *of G is an odd ear-decomposition such that each ear is an M-alternating path or a circuit C with $|E(C) \cap M| + 1 = |E(C) \setminus M|$.*

Corollary 1.30. *For any factor-critical graph G and any near-perfect matching M in G there exists an M-alternating ear-decomposition.*

Definition 1.31. *Let G be a factor-critical graph and M a near-perfect matching in G. Let $r, P_1, \dots, P_k$ be an M-alternating ear-decomposition of G. Two functions $\mu, \varphi : V(G) \to V(G)$ two functions are* **associated with the ear-decomposition** *$r, P_1, \dots, P_k$ if*

- $\{x, y\} \in M \Rightarrow \mu(x) = y$ *(μ points to matched neighbor in ear-decomposition),*
- $\{x, y\} \in E(P_i) \setminus M$ *and* $x \notin \{r\} \cup V(P_1) \cup \dots V(P_{i-1}) \Rightarrow \varphi(x) = y$ *(φ points to unmatched neighbor in ear-decomposition),*
- $\mu(r) = \varphi(r) = r$.

Algorithm 1 Ear-Decomposition Algorithm

Instance: A factor-critical graph G, functions μ, φ associated with an M-alternating ear-decomposition.
Output: An M-alternating ear-decomposition $r, P_1, \dots, P_k$.
see blackboard notes

Proposition 1.32. *Let G be a factor-critical graph and μ, φ functions associated with an M-alternating ear-decomposition. Then this ear-decomposition is unique up to the order of the ears. The Ear-Decomposition Algorithm correctly determines an explicit list of these ears; it runs in linear time.*

Lemma 1.33. *Let G be a factor critical graph and μ, φ two functions associated with an M-alternating ear-decomposition. Let r be the vertex exposed by M. Then the maximal path given by an initial subsequence of*

$$x, \mu(x), \varphi(\mu(x)), \mu(\varphi(\mu(x))), \varphi(\mu(\varphi(\mu(x)))), \dots$$

defines an M-alternating x-r-path of even length for all $x \in V(G)$.

The reverse is not true.

1.6 Edmond's matching algorithm

Definition 1.34. *Let G be a graph and M a matching in G. A* **blossom** *in G w.r.t. M is a factor-critical subgraph B of G with $|M \cap E(B)| = \frac{|V(B)|-1}{2}$. The vertex r of B exposed by $M \cap E(B)$ is called the* **base** *of B.*

Definition 1.35. *Let G be a graph, M a matching in G, B a blossom of G w.r.t. M, and Q be an M-alternating v-r-path Q of even length from a vertex v exposed by M to the base r of B, where $E(Q) \cap E(B) = \emptyset$. Then $Q + B$ is called an* M**-flower**.

It turns out that we can simply shrink certain blossoms in our search for maximum matchings.

Lemma 1.36. *Let G be a graph, M a matching in G. Suppose there is an M-flower $B+Q$. Let G' and M' result from G and M by shrinking $V(B)$ to a single vertex. Then M is a maximum matching in G if and only if M' is a maximum matching in G'.*

Remark 1.37. *If there is no such path the statement of Lemma 1.36 does not hold.*

Lemma 1.38. *Let G be a graph, M a matching in G and $X \subseteq V(G)$ the set of M-exposed vertices. A shortest M-alternating X-X-walk of positive length can be found in $O(m)$.*

Theorem 1.39. *Let $P = \{v_0, v_1, \dots, v_t\}$ be a shortest M-alternating X-X-walk. Then either P is an M-alternating path or $(v_0, v_1, \dots, v_j)$ is an M-flower for some $j \leq t$.*

Algorithm 2 Edmond's Augmenting Path Finding

Instance: A graph G with a matching M
Output: An M-augmenting path if it exists

```
X := of M-exposed vertices.
if ∃M-alternating X-X-walk of positive length then
    Let P = (v_0, v_1, ..., v_t) be a shortest M-alternating X-X-walk.
    if P is a path then return P
    else
        Choose j s.t. ((v_0, v_1, ..., v_j) is an M-flower with blossom B.
        Apply this algorithm recursively to G/B with M/B
        Expand an M/B-augmenting path in G/B to an M-augmenting path in G.
    end if
else
    There is no M-augmenting path
end if
```

Theorem 1.40. *Given a graph $G = (V, E)$, a maximum-size matching can be found in time $O(n^2 m)$, where $n = |V|$ and $m = |E|$.*

1.6.1 Growing forests — an $O(n^3)$ Variant

In order to speed up the matching algorithm, we need to improve the running time for finding augmenting paths and blossom contraction.

Definition 1.41. *Given a graph G and a matching M in G. An* **alternating forest** *w.r.t. M in G is a forest F in G with*

- *$V(F)$ contains all the vertices exportsed by M. Each connected component of F contains exactly one exposed vertex, its* **root***.*
- *We call a vertex $v \in V(G)$ an* **outer (inner)** *vertex if it has even (odd) distance from the root of its component. (Thus, roots are outer vertices, inner vertices have degree 2)*
- *For any $v \in V(F)$ the unique path from v to the root of its connected component is M-alternating.*

A vertex $v \in V(G) \setminus V(F)$ is called **out-of-forest***.*

Proposition 1.42. *In any alternating forest the number of outer vertices that are not a root equals the number of inner vertices.*

The next lemma compromises the main idea behind Edmonds cardinality matching algorithm. By $P(x)$ we denote the unique path from $x \in V(F)$ to its root.

Lemma 1.43. *(High-level description of the algorithm)*
Given a graph G, a matching M in G, and an alternating forest F w.r.t. M in G. Then either M is a maximum matching or there exists an outer vertex $x \in V(F)$ and an edge $\{x, y\} \notin E(F)$ such that one of the following three cases holds

GROW: *$y \notin V(F)$ and we can grow the forest by adding $\{x, y\}$ and the edge matching y.*

AUGMENT: *y is an outer vertex in a different connected component of F. Now, we can augment M along $P(x) \cup \{x, y\} \cup P(y)$.*

SHRINK: *y is an outer vertex in the same connected component of F. Then we can find and shrink a blossom in the following way: let r be the first vertex of $P(x)$ also belonging to $P(y)$. If r is not a root it must have degree at least $3 \Rightarrow r$ is an outer vertex $\Rightarrow C := P(x)_{[x,r]} \cup \{x, y\} \cup P(y)_{[y,r]}$ is a blossam with ≥ 3 vertixes.*

Definition 1.44. *Given a graph G and a matching M in G. A subgraph F of G is a* **general blossom forest** *w.r.t. M if there exists a partition $V(F) = V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_k$ such that $F_i = F[V_i]$ is a maximal factor-critical subgraph of F with $|M \cap E(F_i)| = \frac{|V_i|-1}{2}$ $(i = i, \ldots, k)$ and after contracting each V_i we obtain an alternating forest F'.*

F_i is called an **outer (inner) blossom** *if V_i is an outer (inner) vertex in F'. All vertices of an outer (inner) blossom are called outer (inner).*

A **special blossom forest** *is a general blossom forest where each inner blossom is a single vertex.*

We store a special blossom forest implicitly with three functions $\mu, \varphi, \rho : V(G) \to V(G)$ satisfying

$$\mu(x) = \begin{cases} x & \text{if } x \text{ is exposed by } M \\ y & \text{if } \{x, y\} \in M \end{cases} \tag{1.6}$$

$$\varphi(x) = \begin{cases} x & \text{if } x \notin V(F) \text{ or } x \text{ is base of an outer blossom} \\ y & \text{if } x \text{ is an inner vertex and } \{x, y\} \in E(F) \setminus M \\ y & \text{if } x \text{ is an outer vertex and } \mu, \varphi \text{ are associated with an} \\ & \quad M\text{-alternating ear-decomposition of the blossom containing } x \text{ and} \\ & \quad \{x, y\} \in E(F) \setminus M \end{cases} \tag{1.7}$$

$$\rho(x) = \begin{cases} x & \text{if } x \text{ is not an outer vertex} \\ y & \text{if } x \text{ is an outer vertex and } y \text{ is the base of} \\ & \quad \text{the outer blossom in } F \text{ containing} \end{cases} \tag{1.8}$$

Within a blossom μ and φ define an M-alternating ear-decomposition and $\varphi(v)$ points to the next outer vertex on the path to the root.

Proposition 1.45. *Let F be a special blossom forest w.r.t. a matching M and let $\mu, \varphi, \rho : V(G) \to V(G)$ be satisfying (1.6), (1.7), and (1.8). Then:*

1. *For all outer vertices v*

$$P(v) := \text{maximal path given by a subsequence of } v, \mu(v), \varphi(\mu(v)), \mu(\varphi(\mu(v))), \ldots \tag{1.9}$$

   is an M-alternating v-q-path, where q is the root of the tree containing v.

2. *A vertex v is*
   - *outer $\Leftrightarrow \mu(x) = x$ or $\varphi(\mu(x)) \neq \mu(x)$.*
   - *inner $\Leftrightarrow \varphi(\mu(x)) = \mu(x)$ and $\varphi(x) \neq x$.*
   - *out-of-forest $\Leftrightarrow \mu(x) \neq x$ and $\varphi(x) = x$ and $\varphi(\mu(x)) = \mu(x)$*

Lemma 1.46. *Following invariants hold throughout the course of Algorithm 3.*

α) *The set $\{\{x, \mu(x)\} \ : \ x \in V, x \neq \mu(x)\}$ is a matching in G.*

α) *The set $\{\{x, \mu(x)\} \cup \{x, \varphi(x)\} \ : \ x \in V, (\mu(x) \neq x) \vee (\varphi(x) \neq x) \vee (\varphi(\mu(x)) \neq \mu(x))\}$ forms a special blossom forest F w.r.t. M, (the edges $\{x, \mu(x)\}$ and $\{x, \varphi(x)\}$ incident to outer and inner vertices according to Proposition 1.45, 2.).*

α) *Conditions (1.6), (1.7), and (1.8) are satisfied by μ, φ, and ρ w.r.t. F.*

Theorem 1.47. *Edmond's cardinality matching algorithm (Algorithm 3) correctly determines a maximum matching in $O(n^3)$ time, where $n := |V(G)|$*

1.6.2 Notes on even Faster Algorithms

Remark 1.48. *(Speedup* SHRINK*)*

One running time critical step is the setting of ρ during SHRINK*, which takes $O(n)$ per shrink. Using a union-find data structure as for Kruskal's* MINIMUM SPANNING TREE ALGORITHMS *this can be reduced to $O(\log n)$ (making queries to $\rho(v)$ cost $O(\log n)$). The total running time can be reduced to $O(nm \log n)$.*

By data structures of Tarjan (1975) and Gabow and Tarjan (1983), the running time can be decreased further to $O(nm\alpha(m, n))$, respectively $O(nm)$, where α is the inverse Ackermann-function.

Remark 1.49. *(Do not reset after each augmentation)*
Augment along shortest augmenting paths. Paths of equal length are vertex disjoint (see Exeercise 3.3). Instead of resetting after an augmenting path is found, continue the search for augmenting paths of the same length.

With clever data structures (among others Gabow and Tarjan (1983) as in Remark 1.48) all paths of same length can be augmented in $O(m)$, including all necessary GROW *and* SHRINK *steps.*

Algorithm 3 Edmond's Cardinality Matching Algorithm

Instance: A graph G

Output: A maximum matching M (defined by $\{x, \mu(x)\}$).

$\mu(v) := v$, $\varphi(v) := v$, $\rho(v) := v$ and $scanned(v) := \text{false}$ for all $v \in V(G)$.

OUTER VERTEX SCAN:

if all outer vertices are scanned **then return** μ

else

Let x be an outer vertex with $scanned(x) = \text{false}$.

NEIGHBOR-SEARCH:

Let y be a neighbor of x such that y is either out-of-forest or y is outer and $\rho(y) \neq \rho(x)$.

if there is no such y **then**

$scanned(x) := \text{true}$ and **goto** OUTER VERTEX SCAN.

end if

GROW:

if y is out-of-forest **then**

$\varphi(y) := x$ and **goto** NEIGHBOR-SEARCH.

end if

AUGMENT:

if $P(x)$ and $P(y)$ are vertex-disjoint **then**

$\mu(\varphi(v)) := v$, $\mu(v) := \varphi(v)$ for all $v \in V(P(x)) \cup V(P(y))$ with odd distance from x or y on $P(x)$ or $P(y)$, respectively.

$\mu(x) := y, \mu(y) := x$.

$\varphi(v) := v$, $\rho(v) := v$, $scanned(v) := \text{false}$ for all $v \in V(G)$.

goto OUTER VERTEX SCAN.

end if

SHRINK:

Let r be the first vertex on $V(P(x)) \cap V(P(y))$ with $\rho(r) = r$.

for $v \in V(P(x)_{[x,r]}) \cup V(P(y)_{[y,r]})$ with odd distance from x or y on $P(x)_{[x,r]}$ or $P(y)_{[y,r]}$, respectively, and $\rho(\varphi(v)) \neq r$ **do**

$\varphi(\varphi(v)) := v$.

end for

if $\rho(x) \neq r$ **then**

$\varphi(x) := y$.

end if

if $\rho(y) \neq r$ **then**

$\varphi(y) := x$.

end if

for all $v \in V(G)$ with $\rho(v) \in V(P(x)_{[x,r]}) \cup V(P(y)_{[y,r]})$ **do**

$\rho(v) := r$.

end for

goto NEIGHBOR-SEARCH.

end if

There are at most $2\sqrt{\nu(G)} + 2$ different path lengths (Exercise 3.3). Thus, a running time of $O(\sqrt{n}m)$ is achieved. Exercise 4.4. will cover this for bipartite graphs.

Remark 1.50. *(Skew symmetric flows)*
The fastest algorithms today use **skew-symmetric flows** *(Goldberg and Karzanow (2003); Fremuth-Paeger and Jungnickel (2003)). They generalize of maximum flows (used for bipartite graphs) to solve matching problems in general graphs. They run in $O(\sqrt{n}m\frac{\log n/m}{\log n})$.*

1.7 Gallai Edmonds Decomposition

By Berge's Formula (1.5) we can conclude the following proposition.

Proposition 1.51. *Let G be a graph and $X \subseteq V(G)$ with $|V(G)| - 2\nu(G) = q_G(X) - |X|$. Then any maximum matching of G contains a perfect matching in each even connected component of $G - X$, a near-perfect matching in each odd component of $G - X$, and matches all the vertices in X to distinct odd connected components of $G - X$.*

It turns out that each odd component is factor-critical.

Theorem 1.52. *Let $G = (V, E)$ be a graph. By Y we denote the set of vertices that are exposed by some maximum matching, i.e. $V \setminus Y$ is covered by any maximum matching, $X := \Gamma(Y)$, and $W := V \setminus (Y \cup X)$. Then*

1. *X attains $\max_{X' \subseteq V}(q_G(X') - |X'|)$ (X is a certificate of maximality),*
2. *$G[Y]$ is the union of factor-critical connected components and $G[W]$ is the union of even connected components, and*
3. *any maximum matching contains a perfect matching for $G[W]$, a near perfect matching for each connected component of $G[Y]$ and matches all vertices in X to distinct connected components in Y.*

Y, X, W is called the **Gallai-Edmonds-Decomposition** *of G*

Corollary 1.53. *A graph $G = (V, E)$ has a perfect matching if and only if $G - X$ has at most $|X|$ factor-critical components for all $X \subseteq V$.*

1.8 Weighted Matching Algorithm

For a graph $G = (V, E)$ with weights $c : E \to \mathbb{R}$, the minimum-weight perfect matching problem can be formulated as an integer program

$$\min \left\{ \sum_{e \in E} c(e) x_e \ : \ x_e \in \{0, 1\} \quad (e \in E), \sum_{e \in \delta(v)} x_e = 1 \quad (v \in V) \right\} \tag{1.10}$$

The dual of it's LP-relaxation, where $x_e \in \{0, 1\}$ is replaced by $x_e \geq 0 \quad (e \in E)$, is given by

$$\max \left\{ \sum_{v \in V} z_v \ : \ z_v + z_w \leq c(e) \quad (e = \{v, w\} \in E) \right\} \tag{1.11}$$

Proposition 1.54. *Given a graph $G = (V, E)$, weights $c : E \to \mathbb{R}$, $z \in \mathbb{R}^V$ with $z_v + z_w \leq c(e) \quad (e \in E)$ (dual feasibility), we define $G_z := (V, \{e = \{v, w\} \in E \ : \ z_v + z_w = c(e))$ (subgraph of tight edges). Furthermore, let M be a matching in G_z, F a maximal M-alternating forest in G_z, and X and Y the set of inner respectively outer vertices. Then*

1. *If M is a perfect matching (in G_z and thus G), then it is a minimum perfect matching in G.*

2. *If $\Gamma_G(y) \subseteq X$ for all $y \in Y$, then M is a maximum matching in G.*

3. *Otherwise, let*

$$\epsilon := \min\Big\{ \begin{array}{ll} \min\{ \frac{c(e)-z_v-z_w}{2} & : \ e = \{v,w\} \in E(G[Y])\}, \\ \min\{(c(e) - z_v - z_w) & : \ e = \{v,w\} \in \delta(Y) \cap \delta_G(V(F))\}\Big\}, \end{array} \tag{1.12}$$

$z'_v := z_v - \epsilon \quad (v \in X)$, $z'_v := z_v + \epsilon \quad (v \in Y)$, and $z'_v := z_v \quad (v \in V \setminus (X \cup Y))$. Then z' is (dual) feasible for (1.11). $M \cup E(F) \subseteq E(G_{z'})$, and $\Gamma_{G_{z'}}(y) \setminus X \neq \emptyset$ for some $y \in Y$.

Remark 1.55. *For bipartite graphs Proposition 1.54 is the base for the so called Hungarian Method. It iteratively augments the matching or grows the forest at an outer vertex y with $\Gamma_{G_z}(y) \setminus X \neq \emptyset$, or adjusts the dual variables to obtain such a vertex. When the matching is perfect, it is of minimum-weight.*

The LP-relaxation of (1.10) does not provide optimum solution for general (non-bipartite) graphs, as simple examples show. It allows fractional solutions, where vertices within an odd connected component B are covered entirely by edges within $E(G[B])$. For a matching, the **blossem inequalities**,

$$\sum_{e \in \delta(X)} x_e \geq 1 \quad \text{for all } X \subset V, |X| > 1 \text{ and odd}, \tag{BI}$$

must obviously hold. Let $\mathcal{A} := \{A \subseteq V(G) \ : \ |A| \text{ is odd}\}$. In the minimum-weight perfect matching algorithm we consider the following relaxation.

$$\begin{array}{rrll} & \min & \sum_{e \in E(G)} c(e) x_e & \\ s.t. & x_e & \geq 0 & (e \in E(G)) \\ & \sum_{e \in \delta(v)} x_e & = 1 & (v \in V(G)) \\ & \sum_{e \in \delta(A)} x_e & \geq 1 & (A \in \mathcal{A} : |A| > 1). \end{array} \tag{1.13}$$

The dual problem is

$$\begin{array}{rrll} & \max & \sum_{A \in \mathcal{A}} z_A & \\ s.t. & z_A & \geq 0 & (A \in \mathcal{A} : |A| > 1) \\ & \sum_{A \in \mathcal{A} : e \in \delta(A)} z_A & \leq c(e) & (e \in E(G)). \end{array} \tag{1.14}$$

Edmond's minimum-weight perfect matching algorithm starts with an empty matching ($x = 0$) and a dual feasible solution z, e.g.

$$z_A := \begin{cases} \frac{1}{2} \min\{c(e) \ : \ e \in \delta(A)\} & \text{if } |A| = 1 \\ 0 & \text{else} \end{cases}$$

At any stage z is dual feasible and the complementary slackness conditions are fulfilled

$$\begin{array}{lll} x_e > 0 & \Longrightarrow & \sum_{A \in \mathcal{A}: e \in \delta(A)} z_A = c(e) \quad (e \in E(G)) \\ z_A > 0 & \Longrightarrow & \sum_{e \in \delta(A)} x_e = 1 \qquad\quad (A \in \mathcal{A} : |A| > 1) \end{array} \tag{1.15}$$

Given a dual solution z, we call $e \in E(G)$ **tight** if

$$\sum_{A \in \mathcal{A}: e \in \delta(A)} z_A = c(e).$$

The algorithm keeps a valid (non-perfect) matching consisting of tight edges only. When x is becomes incidence vector of a perfect matching, it is a minimum-weight perfect matching, because (x, z) are a primal-dual feasible solution obeying complementary slackness (1.15).

Algorithm 4 works similar to the Hungarian Method (Proposition 1.54) except for blossoms that complicate the search for aumgmenting paths. Such blossoms are contracted in SHRINK. The algorithm maintains a set $\mathcal{B}$ of all contracted blossoms and introduces a variable z_B. Note that a blossom B is either shrunken into a larger blossom or z_B becomes positive in the next DUAL CHANGE because there is an even length path from an exposed vertex to B.

In analogy to linear programming, we define the reduced cost by

Definition 1.56. *For a graph $G = (V, E)$ with weights $c : E \to \mathbb{R}$ and dual variables z_A ($A \in \mathcal{A}$), we define the reduced costs as*

$$c_z(e) := c(e) - \sum_{A \in \mathcal{A}: e \in \delta(A)} z_A \quad \textit{for all } e \in E. \tag{1.16}$$

The minimum-weight perfect matching algorithm is described in Algorithm 4

Theorem 1.57. *There are at most $\frac{7}{2}|V|^2$ iterations of the* **repeat-until***-loop of Algorithm 4.*

Proof. At any time the collection of blossoms $\mathcal{B}$ is laminar, i.e. for all $T, U \in \mathcal{B} : T \subseteq U, U \subseteq T$, or $T \cap U = \emptyset$. Thus at any stage $|\mathcal{B}| \leq 2|V|$ (see Korte and Vygen [2012], Corollary 2.15).

Observation: Any set U added to $\mathcal{B}$ in SHRINK will not be removed from $\mathcal{B}$ (in UNPACK) before the next augmentation: after shrinking, there is an even-length M-alternating R-U-path. This path remains in G_z at least until the next AUGMENT or until U is included into another blossom $U' \supset U$. which, in turn, will not be resolved before the an augmentation. This proves the observation.

Now consider a sequence of iterations between two augmentations. The number of UNPACK iterations is bounded by $|\mathcal{B}|$ at the beginning of the sequence. The number of SHRINK iterations is bounded by $|\mathcal{B}|$ at the end of the sequence. Thus the number of UNPACK plus SHRINK iterations in the sequence is bounded by $4|V|$.

In addition there can be DUAL CHANGE iterations without UNPACK. This means that ϵ is not determined by a vertex in $\mathcal{X}$. Instead there is an edge e connecting $X \notin \mathcal{X}$ with $Y \in \mathcal{Y}$ for which the reduced cost $c_z(e)$ is decreased to zero.

If $X \notin \mathcal{Y}$, $|V(G_z) \setminus (\mathcal{X} \cup \mathcal{Y})|$ decreases. This can happen in at most $|V|$ iterations.

If $X \in \mathcal{Y}$, we will find an R-R-walk in the next iteration. Thus this can happen at most $2|V|$ times within a sequence.

As there are $\frac{1}{2}|V|$ AUGMENT-steps, we get at most $\frac{1}{2}|V|\,(4|V| + |V| + 2|V|) = \frac{7}{2}|V|^2$ iterations. □

Algorithm 4 Minimum-Weight Perfect Matching Algorithm

Instance: A graph $G = (V, E)$ with edge weights $c : E \to \mathbb{R}$
Output: A minimum-weight perfect matching M in (G, c).

$\mathcal{B} := \{\{v\} \ : \ v \in V\}; z_v := \frac{1}{2}\min\{c(e) \ : \ e \in \delta(v)\}; \quad (v \in V); M = \emptyset;$
repeat
  $G'_z := (V, \{e \in E \ : \ \sum_{A \in \mathcal{A}: e \in \delta(A)} z_A = c(e)\};$
  $G_z := G'_z/\mathcal{B}$
  $R :=$ M-exposed vertices.
  if G_z has M-alternating R-R-walk of positive length **then**
    Choose a shortest such walk P;

    AUGMENT:
    if P is a path **then**
      $M := M \triangle E(P)$;
      Start next iteration
    end if

    SHRINK: ▷ P contains M-flower with blossom B
    $\mathcal{B} := \mathcal{B} \cup B; z_B := 0; M := M \setminus E(G[B])$;
    Start next iteration
  else if G_z has no M-alternating R-R-walk of positive length **then**

    DUAL CHANGE:
    $\mathcal{X} :=$ set of vertices $B \in V(G_z)$ for which G_z has an odd-length R-B-path;
    ▷ "inner vertices"
    $\mathcal{Y} :=$ set of vertices $B \in V(G_z)$ for which G_z has an even-length R-B-path;
    ▷ "outer vertices";
    $z_B := z_B - \epsilon \quad \forall B \in \mathcal{X}$;
    $z_B := z_B + \epsilon \quad \forall B \in \mathcal{Y}$;
    where ϵ is the largest scalar preserving dual feasibility of z;

    UNPACK:
    for $B \in G_z : |B| \geq 3, z_B = 0$ **do** ▷ ($z_B > 0$ before DUAL CHANGE)
      $\mathcal{B} := \mathcal{B} \setminus B$;
      $v :=$ vertex in B covered by M; $M_B :=$ perfect matching in $G[B] - v$;
      $M := M \cup M_B$;
    end for
  end if
until M is perfect matching (in G_z)
Unpack all blossoms to obtain a perfect matching in G;

Corollary 1.58. *A minimum-weight perfect matching can be found in $O(n^2 m)$ time, where $n := |V|$.*

Remark 1.59. *As for the maximum-cardinality matching problem, we can speed up the algorithm to obtain an $O(n^3)$ running time.*

1. *using the result of the previous walk-search in subsequent walk-searches,*
2. *constructing G_z only implicitely, and*
3. *fast computations of ϵ.*

Unlike the cardinality case, blossoms may need to be unpacked to adjust dual variables inside the blossom. Thus we need to maintain the laminar blossom-hierarchy and an M-alternating forest. As blossoms may be outer, inner, or out-of-forest, we have a to maintain a general blossom forest. For a vertex $x \in V$, let $b^1(x), \dots, b^{k_x}(x)$ be the blossoms containing x such that $b^1(x) \subset \cdots \subset b^{k_x}(x)$. We use pointers

- *$\mu(x)$ (matching mate),*
- *$\varphi^i(x)$ for storing the ear-decompositions $(\mu, \varphi^i) \quad (i = 1, \dots, k_x)$, and*
- *$\rho^i(x) \quad (i = 1, \dots, k_x)$ (blossom-base pointers),*

*to store the special blossom-forest. According to Lemma **??** all the $O(|V|)$ ear-decompositions can be augmented in $O(|V|^2)$ time during an augmentation of M , resulting in $O(|V|^3)$ for all augementations.*

Another issue is the computation of ϵ. To achieve the desired time bound, we must not loop through all edges in E in each of the $O(|V|^2)$ DUAL CHANGE *steps. Instead for each pair Y, Z of disjoint sets in $\mathcal{B}$ we keep a pointer to an edge e_{YZ} minimizing $c_z(e_{YZ})$ that is void if no such edge exists.*

Moreover, for each $Y \in \mathcal{B}$ we keep a pointer e_Y with $e_Y = e_{YZ}$ for an outer set $Z \in \mathcal{B}$ that minimizes $c_z(e_{YZ})$. Note that these are the only candidates for becoming infeasible. Using the e_Y we can determine ϵ in $O(|V|)$ time.

The theoretically best running time of $O(n(m+n\log n))$ (strongly polynomial) was achieved by Gabow [1990], and $O(m\log(nW)\sqrt{n\alpha(m,n)\log n})$ (depending on W). for integral weights by Gabow and Tarjan [1991], where $n := |V|, m := |E|, W := \max\{|c(e)| \ : \ e \in E\}$.

1.8.1 The Matching Polytope

Theorem 1.60. *(Edmonds [1965]) Let $G = (V, E)$ be an undirected graph. The set of vectors $x \in \mathbb{R}^E$ satisfying*

$$\begin{array}{rcll} x_e & \geq & 0 & (e \in E) \\ \sum_{e\in\delta(v)} x_e & = & 1 & (v \in V) \\ \sum_{e\in\delta(A)} x_e & \geq & 1 & (A \in \mathcal{A}) \end{array} \tag{1.17}$$

is the convex hull of all perfect matchings. It is called the **perfect matching polytope**.

Theorem 1.61. *(Edmonds [1965]) Let $G = (V, E)$ be an undirected graph. The set of vectors $x \in \mathbb{R}^E$ satisfying*

$$\begin{array}{rcll} x_e & \geq & 0 & (e \in E) \\ \sum_{e\in\delta(v)} x_e & \leq & 1 & (v \in V) \\ \sum_{e\in E(G[A])} x_e & \leq & \frac{|A|-1}{2} & (A \in \mathcal{A}) \end{array} \tag{1.18}$$

is the convex hull of all matchings. It is called the **matching polytope**.

Theorem 1.62. *(Cunningham and Marsh [1978]) Let $G = (V, E)$ be an undirected graph. The system (1.18) is TDI.*

Theorem 1.63. *Let $G = (V, E)$ be a graph*

$$\begin{aligned} P &:= \min\left\{\sum_{e\in E} c(e)x_e \;:\; x_e \geq 0 \;(e \in E), \sum_{e\in\delta(v)} x_e \leq 1 \;(v \in V)\right\} \\ Q &:= \min\left\{\sum_{e\in E} c(e)x_e \;:\; x_e \geq 0 \;(e \in E), \sum_{e\in\delta(v)} x_e = 1 \;(v \in V)\right\} \end{aligned}$$

are called the **fractional matching polytope** *and the* **fractional perfect matching polytope**. *If G is bipartite, P and Q are integral.*

Theorem 1.64. *Let $G = (V, E)$ be a graph and Q be the fractional perfect matching polytope. The vertices of Q are half-integral:*

$$x_e = \begin{cases} \frac{1}{2} & \text{if } e \in E(C_1) \cup \cdots \cup E(C_k), \\ 1 & \text{if } e \in M, \\ 0 & \text{otherwise,} \end{cases}$$

where $C_1, \ldots, C_k$ are vertex disjoint odd circuits and M is a perfect matching in $G - (V(C_1) \cup \cdots \cup V(C_k))$.

2 Extended Formulations

2.1 The Spanning Tree Polytope

Theorem 2.1. *(Edmonds [1907]) Given a connected undirected graph G, let $n := |V(G)|$. Then the polytope*

$$P := \left\{ x \in [0,1]^{E(G)} \ : \ \sum_{e \in E(G)} x_e = n-1, \sum_{e \in E(G[x])} x_e \leq |X| - 1 \text{ for } \emptyset \neq X \subset V(G) \right\}$$

is integral. Its vertices are the incidence vectors of spanning trees of G and P is called the **spanning tree polytope**.

Theorem 2.2. *(Fulkerson [1974]) Let G be a digraph with weights $c : E(G) \to \mathbb{Z}_+$, and $r \in V(G)$ such that G contains a spanning arborescence rooted at r. Then the minimum weight of a spanning arborescence rooted at r equals the maximum number t of r-cuts $C_1, \ldots, C_t$ (repetitions allowed) such that no edge e is contained in more than $c(e)$ of these cuts.*

Corollary 2.3. *Let G be a digraph with weights $c : E(G) \to \mathbb{R}_+$, and $r \in V(G)$ such that G contains a spanning arborescence rooted at r. Then the LP*

$$\min \left\{ c^\intercal x \mid x \geq 0, \sum_{e \in \delta^+(X)} x_e \geq 1 \text{ for all } X \subset V(G) \text{ with } r \in X \right\}$$

has an integral optimum solution, which is the incidence vector of a minimum weight spanning minimum weight rooted arborescence problem rooted at r, plus possibly some edges of zero weight.

2.2 Relaxation Complexity

Definition 2.4. *Let $P \subseteq \mathbb{R}^n$ a polyhedron. A polyhedron $Q \subseteq \mathbb{R}^m$ is an* **extension** *of P if there is a projective map $\pi : \mathbb{R}^m \to \mathbb{R}^n$ with $\pi(Q) = P$. The* **extension complexity** *of a polyhedron P is the minimum number of facets of an extension Q of P.*

Theorem 2.5. *The spanning tree polytope has a polynomial extension complexity*

Proof. Exercise class. □

Remark 2.6.
- *Yannakakis [1991] showed that the traveling salesman polytope does not have a symmetric extension of polynomial size.*
- *Fiorini et. al [2012] showed that the traveling salesman polytope does not have a non-symmetric extension of polynomial size.*
- *Rothvoss [2013] showed that the matching polytope does not have an extension of polynomial size.*

These results are not proven in this lecture. Instead, we study a related question. Consider the traveling salesman problem. The standard IP-formulation uses decision variables $x_{ij} \in \{0,1\}$ $(1 \leq i \neq j \leq n)$ that are one if the tour is using the edge from i to j and (exponentially many) subtour elimination constraints. However, there is a polynomial size IP-formulation that labels n cities from $1, \ldots, n$ and uses vertex labels u_i $(i = 1 \ldots, n)$ (lifting variables) in addition to the decision variables. One question is whether one can omit the lifting variables:

"Is there a integer programming formulation for the traveling salesman problem without additional variables (except for $x_{ij}, 1 \leq i \neq j \leq n$) and only polynomialy many inequalities?"

In this sense Kaibel and Weltge introduced the term *relaxation complexity*:

Definition 2.7. *Let $X \subseteq \mathbb{Z}^n$. A set $R \subseteq \mathbb{R}^n$ is a* **relaxation** *of X if $R \cap \mathbb{Z}^n = \operatorname{conv}(X) \cap \mathbb{Z}^n$. The minimum number of facets of any relaxation of X is called* **relaxation complexity** $rc(X)$.

Note that matching has an exponential extension complexity, but a polynomial relaxation complexity.

Definition 2.8. *Let $X \subset \mathbb{Z}^n$, $H \subseteq \operatorname{aff}(X) \cap \mathbb{Z}^n \setminus \operatorname{conv}(X)$ is called a* **hiding set** *for X if for any two $a, b \in H : \operatorname{conv}\{a, b\} \cap \operatorname{conv}(X) \neq \emptyset$.*

We call a set $X \subseteq \mathbb{Z}^n$ *polyhedral* if $\operatorname{conv}(X)$ is a polyhedron).

Proposition 2.9. *Let $X \subseteq \mathbb{Z}^n$ be polyhedral and let $H \subseteq \operatorname{aff}(X) \cap \mathbb{Z}^n \setminus \operatorname{conv}(X)$ be a hiding set for X. Then $rc(X) \geq |H|$.*

Proof. Let R be a relaxation for X. As $H \subseteq \operatorname{aff}(X) \subseteq \operatorname{aff}(R)$, any $x \in H$ must be separated by facet-defining inequalities of R. Let $a^\intercal x \leq \beta$ be an inequality of R that is violated by $x', x'' \in H$. Then, there is an $x^\star \in \operatorname{conv}\{x', x''\} \cap \operatorname{conv}(X)$. But as $x', x'' \notin R$ their convex combination $x^\star$ fulfills $a^\intercal x^\star > \beta$, contradicting the validity of the inequality in $R \supseteq \operatorname{conv}(X)$.

Therefore any facet-defining inequality of R is violated by at most one point in H, and R has at least $|H|$ facets. □

Constructing appropriate hiding sets allows us to prove lower bounds on the relaxation complexity.

Theorem 2.10. *(Kaibel and Weltge [2014]) The asymptotic growth of the relaxation complexities of the traveling salesman polytopes $rc(ATSP_n)$ and $rc(TSP_n)$ are $2^{\Theta(n)}$, where*

$$ATSP_N := \{\chi(T)|; : \ T \text{ is a directed Hamiltonian Cycle in the complete directed graph}\},$$

and

$$TSP_N := \{\chi(T)|; : \ T \textit{ is a Hamiltonian Cycle in the complete undirected graph}\}.$$

Proof. We first consider the $ATSP$. Let $n = 2N + 2$ for some $N \in \mathbb{N}$ and define a graph with vertex set

$$V := \{v_i, v_i' \ : \ i \in \{1 \dots, N+1\}\} \cup \{w_i, w_i' \ : \ i \in \{1 \dots, N\}\}.$$

Then $|V| = 4N + 2$ and let A be the arc set of the complete directed graph with vertex set V. For a vector $b \in \{0,1\}^N$, we define two vertex disjoint cycles

$$C_b := \{(v_{N+1}, v_1)\} \cup \bigcup_{i:b_i=0} \{(v_i, w_i), (w_i, v_{i+1})\} \cup \bigcup_{i:b_i=1} \{(v_i, w_i'), (w_i', v_{i+1})\}$$

and

$$C_b' := \{(v_{N+1}', v_1')\} \cup \bigcup_{i:b_i=0} \left\{(v_i', w_i'), (w_i', v_{i+1}')\right\} \cup \bigcup_{i:b_i=1} \left\{(v_i', w_i), (w_i, v_{i+1}')\right\}.$$

Let $H_N := \{\chi(C_B \cup C_b') \ : \ b \in \{0,1\}^N\}$.

Claim: H_N is a hiding set for $ATSP_{4N+2}$. Note that

$$H_N \subseteq \text{aff}(ATSP_{4N+s}) = \left\{x \in \mathbb{R}^A \ : \ x(\delta^-(v) = x(\delta^+(v) = 1 \text{ for all } v \in V\right\}.$$

Let $b^1, b^2 \in \{0,1\}^N$ with $b^1 \neq b^2$ Without loss of generality there is an index $j \in N$ with $b_j^1 = 0$ and $b_j^2 = 1$. Now consider the arc sets $(C_{b^1} \cup C_{b^1}')$, $(C_{b^2} \cup C_{b^2}')$, and their modifications

$$T_1 := (C_{b^1} \cup C_{b^1}') \setminus \{(v_j, w_j), (v_j', w_j')\} \cup \{(v_j, w_j'), (v_j', w_j)\}$$

and

$$T_1 := (C_{b^2} \cup C_{b^2}') \setminus \{(v_j, w_j'), (v_j', w_j)\} \cup \{(v_j, w_j), (v_j', w_j')\}.$$

Then T_1 and T_2 are Hamiltonian cycles and

$$\begin{aligned} \tfrac{1}{2}\left(\chi(T_1) + \chi(T_2)\right) &= \tfrac{1}{2}\left(\chi(C_{b^1} \cup C_{b^1}') - \chi(\{(v_j, w_j), (v_j', w_j')\}) + \chi(\{(v_j, w_j'), (v_j', w_j)\})\right) \\ &\quad + \tfrac{1}{2}\left(\chi(C_{b^2} \cup C_{b^2}') - \chi(\{(v_j, w_j'), (v_j', w_j)\}) + \chi(\{(v_j, w_j), (v_j', w_j')\})\right) \\ &= \tfrac{1}{2}\left(\chi(C_{b^1} \cup C_{b^1}') + \chi(C_{b^2} \cup C_{b^2}')\right), \end{aligned}$$

which proves the claim. Similarly, H_N is a hiding set for TSP_{4N+2} when replacing all directed arcs by undirected edges. By Proposition 2.9, $rc(TSP_{4N+2}), rc(ATSP_{4N+2}) \geq |H_N| = 2^{\Omega(n)}$. As relaxations with $\mathcal{O}(2^n)$ many inequalities are known, this shows the asymptotic growth. □

The result can be applied to arborescences and spanning trees as well.

Theorem 2.11. *(Kaibel and Weltge [2014]) Let*

$$ARB_n := \{\chi(T) \ : \ T \text{ is an arborescence in the complete directed graph on } n \text{ vertices}\}$$

and

$$SPT_n := \{\chi(T) \ : \ T \text{ is a spanning tree in } K_n\}.$$

Then the asymptotic growth of $rc(ARB_n)$ *and* $rc(SPT_n)$ *is* $2^{\Theta(n)}$.

Proof. The proof works analogously to the proof of Theorem 2.10, when removing the arc (v'_{N+1}, v'_1) from C'_b. Then $C_b \cup C'_b$ is the node-disjoint union of a cycle and a path, and T_1 and T_2 become spanning arborescences (actually paths), H_N remains a hiding set for this problem and we apply Proposition 2.9 again. □

3 T-Joins and b-Matchings

Consider the following problem. Starting from a graph G, e.g. $G = K_n$ with edge weigths $c : E(G) \to \mathbb{R}$, and a connected subgraph $H \subset G$. Now you want to make H Eulerian by adding edges minimizing the total weight of the extra edges. To this end you need to join pairs of odd degree vertices. This is what T-joins do.

3.1 T-Joins

Definition 3.1. *Let $G = (V, E)$ be an undirected graph and let $T \subseteq V(G)$. A subset $J \subseteq E$ is called a T-**join** if T is equal to the set of vertices of odd degree in (V, J).*

Proposition 3.2. *Let G be a graph $T, T' \subseteq V(G)$, J a T-join, and J' a T'-join. Then $J \triangle J'$ is a $(T \triangle T')$-join.*

Proposition 3.3. *Let G be a graph and $T \subseteq V(G)$. There exists a T-join in G if and only if $|V(C) \cap T|$ is even for each connected component C.*

Observe that each T-join is the edge-disjoint union of circuits and $\frac{1}{2}|T|$ paths connecting dijoint pairs of vertices in T.

MINIMUM WEIGHT T-JOIN PROBLEM	
Input:	an (undirected) graph G, $T \subseteq V(G)$, and weights $c : E(G) \to \mathbb{R}$.
Task:	find a minimum weight T-join in G or decide that no T-join exists.

Theorem 3.4. *Given a graph $G = (V, E)$, edge weights $c : E \to \mathbb{R}$, and a subset $T \subseteq V$, a shortest T-join can be found in strongly polynomial time.*

Corollary 3.5. *Assuming integral weights, the* MINIMUM WEIGHT T-JOIN PROBLEM *can be solved in time $O(APSP_+(n, m, L) + MWPM(n, n^2, nL))$, where $n = |V|, m = |E|$, L is the maximum absolute edge weight, $APSP_+(n, m, L)$ and $MWPM(n, m, L)$ are the times needed to solve the all-pairs shortest path problem respectively the* MINIMUM WEIGHT PERFECT MATCHING PROBLEM *in an undirected graph with n vertices, m edges and integer weights of absolute value at most L, where for $APSP_+$ the weights are non-negative.*

Corollary 3.6. *Given a graph $G = (V, E)$, edge weights $c : E \to \mathbb{R}$, and a subset $T \subseteq V$, a longest T-join can be found in strongly polynomial time.*

Corollary 3.7. *Given a graph $G = (V, E)$, edge weights $c : E \to \mathbb{R}$, one can check if there is a negative-length circuit in strongly polynomial time.*

3.2 T-Join Applications

3.2.1 Christofides' Approximation Algorithm for the METRIC TSP

Let (K_n, c) be an instance for the traveling salesman problem (TSP) on a complete graph with edge weights $c : E(K_n) \to \mathbb{R}$ satisfying the triangle inequality. Obviously every tour is a tree in K_n, actually a path, plus one additional edge. The length of a minimum spanning tree T_{MST} provides a lower bound $c(E(T_{MST}))$ for the length of a tour. Thus, an Eulerian walk in the graph G in the vertex set $V(K_n)$ containing two copies of each edge of T_{MST} induces a tour, visiting the vertices in the order as they first appear in the Eulerian walk. By the triangle-inequality the length of the tour is no longer than the length of the walk.

Doubling all edges in T_{MST} is a very simple way to make the graph Eulerian. The shortest way to accomplish this is to compute a T-join, where T is the set of odd-degree vertices.

Algorithm 5 Christofides' Algorithm

Instance: an instance (K_n, c) of the METRIC TSP.
Output: a tour

Find a minimum weight spanning tree T in (K_n, c).
Let W be the set of vertices with odd degree in T.
Find a minimum weight W-join J in K_n w.r.t. c.
Find an Eulerian walk in $(V(K_n), E(T) \dot\cup J)$.
Add vertices to the TSP-tour in order of the first appearance in the Eulerian walk.

Theorem 3.8. *(Christifides [1976]) Christofides' Algorithm is a $\frac{3}{2}$-factor approximation algorithm for the* METRIC TSP.

After 36 years this is still the best known approximation-factor for metric TSP-instances.

3.2.2 The Shortest Path Problem for Undirected Graphs

Let G be an undirected graph with edge weights $c : E(G) \to \mathbb{R}$ and $s, t \in V(G)$. If $c \geq 0$ we can compute a shortest s-t-path by transforming the problem into a directed problem in the directed graph $G' = (V(G), \{\{(v, w), (w, v)\} \ : \ \{v, w\} \in E(G)\})$ with edge costs c' defined by $c'(v, w) := c'(w, v) := c(\{v, w\})$ for all $\{v, w\} \in E(G)$.

In case of negative weigths this leads to negative cycles for which the directed shortest path problem becomes *NP*-hard. We can still solve the undirected problem using T-joins.

Corollary 3.9. *Given $G, s, t \in V(G)$, $c : E(G) \to \mathbb{Q}$ s.t. each circuit has non-negative length. A shortest s-t-path can be found in stronlgy polynomial time.*

3.2.3 The Chinese Postman Problem

Definition 3.10. *A walk $C = (v_0, e_1, v_1, \ldots, e_t, v_t)$ in a graph G is called a* **Chinese postman tour** *if $v_t = v_0$ and each edge of G occurs at least once in C.*

CHINESE POSTMAN PROBLEM	
Input:	an connected graph G, weights $c : E(G) \to \mathbb{R}_+$.
Task:	find a shortest Chinese postman tour.

Background: G models a street map, where a postman has to deliver letters. He needs to traverse every street and wants to minimize his walk/route.

If G is Eulerian, an Euler walk is a shortest Chinese postman tour.

Corollary 3.11. *The* CHINESE POSTMAN PROBLEM *can be solved in strongly polynomial time.*

3.3 T-Joins and T-Cuts

Definition 3.12. *Let G be a graph and $T \subseteq V(G)$. A T-cut is a cut $C = \delta(X)$ with $|X \cap T|$ odd for some $X \subseteq V(G)$.*

Proposition 3.13. *Let G be a graph and $T \subseteq V(G)$ with $|T|$ even.*

1. *For any T-join J and any T-cut C we have $J \cap C \neq \emptyset$.*
2. *Inclusion wise minimal T-cuts (T-joins) are exactly the inclusion wise minimal edge sets intersecting all T-joins (T-cuts).*

Thus, the minimum cardinality of a T-join is not less than the maximum number of edge-disjoint T-cuts. Equality does not hold as $G = K_4, T = V(G)$ shows. For bipartite graphs equality holds:

Definition Let G be a graph. A familiy of cuts $\delta(X_1), \dots, \delta(X_k)$, with $X_1, \dots, X_k \subseteq V(G)$ is called **cross-free** if the vertex sets $X_1, \dots, X_k$ are cross-free, i.e. for all $1 \leq i < j \leq k$ $X_i \subseteq X_j$ or $X_j \subseteq X_i$ or $X_i \cap X_j = \emptyset$ or $X_i \cup X_j = V(G)$.

Theorem 3.14. *(Seymour [1981]) Let G be a bipartite graph and $T \subseteq V(G)$ such that a T-join in G exists. Then the minimum cardinality of a T-join equals the maximum number of pairwise edge-disjoint T-cuts. This maximum is attained by a cross-free family of cuts.*

Corollary 3.15. *Let G be a graph, $c : E(G) \to \mathbb{Z}_+$, and and $T \subseteq V(G)$ such that a T-join exists. The minimum weight of a T-join is equal to half of the maximum number of T-cuts covering each edge $e \in E(G)$ at most $2c(e)$ times. The maximum is attained by a cross-free family of cuts.*

3.3.1 The T-Join Polytope

Let G be a graph and $T \subseteq V(G)$. The T-join polytope denoted by $P_{T-\text{join}}(G)$ is the convex hull of the incidence vectors of T-joins. The **up-hull** of $P_{T-\text{join}}(G)$ or T**-join polyhedron** is the polyhedron

$$P^{\uparrow}_{T-\text{join}}(G) := P_{T-\text{join}}(G) + \mathbb{R}^{E(G)}_+, \tag{3.1}$$

Corollary 3.16. *The polyhedron $P^{\uparrow}_{T-join}(G)$ is determined by the system*

$$\begin{array}{rcll} x_e & \geq & 0 & \text{for each } e \in E(G), \\ x(C) & \geq & 1 & \text{for each } T\text{-cut } C. \end{array} \tag{3.2}$$

3.4 Excursus on Gomory-Hu Trees

In this section we consider the following problem. Given an undirected graph G, edge capacities $u : E(G) \to \mathbb{R}_+$ we want to compute a minimum capacity cut, i.e. set $X \subset V(G)$ such that $u(\delta(X))$ is minimum. For a given pair of vertices $s, t \in V(G)$, we can compute a minimum capacity cut by transforming the problem into a minimum capacity cut problem in the directed graph $(V(G), \{(v,w),(w,v) : \{v,w\} \in E(G)\}$ with capacities u' defined by $u'(v,w) = u'(w,v) = u(\{v,w\})$. Picking a vertex $s \in V(G)$ and computing a minimum capacity s-t-cut for all $t \in V(G) \setminus \{s\}$, we can determine X in $(|V(G)| - 1)$ minimum-cut computations.

However, when we have additional requirements on the cut, e.g. if we are seeking a T-cut, it can be benefitial to compute minimum capacity s-t-cuts for all choices of $s, t \in V(G)$. Gomory and Hu invent a tree structure that represents all minimum s-t-cuts ($s, t \in V(G)$), and whose computation requires only $|V(G)| - 1$ minimum-capacity s-t-cut computations. This will help us to compute minimum capacity T-cuts

Definition 3.17. *Let G be a graph and $u : E(G) \to \mathbb{R}_+$ a capacity function. For vertices $s, t \in V(G)$ we denote by λ_{st} their* **local edge-connectivity***, which is the minimum capacity of a cut separating s and t.*

Lemma 3.18. *For all $u, v, w \in V(G)$: $\lambda_{uw} \geq \min\{\lambda_{uv}, \lambda_{vw}\}$.*

Definition 3.19. *Given G and u as above, a tree T is called a* **Gomory-Hu-tree** *for (G, u) if $V(T) = V(G)$ and*

$$\lambda_{st} = \min_{e \in E(P_{st})} u(\delta_G(C_e)) \text{ for all } s, t \in V(G),$$

where P_{st} is the unique s-t-path in T, and for $e \in E(T)$, C_e and $V(G) \setminus C_e$ are the connected components of $T - e$. ($\delta(C_e)$ is the fundamental cut of e w.r.t. T)

The following proposition provides an alternative definition and motivates the algorithm for constructing Gomory-Hu trees (though we do not know whether they exist so far):

Lemma 3.20. *Given G and u as above, and a tree T with vertex set $V(T) = V(G)$. T is a Gomory-Hu tree for (G, u) if and only if for each edge $e = \{s, t\} \in E(T)$, $\delta_G(C_e)$ is a minimum-capacity s-t-cut, where C_e and $V(G) \setminus C_e$ are the connected components of $T - e$.*

Lemma 3.21. *Let G be a graph and $u : E(G) \to \mathbb{R}_+$ a capacity function. Let $s, t \in V(G)$ and let $\delta(A)$ be a minimum capcacity s-t-cut in (G, u). For $s', t' \in V(G) \setminus A$, let (G', u') arise from (G, u) by contracting A to a single vertex. Then for any minimum s'-t'-cut $\delta(K \cup \{A\})$ in (G', u'), $\delta(K \cup A)$ is a minimum s-t-cut in (G, u).*

Algorithm 6 Gomory-Hu Algorithm

Input: An undirected graph G and a capacity function $u : E(G) \to \mathbb{R}_+$.
Output: Gomory-Hu tree T for (G, u).

Set $V(T) := \{V(G)\}$ and $E(T) := \emptyset$.

CHOOSECOMPONENT:
Choose some $X \in V(T)$ with $|X| \geq 2$.
if no such X exists **then goto** FINISHTREE
end if
Choose $s, t \in X$ with $s \neq t$.

CONTRACT:
for each connected component C of $T - X$ **do**
Let $S_C := \bigcup_{Y \in V(C)} Y$.
end for
Let (G', u') arise from (G, u) by contracting S_C to a single vertex v_C for each connected component C of $T - X$.

MINCUT:
Find a minimum s-t-cut $\delta(A')$ in (G', u').
Let $B' := V(G') \setminus A'$.
Set $A := \left(\bigcup_{v_C \in A' \setminus X} S_C \right) \cup (A' \cap X)$ and $B := \left(\bigcup_{v_C \in B' \setminus X} S_C \right) \cup (B' \cap X)$.

MODIFYTREE:
Set $V(T) := (V(T) \setminus \{X\}) \cup \{A \cap X, B \cap X\}$.
for each edge $e = \{X, Y\} \in E(T)$ incident to the vertex X **do**
if $Y \subseteq A$ **then**
set $e' := \{A \cap X, Y\}$
else
set $e' := \{B \cap X, Y\}$.
end if
Set $E(T) := (E(T) \setminus \{e\}) \cup \{e'\}$ and $w(e') := w(e)$.
end for
Set $E(T) := E(T) \cup \{\{A \cap X, B \cap X\}\}$.
$w(\{A \cap X, B \cap X\}) := u'(\delta_{G'}(A'))$.
goto CHOOSECOMPONENT.

FINISHTREE:
Replace all $\{x\} \in V(T)$ by x and all $\{\{x\}, \{y\}\} \in E(T)$ by $\{x, y\}$. .

Lemma 3.22. *Each time at the end of* MINCUT *we have*

1. $A \dot\cup B = V(G)$
2. $E(A, B)$ *is a minimum* s-t*-cut in* (G, u).

Lemma 3.23. *At any stage of the algorithm before* FINISHTREE *we have*

$$w(e) = u\left(\delta_G\left(\bigcup_{Z \in C_e} Z\right)\right) \text{ for all } e \in E(T),$$

where C_e *and* $V(G) \setminus C_e$ *are the connected components of* $T - e$. *Moreover, for all* $e = \{P, Q\} \in E(T)$ *there are vertices* $p \in P, q \in Q$ *with* $\lambda_{pq} = w(e)$.

Theorem 3.24. *(Gomory and Hu [1961]) Every undirected graph has a Gomory-Hu tree. The Gomory-Hu Algorithm finds a Gomory-Hu tree in* $O(n^3\sqrt{m})$ *time.*

3.5 Finding Minimum-Capacity T-Cuts

MINIMUM CAPACITY T-CUT PROBLEM	
Input:	an undirected graph G, $T \subseteq V(G)$ with $\lvert T\rvert$ even , and capacities $u : E(G) \to \mathbb{R}_+$.
Task:	find a minimum capacity T-cut in G.

Theorem 3.25. *Let* G *be a graph with capacities* $u : E(G) \to \mathbb{R}_+$. *Let* H *be a Gomory-Hu tree for* (G, u). *Let* $T \subseteq V(G)$ *with* $|T| \geq 2$ *even. Then a minimum capacity* T*-cut can be found among the fundamental cuts of* H. *It can be found in* $O(n^4)$ *time.*

3.6 b-Matchings

Definition 3.26. *Let* G *be a graph with capacities* $u : E(G) \to \mathbb{N} \cup \{\infty\}$ *and numbers* $b : V(G) \to \mathbb{N}$. *A* b**-matching** *is a function* $f : E(G) \to \mathbb{N}$ *such that* $f \leq u$ *(component-wise) and* $\sum_{e \in \delta(V)} f(e) \leq b(v)$ *for all* $v \in V(G)$.

If $u \equiv 1$, *we call* f *a* **simple** b*-matching, and if* $\sum_{e \in \delta(v)} f(e) = b(v)$, *we call* f *a perfect* b*-matching.*

Simple perfect b-matchings are also called b**-factors** and simple 1-matchings are obviously matchings.

MAXIMUM WEIGHT b-MATCHING PROBLEM	
Input:	an (undirected) graph G, capacities $u : E(G) \to \mathbb{N} \cup \{\infty\}$, weights $c : E(G) \to \mathbb{R}_+$, and numbers $b : V(G) \to \mathbb{N}$.
Task:	find a b-matching f in (G, u) with maximum weight $\sum_{e \in E(G)} c(e) f(e)$.

The probably most prominent occurence of b-matchings with $b \neq 1$, i.e. except for matchings), is in the context of the TSP-problem: each Hamiltonian circuit is a simple perfect 2-matching. Thus, every valid inequality for the (simple) perfect 2-matching polytope is also valid for the TSP polytope. This motivates the examination of the b-matching polytope.

First, we take a look at the polytope with infinte capacities.

Theorem 3.27. *(Edmonds [1965]) Let G be an undirected graph and $b : V(G) \to \mathbb{N}$. The b-matching polytope of (G, ∞) is the set of vectors $x \in \mathbb{R}^{E(G)}$ satisfying*

$$\begin{array}{rcll} x & \geq & 0 & \\ \sum_{e \in \delta(v)} x_e & \leq & b(v) & (v \in V(G)) \\ \sum\limits_{e \in E(G[X])} x_e & \leq & \left\lfloor \frac{1}{2} \sum\limits_{v \in X} b(v) \right\rfloor & (X \subseteq V(G)) \end{array} \tag{3.3}$$

We can use the uncapacitated result for describing the b-matching polytope with edge capacities.

Theorem 3.28. *(Edmonds [1965]) Let G be an undirected graph, $u : E(G) \to \mathbb{N} \cap \{\infty\}$ and $b : V(G) \to \mathbb{N}$. The b-matching polytope of (G, u) is the set of vectors $x \in \mathbb{R}^{E(G)}$ satisfying*

$$\begin{array}{rcll} x & \geq & 0 & \\ x & \leq & u & \\ \sum_{e \in \delta(v)} x_e & \leq & b(v) & (v \in V(G)) \\ \sum\limits_{e \in E(G[X])} x_e + \sum\limits_{e \in F} x_e & \leq & \left\lfloor \frac{1}{2} \left(\sum\limits_{v \in X} b(v) + \sum\limits_{e \in F} u(e) \right) \right\rfloor & (X \subseteq V(G), F \subseteq \delta(X)). \end{array} \tag{3.4}$$

Theorem 3.29. *(Tutte [1952]) Let G be an undirected graph, $u : E(G) \to \mathbb{N} \cap \{\infty\}$, and $b : V(G) \to \mathbb{N}$. (G, u) has a perfect b-matching if and only if for any two subsets $X, Y \subset V(G)$ with $X \cap Y = \emptyset$, the number of connected components C in $G - X - Y$ for which $\sum_{v \in V(C)} b(c) + \sum_{e \in E(V(C),Y)} u(e)$ is odd is upper bounded by*

$$\sum_{v \in X} b(v) + \sum_{y \in Y} \left(\sum_{e \in \delta(y)} u(e) - b(y) \right) - \sum_{e \in E(X,Y)} u(e).$$

3.7 The Padberg-Rao Theorem

Lemma 3.30. *(Letchford, Reinelt, and Theis [2008]) Let G be an undirected graph with $|E(G)| \geq 1$, $T \subseteq V(G)$ with $|T|$ even, and $c, c' : E(G) \to \mathbb{R}_+ \cup \{\infty\}$. Then there is an $O(n^4)$-algorithm that finds sets $X \subseteq V(G)$ and $F \subseteq \delta(X)$ such that $|X \cap T| + |F|$ is odd and*

$$\sum_{e \in \delta(X) \setminus F} c(e) + \sum_{e \in F} c'(e)$$

is minimum.

Theorem 3.31. *(Padberg and Rao [1982]) Let G be an undirected graph $u : E(G) \to \mathbb{N} \cup \{\infty\}$ and $b : V(G) \to \mathbb{N}$. The* SEPARATION PROBLEM *for the b-matching polytope of (G, u) can be solved in $O(n^4)$ time.*

Corollary 3.32. *The* MAXIMUM WEIGHT b-MATCHING PROBLEM *can be solved in polynomial time.*

Ko

4 Matroids & Generalization

4.1 Properties, Axioms, Constructions

Definition 4.1. *A set system $(E, \mathcal{F})$ is called an* **independence system** *if*

(M1) $\emptyset \in \mathcal{F}$

(M2) *If $X \subseteq Y \in \mathcal{F}$, then $X \in \mathcal{F}$.*

The elements of $\mathcal{F}$ are called **independent***. An inclusionwise maximal independent subset of a set $A \subseteq E$ is called a* **basis** *of A. Its cardinality is called the* **rank** *of A denoted by $r(A)$. Minimal dependent sets are called* **circuits***.*

An independece system $(E, \mathcal{F})$ is called a **matroid** *if the following axiom holds.*

(M3) *For $X, Y \in \mathcal{F}$ with $|X| > |Y|$ there is an $x \in X \setminus Y$ such that $Y \cup \{x\} \in \mathcal{F}$.*

Algorithm 7 Best-In Greedy Algorithm

Instance: An independece system $(E, \mathcal{F})$ and $c : E \to \mathbb{R}$.
Task: Find an $X \in \mathcal{F}$ such that $c(X)$ is maximmum.

```
X = ∅;
while ∃x ∉ X with c(x) > 0 and X ∪ {x} ∈ F do
    Choose such an x with c(x) maximum
    X := X ∪ {x}
end while
```

Theorem 4.2. *An independece system $(E, \mathcal{F})$ is a matroid if and only if Algorithm 7 finds an optimal independent set for every $c \in \mathbb{R}^E$.*

Example 4.3. *There are some well-known matroids (proofs left to the reader):*

Vector matroid: *E is the set of columns of a matrix A over some field $\mathbb{K}$ and $\mathcal{F} := \{F \subseteq E : \text{the columns in } F \text{ are linear independent}\}$.*

Cycle matroid: *E is the edge set of an undirected graph G and $\mathcal{F} := \{F \subseteq E : (V(G), F) \text{ is a forest}\}$.*

Graphic matroid: *a cycle matroid where G may contain loops.*

Matching matroid: *E is the vertex set of an undirected graph G and $\mathcal{F} := \{F \subseteq E : F \text{ is covered by some matching in } G\}$.*

Uniform matroid: *E is finite set, $k \in \mathbb{N}$, and $\mathcal{F} := \{F \subseteq E \ : \ |F| \leq k\}$.*

Algebraic matroid: *Let $\mathbb{F}$ be a field with field extension $\mathbb{K}$, E a finite subset of $\mathbb{K}$, and $\mathcal{F} := \{F \subseteq E \ : \ F$ is algebraically independent in $\mathbb{F}\}$. The rank of the matroid $(E, \mathcal{F})$ is the degree of transcendence of E.*

Example 4.4. *Examples for independent set systems that are no matroids are*

- *matchings*
- *stable sets and cliques,*
- *feasible solutions of knapsack problems*
- *subsets of traveling salesman tours or Steiner trees*

Theorem 4.5. *(Edmonds [1970]) Let $(E, \mathcal{F})$ be a matroid and $r : 2^E \to \mathbb{Z}_+$ its rank function. Then the* **matroid polytope** *of $(E, \mathcal{F})$, i.e. the convex hull of the incidence vectors of elements of $\mathcal{F}$, is equal to*

$$\left\{ x \in \mathbb{R}^E \ : \ x \geq 0, \sum_{e \in A} x_e \leq r(A) \text{ for all } A \subseteq E \right\}. \tag{4.1}$$

Corollary 4.6. *Let $M = (E, \mathcal{F})$ be a matroid, let $c \in \mathbb{R}^E$, and $J \in \mathcal{F}$. Then J is a maximum-weight independent set w.r.t. c if and only if*

a) $e \in J$ implies $c_e \geq 0$;

b) $e \notin J, J \cup \{e\} \in \mathcal{F}$ implies $c_e \leq 0$;

c) $e \notin J, f \in J, (J \cup \{e\}) \setminus \{f\} \in \mathcal{F}$ implies $c_e \leq c_f$.

Furthermore, if J is a basis, it is a maximum weight basis if and only if

$$e \notin J, f \in J, (J \cup \{e\}) \setminus \{f\} \in \mathcal{F} \text{ imply } c_e \leq c_f.$$

Theorem 4.7. *Let G be a graph. The convex hull of characteristic vectors of forests of G is the of incedence vectors satisfying*

$$\begin{aligned} x(\delta(T)) &\leq |T| - 1 \quad \text{for all } \emptyset \subset T \subseteq V(G), \\ x &\geq 0. \end{aligned} \tag{4.2}$$

4.2 Matroid Intersection

Matchings are obviously no matroids. However, bipartite matchings are in the intersection of two transversal matroids (one for each vertex partition).

Similarly branchings in directed graphs are in the intersection of the cycle matroid of the underlying undirected graph and the edge sets having at most one entering edge per vertex.

Proposition 4.8. *Any independence system $(E, \mathcal{F})$ is the intersection if a finite number of matroids.*

However, the intersection of matroids is usually not a matroid, e.g. the set of matchings is not a matroid. We cannot hope for effiecient algorithms to intersect three matroids.

Theorem 4.9. *Finding a maximum independent set in the intersection of three matroids is NP-hard.*

Theorem 4.10. *Let $\mathcal{M}_1 = (E, \mathcal{F}_1)$ and $\mathcal{M}_2 = (E, \mathcal{F}_2)$ be matroids on E. Then*

$$\begin{aligned} \max\{|J| \ : \ J \in \mathcal{F}_1 \cap \mathcal{F}_2\} &= \min\{r_1(A) + r_2(\bar{A}) \ : \ A \subseteq E\} \\ (&= \min\{r_1(\bar{A}) + r_2(A) \ : \ A \subseteq E\}) \end{aligned}$$

4.2.1 Matroid Intersection Algorithm

The matroid intersection algorithm is motivated by the maximum cardinality matching algorithm applied to bipartite graphs. Given a set $J \in \mathcal{F}_1 \cap \mathcal{F}_2$, we want to augment this set. In bipartite matching we search for an alternating augmenting path. Here we do almost the same.

To this end, we construct an **auxiliary graph** $G = G(\mathcal{M}_1, \mathcal{M}_2, J)$ with $V(G) = E \dot\cup \{s, t\}$ and following edges

(1) (s, e) for all $e \in E \setminus J$ s.t. $J \cup \{e\} \in \mathcal{F}_1\}$,

(2) (e, t) for all $e \in E \setminus J$ s.t. $J \cup \{e\} \in \mathcal{F}_2\}$,

(3) (e, f) for all $e \in E \setminus J, f \in J$ s.t. $J \cup \{e\} \notin \mathcal{F}_2$ and $(J \cup \{e\}) \setminus \{f\} \in \mathcal{F}_2$,

(4) (f, e) for all $e \in E \setminus J, f \in J$ s.t. $J \cup \{e\} \notin \mathcal{F}_1$ and $(J \cup \{e\}) \setminus \{f\} \in \mathcal{F}_1$,

We refer to these edge sets as **type-(1)**, **type-(2)**, **type-(3)**, respectively **type-(4)** edges.

There is a symmetry in the sense that $G = G(\mathcal{M}_2, \mathcal{M}_1, J)$ is the same graph in which s and t are exchanged and all edges are reversed. We will use this fact in the proof of the augmenting path theorem for matroids:

Theorem 4.11. *a) If there is no s-t-dipath in G, then J is maximum. In fact, if $A \subseteq E$ and $\delta^+(A \cup \{s\}) = \emptyset$, then*

$$|J| = r_1(\bar{A}) + r_2(A).$$

b) If there exists an s-t-dipath in G, then J is not maximum.
In fact, if $s, e_1, f_1, e_2, f_2, \ldots, e_m, f_m, e_{m+1}, t$ is the sequence of a chordless (e.g. a shortest) s-t-dipath, then

$$J \triangle \{e_1, f_1, e_2, f_2, \ldots, e_m, f_m, e_{m+1}\} \in \mathcal{F}_1 \cap \mathcal{F}_2.$$

This theorem motivates Algorithm 8 for computing a maximum cardinality set in tht intersection of two matroids.

Algorithm 8 Matroid Cardinality Intersection Algorithm

Instance: Two matroids $\mathcal{M}_1 = (E, \mathcal{F}_1)$ and $\mathcal{M}_2 = (E, \mathcal{F}_2)$.
Output: A maximum cardinality set $J \in \mathcal{F}_1 \cap \mathcal{F}_2$ and a verifier $A \subseteq E$.

```
J = ∅
while true do
    Construct G = G(M_1, M_2, J);
    if there is an s-t-dipath in G then
        Let s, e_1, f_1, e_2, f_2, ..., e_m, f_m, e_{m+1}, t be the sequence of a shortest s-t-dipath in G;
        J := J △ {e_1, f_1, e_2, f_2, ..., e_m, f_m, e_{m+1}};
    else
        Let A := {e ∈ E : ∃ an s-e-dipath in G};
    end if
end while
```

Theorem 4.12. *The matroid cardinality intersection problem can be solved using $O(|E|^3)$ oracle calls.*

4.2.2 Matroid Constructions

Proposition 4.13. *(Restriction Matroid)*
Given a matroid $\mathcal{M} = (E, \mathcal{F})$ and $B \subseteq E$. Then $\mathcal{M}' = (E \setminus B, \{J \subseteq E \setminus B : J \in \mathcal{F}\})$ is a matroid (restricted to B). We write $\mathcal{M}' = \mathcal{M} \setminus B$.

Proposition 4.14. *(Disjoint Union Matroid)*
Given two matroids $\mathcal{M}_1 = (E_1, \mathcal{F}_1)$ and $\mathcal{M}_2 = (E_2, \mathcal{F}_2)$ where $E_1 \cap E_2 = \emptyset$. Then their disjoint union $\mathcal{M} = \mathcal{M}_1 \oplus \mathcal{M}_2 = (E, \mathcal{F})$ is a matroid, where $E = E_1 \dot\cup E_2$ and $\mathcal{F} = \{J_1 \cup J_2 : J_1 \in \mathcal{F}_1, J_2 \in \mathcal{F}_2\}$. The rank function of $\mathcal{M}$ is given by

$$r(A) = r_1(A \cap E_1) + r_2(A \cap E_2) \qquad (A \subseteq E).$$

Proposition 4.15. *(Partition Matroid)*
Assume you have a partition $E = E_1 \dot\cup E_2 \dot\cup \ldots \dot\cup E_m$ Define $\mathcal{F} := \{J \subseteq E : |J \cap E_i| \leq 1$ for $1 \leq i \leq m\}$. Then $\mathcal{M} = (E, \mathcal{F})$ is a matroid. The rank function of $\mathcal{M}$ is given by

$$r(A) = |\{i : A \cap E_i \neq \emptyset\}| \qquad (A \subseteq E).$$

Proposition 4.16. *(Contraction Matroid)*
Let $\mathcal{M} = (E, \mathcal{F})$ be a matroid and $B \subseteq E$. Choose a basis J of B and define $\mathcal{M}' = (E', \mathcal{F}')$ by $E' = E \setminus B$ and $\mathcal{F}' = \{J' \subset E' \ : \ J' \cup J \in \mathcal{F}\}$. Then $\mathcal{M}'$ is a matroid that does not depend on J, and its rank function r' is given by $r'(A) = r(A \cup B) - r(B)$, for all $A \subseteq E \setminus B$. We write $\mathcal{M}' = \mathcal{M}/B$.

Corollary 4.17.
Let $\mathcal{M} = (E, \mathcal{F})$ be a matroid and $B \subseteq E$. Then $\mathcal{M}' := (\mathcal{M} \setminus B) \oplus (\mathcal{M}/\bar{B})$ is a matroid on E. The bases of $\mathcal{M}'$ are exactly the bases of $\mathcal{M}$ that intersect B in a basis of B.

We can extend this idea to the following useful proposition.

Proposition 4.18. *Let $\mathcal{M} = (E, \mathcal{F})$ be a matroid. Let $\emptyset = T_0 \subseteq T_1 \subseteq T_2 \subseteq \cdots \subseteq T_{l+1} = E$. The bases of T_l in $\mathcal{M}$ that intersect T_i in a basis of T_i for $1 \leq i \leq l$ are the bases of T_l in the matroid $N := N_0 \oplus N_1 \oplus \cdots \oplus N_l$, where for each i, $N_i = (\mathcal{M}/T_i) \setminus \bar{T}_{i+1}$.*

4.2.3 Matroid Partitioning

Definition 4.19. *Let $\mathcal{M}_i = (E, \mathcal{F}_i) \quad (1 \leq i \leq k)$ be matroids. A set $J \subseteq E$ is* **partitionable** *with respect to $\mathcal{M}_1, \ldots, \mathcal{M}_k$ if $J = \bigcup_{1 \leq i \leq k} J_i$, where $J_i \in \mathcal{F}_i$ for $1 \leq i \leq k$. $\{J_1, \ldots, J_k\}$ is called a* **partition** *of J with respect to the $\mathcal{M}_1, \ldots, \mathcal{M}_k$.*

Remark 4.20. *By axiom (M2), we may assume that $J_i \cap J_j = \emptyset \quad (1 \leq i < j \leq k)$, i.e. they form a "real" partition $J = \dot{\bigcup}_{1 \leq i \leq k} J_i$.*

It turns out that matroid partition is equivalent to matroid intersection. To see this we need two propositions on matroid construction.

Theorem 4.21. *Let J be a maximum cardinality partitionable subset with respect to matroids $\mathcal{M}_i = (E, \mathcal{F}_i) \quad (1 \leq i \leq k)$. Then*

$$|J| = \min \left\{ \sum_{i=1}^{k} (r_i(A) + |\bar{A}| \ : \ A \subseteq E \right\}.$$

Theorem 4.22. *The subsets of E that are partitionable with respect to matroids $\mathcal{M}_1, \ldots, \mathcal{M}_k$ form the independent sets of a matroid. Its rank function is given by*

$$r(B) = \min\{|B \setminus A| + \sum_{i=1}^{k} r_i(A) \ : \ A \subseteq B\}.$$

4.3 Weighted Matroid Intersection

WEIGHTED MATROID INTERSECTION PROBLEM	
Input:	Two matroids $\mathcal{M}_1 = (E, \mathcal{F}_1)$ and $\mathcal{M}_2 = (E, \mathcal{F}_2)$, weights $c : E \to \mathbb{R}$.
Task:	Find a set $J \in \mathcal{F}_1 \cap \mathcal{F}_2$ whose weight $c(J)$ is maximum.

To solve the WEIGHTED MATROID INTERSECTION PROBLEM we extend to matroid cardinality intersection algorithm similarly to the to successive shortest path algorithm for transshipment, resp. weighted matching in bipartite graphs. In addition to transshipment/bipartite matching, the paths should have as few edges as possible (among all shortest paths).

Definition 4.23 (Auxiliary Graph 1)**.** *Given a $J \in \mathcal{F}_1 \cap \mathcal{F}_2$ we define an auxiliary graph $G = G(\mathcal{M}_1, \mathcal{M}_2, J, c)$ to be $G(\mathcal{M}_1, \mathcal{M}_2, J)$ (the graph from the cardinality intersection algorithm on page 39) with the following cost assignment $p : E(G) \to \mathbb{R}$ by*

(1) $p(s, e) = 0$ for all type-(1) arcs

(2) $p(e, t) = -c(e)$ for all type-(2) arcs

(3) $p(e, f) = c(f) - c(e)$ for all type-(3) arcs

(4) $p(f, e) = 0$ for all type-(4) arcs

Theorem 4.24. *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be matroids on E, let $c : E \to \mathbb{R}, k \in \mathbb{Z}$, let J be a maximum-weight common independent set of cardinality k, let P be a shortest path in $G(\mathcal{M}_1, \mathcal{M}_2, J, c)$ having as few arcs as possible, and let J' arise from J by augmenting with the vertices in $V(P) \cap E$. Then J' is a maximum-weight common independent set of cardinality $k + 1$.*

This theorem motivates the following algorithm for WEIGHTED MATROID INTERSECTION PROBLEM. However, we don't have a proof of Theorem 4.24 yet, it is not clear whether the

Algorithm 9 Primal Weighted Matroid Intersection Algorithm

Instance: Two matroids $\mathcal{M}_1 = (E, \mathcal{F}_1)$ and $\mathcal{M}_2 = (E, \mathcal{F}_2)$.
Output: A maximum cardinality set $J \in \mathcal{F}_1 \cap \mathcal{F}_2$ and a verifier $A \subseteq E$.

```
k := 0
J_0 := ∅
while true do
    Construct G = G(M_1, M_2, J_k, c);
    if there is an s-t-dipath in G then
        Find a shortest s-t-dipath P having as few arcs as possible;
        Augment J_k by P to obtain J_{k+1};
        k++;
    else
        Choose p s.t. c(J_p) ≥ c(J_i), 1 ≤ i ≤ k;
        return J_p;
    end if
end while
```

constructed J_i in the algorithm are common independent. Also the algorithm does not provide us a certificate of correctness.

To overcome these limitations and prove it's correctness we will develop a sophisticated primal-dual algorithm that provides certificates and takes essentially the same decisions as Algorithm 9, but provides certificates with which for which we can prove correctness.

Definition 4.25. *Let E be a finite set and $c : E \to \mathbb{R}$. The function-pair $c^1, c^2 : E \to \mathbb{R}$ is a* **weight-splitting** *for c if $c = c^1 + c^2$.*

Definition 4.26 (Auxiliary Graph 2)**.**
Let $J \in \mathcal{F}_1 \cap \mathcal{F}_2$ and let (c^1, c^2) be a a weight-splitting for c. Let

$$c_0^i := \max\{c^i(e) : e \notin J, J \cup \{e\} \in \mathcal{F}_i\} \text{ for } i \in \{1, 2\}.$$

If there is no maximizer, J is a maximum cardinality set and we are done. We define an new auxiliary graph $G = G(\mathcal{M}_1, \mathcal{M}_2, J, c^1, c^2)$ as $(\mathcal{M}_1, \mathcal{M}_2, J)$ with arc costs $p : E(G) \to \mathbb{R}$ defined by

(1) $p(s, e) = c_0^2 - c^2(e)$ for all type-(1) arcs

(2) $p(e, t) = c_0^1 - c^1(e)$ for all type-(2) arcs

(3) $p(e, f) = -c^1(e) + c^1(f)$ for all type-(3) arcs

(4) $p(f, e) = -c^2(e) + c^2(f)$ for all type-(4) arcs

Note that a path P is a shortest path in $G(\mathcal{M}_1, \mathcal{M}_2, J, c^1, c^2)$ if and only if it is a shortest path in $G(\mathcal{M}_1, \mathcal{M}_2, J, c)$: Let the vertices of P be ordered as $s, e_1, f_1, \ldots, e_m, f_m, e_{m+1}, t$ and J, J' be the common sets before and after the augmentation, then

$$\begin{aligned} p(P) &= (c_0^2 - c^2(e_1)) + (-c^1(e_1) + c^1(f_1)) + (-c^2(e_2) + c^2(f_1)) + \ldots \\ &\qquad +(-c^2(e_{m+1}) + c^2(f_m)) + (c_0^1 - c^2(e_{m+1})) \\ &= c_0^1 + c_0^2 + c(J) - c(J'). \end{aligned}$$

So the difference of in the cost $p(P)$ between the two graph models is the constant $c_0^1 + c_0^2$.

Definition 4.27. *We say a weight-splitting (c^1, c^2)* **certifies** *a common independent set J if, for $i = 1, 2$, J has maximum c^i-weight over all independent sets in $\mathcal{F}_i$ with cardinality $|J|$.*

So if (c^1, c^2) **certifies** J, then J is a maximum-weight common independent set of cardinality $|J|$ with respect to c.

Proposition 4.28. *A weight-splitting (c^1, c^2)* **certifies** *a common independent set J if and only if*

a) The arc costs of G are all nonnegative and

b) For $i = 1, 2$, $f \in J$ implies $c_0^i \leq c^i(f)$.

Proof. We will use the following simple fact:
Given a matroid $\mathcal{M} = (E, \mathcal{F})$ and $k \in \mathbb{N}$. Define $\mathcal{F}' = \{J \in \mathcal{F} \ : \ |J| \leq k\}$.
Then $\mathcal{M}' = (E, \mathcal{F}')$ is a matroid, obtained from $\mathcal{M}$ by **truncation** *to rank k.*

Now let $k := |J|$. A weight-splitting (c^1, c^2) certifies J if and only if, for $i = 1$ and 2, J is a c^i-optimum basis of $\mathcal{M}'_i$, which is obtainted from $\mathcal{M}$ by truncating it to rank k. By Corollary 4.6 (final statement) this holds if and only if, for $i = 1, 2$,

$$e \notin J, f \in J, (J \cup \{e\}) \setminus \{f\} \in \mathcal{F}_i \text{ implies } c^i(e) \leq c^i(f). \tag{4.3}$$

Consider (4.3) for elements e, f such that $J \cup \{e\} \notin \mathcal{F}_i$. If $i = 2$, then (f, e) is a type-(4) arc and if $i = 1$, then (e, f) is a type-(3) arc. In either case (4.3) is equivalent to condition a).

Now consider (4.3) for elements e, f such that $J \cup \{e\} \in \mathcal{F}_i$. If $i = 1$, then (s, e) is a type-(1) arc and if $i = 2$, then (e, t) is a type-(2) arc. In either case (4.3) is equivalent to condition b). □

Proposition 4.29 (Frank [1981])**.**
Let $(E, \mathcal{F})$ be a matroid, $c : E \to \mathbb{R}$ and $J \in \mathcal{F}$. Let $\{f_1, \dots, f_l\} \in J$ and $\{e_1, \dots, e_l\} \notin J$ with

1. *$J \cup \{e_i\} \notin \mathcal{F}, (J \cup \{e_i\}) \setminus \{f_i\} \in \mathcal{F}$ and $c(f_i) = c(e_i)$ for all $i = 1, \dots l$, and*
2. *$J \cup \{e_j\} \notin \mathcal{F}, (J \cup \{e_j\}) \setminus \{f_i\} \in \mathcal{F}$ or $c(f_i) > c(e_j)$ for $1 \leq i \neq j \leq l$.*

Then $(J \setminus \{f_1, \dots, f_l\}) \cup \{e_1, \dots, e_l\} \in \mathcal{F}$.

Proof. The proposition is proven by induction on l. For l=1 the statement follows from the first assumption. Let $l > 1, h := \arg\min_{1 \leq i \leq l} c(f_i)$, $\mu := c(f_h)$, and $J' := (J \setminus \{f_h\}) \cup \{e_h\}$. By the first assumption we have $J' \in \mathcal{F}$.

For $X \in \mathcal{F}$, let $C(X, y)$ denote the unique $\mathcal{F}$-circuit if $X \cup \{y\} \notin \mathcal{F}$ and the empty set if $X \cup \{y\} \in \mathcal{F}$. We claim:

$$C(J', e_j) = C(J, e_j) \text{ for all } j \neq h.$$

If so, the two assumptions of the statement hold and we can apply induction hypothesis.

Let $j \neq h$. Assume $C(J', e_j) \neq C(J, e_j)$. Then we have $f_h \in C(J, e_j)$. But by the second and first assumption $\mu = c(f_h) > c(e_j) = c(f_j) \geq \mu$, a contradiction.

□

Lemma 4.30. *After a change in the weight-splitting of Algorithm 10, the weight-splitting still certifies J_k.*

Proof. The initial weight-splitting $c^1 = c, c^2 = 0$ is obviously certifying $J = \emptyset$. Consider the first iteration where the claim fails. Let c^1, c^2 denote the old weight-splitting, let p denote the old arc costs in G, and let p' denote the new arc costs. We will show that the new weight-splitting fulfills the conditions of Proposition 4.28.

Suppose that (e, f) is a type-(3) arc, i.e. $e \notin J$. Then

$$p'(e, f) = -(c^1(e) - \sigma_e) + (c^1(f) - \sigma_f) = p(e, f) + \sigma_e - \sigma_f.$$

Algorithm 10 Primal-Dual Weighted Matroid Intersection Algorithm

Instance: Two matroids $\mathcal{M}_1 = (E, \mathcal{F}_1)$ and $\mathcal{M}_2 = (E, \mathcal{F}_2)$.
Output: A maximum cardinality set $J \in \mathcal{F}_1 \cap \mathcal{F}_2$ and a verifier $A \subseteq E$.

```
k := 0;
J_0 := ∅;
c^1 = c; c^2 = 0;
while J_k is neither an M_1-basis nor an M_2-basis do
    ADJUST WEIGHT-SPLITTING ("DUAL ADJUST"):
    Construct G = G(M_1, M_2, J_k, c^1, c^2);
    Find shortest path tree rooted in s with distance d(v) for each v ∈ V(G);
    for v ∈ E do
        σ_v := min{d_v, d_t};
        c^1(v) := c^1(v) − σ_v;
        c^2(v) := c^2(v) + σ_v;
    end for
    AUGMENT ("PRIMAL ADJUST"):
    Construct G = G(M_1, M_2, J_k, c^1, c^2);
    if there is an s-t-dipath in G then
        Find shortest s-t-dipath P having as few arcs as possible;
        Augment J_k by P to obtain J_{k+1};
        k++;
    else
        Choose p s.t. c(J_p) ≥ c(J_i), 1 ≤ i ≤ k;
        return J_p;
    end if
end while
```

If $\sigma_e = d(t)$, then $p'(e,f) \geq p(e,f) + d(t) - d(t) = p(e,f) \geq 0$. Otherwise, $\sigma_e = d(e)$ and $p'(e,f) \geq p(e,f) + d(e) - d(f) \geq 0$, because the d-values are shortest path labels with respect to p.

The proof for type-(4) arcs is similar. By definition type-(1) and type-(2) arcs have nonnegative costs. Altogether, a) in Proposition 4.28 is satisfied.

Now suppose that condition b) in the Proposition is violated, i.e. there exist $f \in J_k, e \notin J_k$ such that $J_k \cup \{e\} \in \mathcal{F}_1$ and $c^1(e) - \sigma_e > c^1(f) - \sigma_f$ (the "new c_0^1" is greater than $c^1(f) - \sigma_f$). As $c^1(e) \leq c_0^1 \leq c^1(f)$, we must have $\sigma_f > \sigma_e$ and, thus, $\sigma_e = d(e)$ (otherwise $\sigma_e = d(t) \geq \sigma_f$). Therefore,

$$c^1(e) - d(e) > c^1(f) - \sigma_f \geq c^1(f) - d(t) \geq c^1(f) - (d(e) + c_0^1 - c^1(e)).$$

We conclude $c^1(f) < c_o^1$, a contradiction.

Observe that by construction type-(1) arcs will always have cost zero, because c^2 is initialized as $c^2 = 0$ and direct successors e of s have distance $d(e) = 0$ and $\sigma_e = 0$ throughout the algorithm. □

Lemma 4.31. *In Algorithm 10, every arc of P has cost zero.*

Proof. Let P' be the shortest s-t-dipath with respect to p dertermined by c^1, c^2 before the change of the weight-splitting. Then for any $(u,v) \in E(P')$ we have $d(u) + p(u,v) = d(v)$. Thus, $d(v) \leq d(t)$ for all $v \in V(P')$. Now consider a type-(3) arc (e,f) of P'. Then the new cost is $p'(e,f) = p(e,f) - (-\sigma_e) + (-\sigma_f) = p(e,f) + d(e) - d(f) = 0$. Similarly, for a type-(4) arc (f,e) of P' we get $p'(f,e) = p(f,e) - d(e) + d(f) = 0$.

Now, for a type-(2) arc (e,t) (not necessarily on P'), consider

$$c^1(e) - \sigma_e \leq c^1(e) - d(e) \leq c^1(e) - d(t) + p(e,t) = c_0^1 - d(t). \tag{4.4}$$

If $e^\star$ is the second last node of P', then $d(t) \geq d(e)$, so $\sigma_e = d(e)$ and $d(e) + p(e,t) = d(t)$. Thus, both inequlities in (4.4) hold with equality.

Now among all $e \in E$ such that (e,t) is an arc in G, $c^1(e) - \sigma_e$ is maximized by $e^\star$, which, thus, will determine the new value of c_0^1. Therefore, $p'(e^\star, t) = 0$.

Type-(1) arcs have cost zero throughout the course of the algorithm. It follows that every shortest path with respect to p' has zero length (one example is P') and as $p' \geq 0$, every arc has length zero. □

Lemma 4.32. *In Algorithm 10, if $J_k \in \mathcal{F}_1 \cap \mathcal{F}_2$, then $J_{k+1} \in \mathcal{F}_1 \cap \mathcal{F}_2$.*

Proof. Let P be the path found by the algorithm with vertices $s, e_1, f_1, \ldots, e_m, f_m, e_{m+1}, t$. We show that $\mathcal{M}_1, J_k \cup \{e_1\}, \{f_1, \ldots, f_m\}$, and $\{e_2, \ldots, e_{m+1}\}$ satisfy the requirements of Proposition 4.29. First note that $J_k \cup \{e_1\} \in \mathcal{F}_1$ as $(s, e_1) \in E(G)$. The first condition in Proposition 4.29 follows from the fact that every arc in P has length zero. The second condition follows from the fact that P has fewest edges and weigths are nonnegative. Thus by Proposition 4.29 $J_{k+1} \in \mathcal{F}_1$. Similarly, it can be shown that $J_{k+1} \in \mathcal{F}_2$ by applying the proposition to $\mathcal{M}_1$, $J_k \cup \{e_{m+1}\}, \{f_1, \ldots, f_m\}$, and $\{e_1, \ldots, e_m\}$ □

Lemma 4.33. *After an augmentation in Algorithm 10, the weight-splitting certifies J_{k+1}.*

Proof. Let P be the path found in the AUGMENT-step of Algorithm 10. We show that J_{k+1} is c^1-optimal among all common independent sets of cardinality $k+1$. Let e_{m+1} be the second last node of P. Then by Lemma 4.31 $c^1(e_{m+1}) = c_0^1$, so e_{m+1} is the heaviest element that can be added freely to J_k such that $J_k \cup \{e_{m+1}\} \in \mathcal{F}_2$. Thus, $J_k \cup \{e_{m+1}\}$ is c^1-optimal of cardinality $k+1$ in $\mathcal{F}_2$.

Now because each arc on P has cost zero each element of J_k on P has the same c_1-weight as its immediate predecessor (that is not in J_k). Therefore, J_{k+1} has the same c^1-weight as $J_k \cup \{e_{m+1}\}$ and it is c^1-optimal of cardinality $k+1$ in $\mathcal{F}_2$.

The c^2-optimality follows similarly. □

This completes the proof of Theorem 4.24 and shows that Algorithms 9 and 10 work correctly. Both algorithms take at most $|E|$ iterations, which in turn are dominated by the graph construction and shortest path search. For Algorithm 9 this takes $O(|E|^2 \log |E|)$ time, while for Algorithm 10 this can be accomplished in $O(|E|^2)$ time, because the shortest path with fewest edges can be found by breadth-first search using zero-cost arcs only. We conclude.

Theorem 4.34. *The* WEIGHTED MATROID INTERSECTION PROBLEM *can be solved in time $O(|E|^3\theta)$, where θ is the time for calling the independent set oracle.*

4.4 Polymatroids

The rank-function r of a matroid $\mathcal{M} = (E, \mathcal{F})$ is submodular, i.e.

$$r(X) + r(Y) \geq r(X \cap Y) + r(X \cup Y) \quad \text{for all } X, Y \subseteq E.$$

In Theorem 4.5, we have seen that incedence vectors of independent sets in $\mathcal{F}$ are corners of the matroid polytope (4.1):

$$\left\{ x \in \mathbb{R}^E \ : \ x \geq 0, \sum_{e \in A} x_e \leq r(A) \text{ for all } A \subseteq E \right\}.$$

Polymatroids generalize matroid-polytopes.

Definition 4.35. *A* **polymatroid** *is a polytope*

$$P(f) := \left\{ x \in \mathbb{R}^E \ : \ x \geq 0, \sum_{e \in A} x_e \leq f(A) \textit{ for all } A \subseteq E \right\},$$

where E is a finite set and $f : 2^E \to \mathbb{R}_+$ is a submodular function.

For a matroid rank-function r the polymatroid $P(r)$ is its matroid-polytope.

Proposition 4.36. *For any polymatroid f can be chosen such that $f(\emptyset) = 0$ and f is monotone, i.e. $f(X) \leq f(Y)$ for all $X \subseteq Y \subseteq E$*

Proof. See exercise sheets. □

Thus the defining function f for a polymatroid satisfies two central characteristics of a matroid rank-function, i.e. submodularity and monotonicity, but the third characteristic $|X| \geq f(X)$ may be violated. It turns out that optimization over a polymatroid can be done by a similar greedy algorithm (Algorithm 11).

Proposition 4.37. *Let $E = \{e_1, \dots, e_n\}$ and $f : 2^E \to \mathbb{R}$ be a submodular function with $f(\emptyset) \geq 0$. Let $b : E \to \mathbb{R}$ with $b(e_1) \leq f(\{e_1\})$ and $b(e_i) \leq f(\{e_1, \dots, e_i\}) - f(\{e_1, \dots, e_{i-1}\})$ for $i = 2, \dots, n$. Then*

$$\sum_{a \in A} b(a) \leq f(A) \quad \textit{for } A \subseteq E.$$

Proof. We prove the statement by induction in $i = \max\{j \ : \ e_j \in A\}$. The assertion is trivial for $A = \emptyset$. Let $i \geq 1$. Then

$$\begin{array}{rcl} \sum_{a \in A} b(a) & = & \sum_{a \in A \setminus \{e_i\}} b(a) + b(e_i) \\ & \leq & f(A \setminus \{e_i\}) + b(e_i) \\ & \leq & f(A \setminus \{e_i\}) + f(\{e_1, \dots, e_i\}) - f(\{e_1, \dots, e_{i-1}\}) \\ & \leq & f(A), \end{array}$$

where the inequalities follow from the induction hypothesis, the choice of b, and the submodularity of f (with $X = A \ni e_i$ and $Y = \{e_1, \dots, e_{i-1}\}$). □

Algorithm 11 Polymatroid Greedy Algorithm

Instance: A finite set E and a submodular, monotone function $f : 2^E \to \mathbb{R}_+$ with $f(\emptyset) \geq 0$ (given by an oracle), and a vector $c \in \mathbb{R}^E$

Output: A vector $x \in P(f)$ with $c^\intercal x$ maximum.

Sort $E = \{e_1, \dots, e_n\}$ such that $c(e_1) \geq \cdots \geq c(e_k) > 0 \geq c(e_{k+1}) \geq \cdots \geq c(e_n)$.
if $k \geq 1$ **then**
  $x(e_1) := f(\{e_1\})$;
end if
for $i = 2, \dots, k$ **do**
  $x(e_i) := f(\{e_1, \dots, e_i\}) - f(\{e_1, \dots, e_{i-1}\})$;
end for
for $i = k+1, \dots, n$ **do**
  $x(e_i) := 0$;
end for

Theorem 4.38. *The Polymatroid Greedy Algorithm correctly finds an $x \in P(f)$ with $c^\intercal x$ maximum. If f is integral, then x is integral.*

Proof. Let x be the output of the Polymatroid Greedy Algotrithm for E, f, and c. By construction, if f is integral so is x. As f is monotone, $x \geq 0$, and by Proposition 4.37 $x \in P(f)$.

Now assume there is a $y \in \mathbb{R}_+^E$ with $c^\intercal y > c^\intercal x$. Similar to the *Dual Greedy Algorithm* in the proof of Theorem 4.5 we set $d_j := c(e_j) - c(e_{j+1})$ $(j = 1, \dots, k-1)$, $d_k := c(e_k)$, and obtain:

$$\sum_{j=1}^{k} d_j \sum_{i=1}^{j} x(e_j) = c^\intercal x < c^\intercal y \leq \sum_{j=1}^{k} c(e_j) y(e_j) = \sum_{j=1}^{k} d_j \sum_{i=1}^{j} y(e_j). \tag{4.5}$$

As $d \geq 0$, there exists an index $j \in \{1, \dots, k\}$ with $\sum_{i=1}^{j} y(e_i) > \sum_{i=1}^{j} x(e_i)$. But as $\sum_{i=1}^{j} x(e_i) = f(\{e_1, \dots, e_j\})$ this means that $y \notin P(f)$. □

The following theorem on the intersection of two polymatroids allows us to derive several interesting results.

Theorem 4.39. *(Edmonds [1970,1979]) Let E be a finite set and $f, g : 2^E \to \mathbb{R}_+$ two submodular functions. Then the following system of inequalities is TDI:*

$$\begin{aligned} \textstyle\sum_{e \in A} x_e &\leq f(A) \quad (A \subseteq E), \\ \textstyle\sum_{e \in A} x_e &\leq g(A) \quad (A \subseteq E), \\ x &\geq 0. \end{aligned}$$

Corollary 4.40. *(Edmonds [1970]) Let $(E, \mathcal{F}_1)$ and $(E, \mathcal{F}_2)$ be two matroids with rank functions r_1 and r_2. Then the convex hull of the incedence vectors of the elements of $\mathcal{F}_1 \cap \mathcal{F}_2$ is the polytope*

$$\left\{ x \in \mathbb{R}_+^E \ : \ \sum_{e \in A} x_e \leq \min\{r_1(A), r_2(A)\} \text{ for all } A \subseteq E \right\}.$$

Proof. By Theorem 4.39 the system is TDI, and since rank functions are integral the polytope is integral (TDI-systems with integral right hand side are integral).

Since $r_1(A) \leq |A|$ for all $A \subset E$, the vertices are 0-1-vectors, and thus incidence vectors of common independent sets, i.e. representing elements in $\mathcal{F}_1 \cap \mathcal{F}_2$.

Vice versa, each incedence vector of an element in $\mathcal{F}_1 \cap \mathcal{F}_2$ satisfies the inequalities of the polytope. □

Corollary 4.41. *(Edmonds [1970]) Let E be a finite set, and $f, g : 2^E \to \mathbb{R}_+$ be submodular and monotone functions with $f(\emptyset) = g(\emptyset) = 0$. Then*

$$\max\{\mathbb{1}^\intercal x \ : \ x \in P(f) \cap P(g)\} = \min_{A \subseteq E}(f(A) + g(E \setminus A)).$$

Furthermore, if f and g are integral there exists an integral maximum solution x.

Proof. Consider the dual to

$$\max\{\mathbb{1}^\intercal x \ : \ x \in P(f) \cap P(g)\},$$

which is

$$\min \left\{ \sum_{A \subseteq E} (f(A)y_A + g(A)z_F) \ : \ y, z \geq 0, \sum_{A \subseteq E : e \in A} (y_A + z_A) \geq 1 \text{ for all } e \in E \right\}.$$

By Theorem 4.39 it has an integral solution y, z. Let

$$B := \bigcup_{A : y_A \geq 1} A \text{ and } C := \bigcup_{A : z_A \geq 1} A$$

By integrality and the inequalities for each $e \in E$ in the dual, we have $B \cup C = E$. Since f and g are submodular, and $f(\emptyset) = g(\emptyset) = 0$,

$$\sum_{A \subseteq E} (f(A)y_A + g(A)z_A) \geq f(B) + g(C) \geq f(B) + g(E \setminus B),$$

where the second inequality follows from $E \setminus B \subseteq C$ and the monotonicity of g, proving "$\geq$".

The inequality "$\leq$" is trivial, because for any $A \in E$, we obtain a feasible solution by setting $y_A := 1$ and $z_{E \setminus A} := 1$ and all other components to zero. □

As a special case we get Theorem 4.10.

Corollary 4.42. *(Frank's Discrete Sandwich Theorem [1982]) Let E be a finite set and $f, g : 2^E \to \mathbb{R}$ such that f is supermodular, g is submodular, and $f \leq g$. Then there exists a modular function $h : 2^E \to \mathbb{R}$ with $f \leq h \leq g$. If f and g are integral, h can also be chosen integral.*

Proof. We may assume that $f(\emptyset) = g(\emptyset)$ and $f(E) = g(E)$.

Let $M := 2\max\{|f(A)| + |g(A)| \ : \ A \subseteq E\}$. Let $f'(A) := g(E) - f(E \setminus A) + M|A|$ and $g'(A) := g(A) - f(\emptyset) + M|A|$ for all $A \subseteq E$. By construction, f' and g' are nonnegative, submodular, monotone, and $f'(\emptyset) = g'(\emptyset) = 0$. Corollary 4.41 yields

$$\begin{aligned} & \max\{\mathbb{1}^\intercal x \ : \ x \in P(f') \cap P(g')\} \\ = \ & \min_{A \subseteq E}(f'(A) + g'(E \setminus A)) \\ = \ & \min_{A \subseteq E}(g(E) - f(E \setminus A) + M|A| + g(E \setminus A) - f(\emptyset) + M|E \setminus A|\} \\ \geq \ & g(E) - f(\emptyset) + M|E|. \end{aligned}$$

So let $x \in P(f') \cap P(g')$ with $\mathbb{1}^\intercal x = g(E) - f(\emptyset) + M|E|$. If f and g are integral, x can be chosen integral. Define $h'(A) := \sum_{e \in A} x_e$ and $h(A) := h'(A) + f(\emptyset) - M|A|$ for all $A \subseteq E$. Then h is modular. Moreover, for all $A \subseteq E$, we have $h(A) \leq g'(A) + f(\emptyset) - M|A| = g(A)$ and $h(A) = \mathbb{1}^\intercal x - h'(E \setminus A) + f(\emptyset) - M|A| \geq g(E) + M|E| - M|A| - f'(E \setminus A) = f(A)$. □

4.5 Greedoids

Greedoids are a generalization of matroids, where we are dropping the subclusiveness axoim (M2). Thus, a greedoid is not an independence system. They were introduced by Korte and Lovasz in 1980. We will see that generally optimizing over greedoids is NP-hard. But we can specify certain necessary and sufficient conditions on the objective functions so that a greedy algorithms works.

Definition 4.43. *(Greedoid) A set system $(E, \mathcal{F})$ is* **accessible** *if $\emptyset \in \mathcal{F}$ and for any $X \in \mathcal{F} \setminus \{\emptyset\}$ there exists an $x \in X$ with $X \setminus \{x\} \in \mathcal{F}$.*

A **greedoid** *is a set system $(E, \mathcal{F})$ satisying (M1) and (M3).*

Note that Greedoids are accessible.

Theorem 4.44. *Let $(E, \mathcal{F})$ be an accessible set system. The following statements are equivalent:*

1. *For any $X \subseteq Y \subset E$ and $z \in E \setminus Y$ with $X \cup \{z\} \in \mathcal{F}$ and $Y \in \mathcal{F}$ we have $Y \cup \{z\} \in \mathcal{F}$.*
2. *$\mathcal{F}$ is closed under union.*

Definition 4.45. *An accessible set system $(E, \mathcal{F})$ that is closed under union is called an antimatroid.*

Proposition 4.46. *Every antimatroid is a greedoid.*

Proposition 4.47. *Let $(E, \mathcal{F})$ be a set system such that $\mathcal{F}$ is closed under union and $\emptyset \in \mathcal{F}$. Define*

$$\tau(A) := \bigcap \{X \subseteq E \ : \ A \subseteq X, E \setminus X \in \mathcal{F}\}.$$

Then τ is a closure operator, i.e. τ satisfies (S1)-(S3).

Theorem 4.48. *Let $(E, \mathcal{F})$ be a set system such that $\mathcal{F}$ is closed under union and $\emptyset \in \mathcal{F}$. Then $(E, \mathcal{F})$ is accessible if and only if the closure operator of Proposition 4.47 satisfies the anti-exchange property: if $X \subseteq E, y, z, \in E \setminus \tau(X), y \neq z$ and $z \in \tau(X \cup \{y\})$, then $y \notin \tau(X \cup \{z\})$.*

4.6 Submodular Function Maximization

Let $f : 2^E \to \mathbb{R}$ be a submodular function. Submodularity can equivalently be characterized as

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y) \tag{4.6}$$

for all $X \subseteq Y \subseteq E$ and $x \in E$, capturing the principle of *diminishing returns* in economics.

Given a non-negative submodular function, the *unconstrained submodular function maximization problem* (USM) is to find a subset $S \subset E$ that maximizes $f(S)$. USM covers many well studied optimization problems such as Max-Cut, Max-DiCut that are known to be *NP*-hard.

We present a recent radnomized 1/2-approximation algorithms by [1] that is using the so called double-greedy technique. We assume that the $n := |E|$ elements of E are given in an arbitrary order $E = \{e_1, e_2, \ldots, e_n\}$.

4.6.1 Deterministic USM

We first present a deterministic $1/3$-approximation algorithm. It maintains two sets X and Y, which are initialized as the empty set $X = \emptyset$, and as the universe $Y = E$. In each iteration $1 \leq i \leq n$ either e_i is added to X or it is removed from Y so that after n iterations $X = Y$.

Algorithm 12 Deterministic Double Greedy Algorithm for USM

Instance: A finite set E and a submodular function $f : 2^E \to \mathbb{R}_+$.
Output: A set $S \subseteq E$

$X_0 \leftarrow \emptyset, Y_0 \leftarrow E$
for $i = 1 \to n$ **do**
  $a_i \leftarrow f(X_{i-1} \cup \{e_i\}) - f(X_{i-1})$;
  $b_i \leftarrow f(Y_{i-1} \setminus \{e_i\}) - f(Y_{i-1})$;
  if $a_i \geq b_i$ **then**
    $X_i \leftarrow X_{i-1} \cup \{e_i\}, Y_i \leftarrow Y_{i-1}$
  else
    $X_i \leftarrow X_{i-1}, Y_i \leftarrow Y_{i-1} \setminus \{e_i\}$
  end if
end for
return $S \leftarrow X_n (= Y_n)$;

Lemma 4.49. *For every* $1 \leq i \leq n$, $a_i + b_i \geq 0$.

Proof. This follows from submodularity as follows. First note that $(X_{i-1} \cup \{e_i\}) \cup (Y_i \setminus \{e_i\}) = Y_{i-1}$ and $(X_{i-1} \cup \{e_i\}) \cap (Y_i \setminus \{e_i\}) = X_{i-1}$. Thus,

$$\begin{aligned} a_i + b_i &= (f(X_{i-1} \cup \{e_i\}) - f(X_{i-1})) + (f(Y_{i-1} \setminus \{e_i\}) - f(Y_{i-1})) \\ &= (f(X_{i-1} \cup \{e_i\}) + f(Y_{i-1} \setminus \{e_i\})) - (f(X_{i-1}) - f(Y_{i-1})) \\ &\geq 0, \end{aligned}$$

where the inequality follows from the submodularity (4.6). □

Let OPT denote an optimum solution and define $OPT_i := (OPT \cup X_i) \cap Y_i$ $(1 \leq i \leq n)$. Thus, OPT_i coincides with X_i, Y_i on the elements $1, \ldots, i$, and it coincides with OPT on the elements $i+1, \ldots, n$. In particular $OPT_0 = OPT$ and $OPT_n = X_n = Y_n$.

Lemma 4.50. *For every* $1 \leq i \leq n$

$$f(OPT_{i-1}) - f(OPT_i) \leq (f(X_i) - f(X_{i-1})) + (f(Y_i) - f(Y_{i-1}))\,.$$

Proof. W.l.o.g $a_i \geq b_i$, i.e. $X_i \leftarrow X_{i-1} \cup \{e_i\}$ and $Y_i \leftarrow Y_{i-1}$ (the other case is proven analogously). Then $OPT_i = OPT_{i-1} \cup \{e_i\}$. Thus, we need to prove

$$f(OPT_{i-1}) - f(OPT_i) \leq f(X_i) - f(X_{i-1}) = a_i.$$

If $e_i \in OPT$, the left hand side of the last inequality is 0 and $a_i \geq 0$ since $a_i + b_i \geq 0$ by Lemma 4.49.

Otherwise, if $e_i \notin OPT$, then $e_i \notin OPT_{i-1}$. Thus, $OPT_{i-1} = ((OPT \cup X_{i-1}) \cap Y_{i-1}) \subseteq Y_{i-1} \setminus \{e_i\}$. Applying submodularity we get:

$$f(OPT_{i-1}) - f(OPT_{i-1} \cup \{e_i\}) \leq f(Y_{i-1} \setminus \{e_i\}) - f(Y_{i-1}) = b_i \leq a_i.$$

□

This allows us to prove the main result of the deterministic algorithm.

Theorem 4.51 (Buchbinder et al. 2012)**.** *Algorithm 12 returns a* $1/3$*-approximation for the unconstrained submodular function maximization problem.*

Proof. By Lemma 4.50 we get.

$$\sum_{i=1}^{n} (f(OPT_{i-1}) - f(OPT_i)) \leq \sum_{i=1}^{n} (f(X_i) - f(X_{i-1})) + (f(Y_i) - f(Y_{i-1}))\,.$$

Collapsing the telescopic sum we get:

$$f(OPT_0) - f(OPT_n) \leq (f(X_n) - f(X_0)) + (f(Y_n) - f(Y_0)) \leq f(X_n) + f(Y_n).$$

As $OPT_0 = OPT$ and $OPT_n = X_n$, we get $f(X_n) \geq \frac{1}{3} OPT$. □

Remark 4.52. *Buchbinder et al. [1] gave an example, for which Algorithm 12 provides only a* $(1/3 + \epsilon)$*-approximation. Thus, the analysis is best possible.*

Algorithm 13 Randomized Double Greedy Algorithm for USM

Instance: A finite set E and a submodular function $f : 2^E \to \mathbb{R}_+$.
Output: A set $S \subseteq E$

$X_0 \leftarrow \emptyset, Y_0 \leftarrow E$
for $i = 1 \to n$ **do**
  $a_i \leftarrow f(X_{i-1} \cup \{e_i\}) - f(X_{i-1})$;
  $b_i \leftarrow f(Y_{i-1} \setminus \{e_i\}) - f(Y_{i-1})$;
  $a'_i \leftarrow \max\{a_i, 0\}$;
  $b'_i \leftarrow \max\{b_i, 0\}$;
  $p = \begin{cases} a'_i/(a'_i + b'_i) & \text{if } (a'_i + b'_i) > 0, \\ 1 & \text{if } (a'_i + b'_i) = 0; \end{cases}$
  With probability p **do:**
  $X_i \leftarrow X_{i-1} \cup \{e_i\}, Y_i \leftarrow Y_{i-1}$
  Else:
  $X_i \leftarrow X_{i-1}, Y_i \leftarrow Y_{i-1} \setminus \{e_i\}$
end for
return $S \leftarrow X_n (= Y_n)$;

4.6.2 Randomized USM

Replacing the deterministic choices in Algorithm 12, we can achieve a $1/2$-factor approximation (in expectation). Algorithm and analysis are similar to the deterministic one.

Lemma 4.50 is now replaced by the following variant.

Lemma 4.53. *For every* $1 \leq i \leq n$,

$$\mathbb{E}\left[f(OPT_{i-1}) - f(OPT_i)\right] \leq \frac{1}{2}\mathbb{E}\left[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})\right], \tag{4.7}$$

where the expectations are taken over the random choices of the algorithm.

Proof. Note that it suffice to prove (4.7) conditioned on any event of the form $X_{i-1} = S_{i-1}$, when $S_{i-1} \subseteq \{e_1, \ldots, e_{i-1}\}$ and the probabolity that $X_{i-1} = S_{i-1}$ is non-zero.

Hence, we fix such an event S_{i-1}. The remainder of this proof assumes that everything is conditioned on this event. Then, the following quantities become constants: $Y_{i-1} = S_{i-1} \cup \{e_i, \ldots, e_n\}, OPT_{i-1} = (OPT \cup X_{i-1}) \cap Y_{i-1}$, a_i, and b_i. By Lemma 4.49: $a_i + b_i \geq 0$. Hence, it suffices to consider the following three cases.

Case 1 ($a_i \geq 0$ and $b_i \leq 0$): Then $p = 1$, and $Y_i = Y_{i-1} = S_{i-1} \cup \{e_i, \ldots, e_n\}$ and $X_i \leftarrow S_i \cup \{e_i\}$. Thus, $f(Y_i) - f(Y_{i-1}) = 0$ and $OPT_i = (OPT \cup X_i) \cap Y_i = OPT_{i-1} \cup \{e_i\}$. We claim that

$$f(OPT_{i-1}) - f(OPT_{i-1} \cup \{e_i\}) \leq \frac{1}{2}\left(f(X_i) - f(X_{i-1})\right) = \frac{a_i}{2}.$$

If $e_i \in OPT$, $f(OPT_{i-1}) - f(OPT_i \cup \{e_i\}) = 0 \leq \frac{a_i}{2}$. If $e_i \notin OPT$, then $OPT_i \subseteq Y_i \setminus \{e_i\}$ and by the submodularity of f: $f(OPT_{i-1}) - f(OPT_{i-1} \cup \{e_i\}) \leq f(Y_{i-1} \setminus \{e_i\}) - f(Y_{i-1}) =$

$b_i \leq 0 \leq \frac{a_i}{2}$, where the last inequality follows from the current case condition ($a_i \geq 0$ and $b_i \leq 0$).

Case 2 ($a_i < 0$ and $b_i > 0$): Then $p = 0$ and the case is analoguos to Case 1.

Case 3 ($a_i \geq 0$ and $b_i \geq 0$): Now $a'_i = a'_i$ and $b'_i = b_i$. With probabiliy $p = a_i/(a_i + b_i)$, $X_i \leftarrow X_{i-1} \cup \{e_i\}$ and $Y_i = Y_{i-1}$, and with probability $(1-p) = b_i/(a_i + b_i)$, $X_i \leftarrow X_{i-1}$ and $Y_i = Y_{i-1} \setminus \{e_i\}$. Thus

$$\begin{aligned} \mathbb{E}[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})] &= p\,(f(X_{i-1} \cup \{e_i\}) - f(X_{i-1})) + \\ &\quad (1-p)\,(f(Y_{i-1} \setminus \{e_i\}) - f(Y_{i-1})]) \\ &= \tfrac{a_i^2 + b_i^2}{a_i + b_i}. \end{aligned} \tag{4.8}$$

Now we upper bound the left hand side of (4.7). Recall, $OPT_i = (OPT \cup X_i) \cap Y_i$ and, thus,

$$\begin{aligned} \mathbb{E}[f(OPT_{i-1}) - f(OPT_i)] &= p\,(f(OPT_{i-1}) - f(OPT_{i-1} \cup \{e_i\})) + \\ &\quad (1-p)\,(f(OPT_{i-1}) - f(OPT_{i-1} \setminus \{e_i\})) + \\ &\leq \tfrac{a_i b_i}{a_i + b_i}. \end{aligned} \tag{4.9}$$

To see the final inequality, note that $e_i \in Y_{i-1}$ and $e_i \notin X_{i-1}$. We consider two cases:

If $e_i \notin OPT_{i-1}$, then $(f(OPT_{i-1}) - f(OPT_{i-1} \setminus \{e_i\})) = 0$. Furthermore, as $OPT_{i-1} = (OPT \cup X_{i-1}) \cap Y_{i-1} \subseteq Y_{i-1} \setminus \{e_i\}$, by submodularity,

$$f(OPT_{i-1}) - f(OPT_{i-1} \cup \{e_i\}) \leq f(Y_{i-1} \setminus \{e_i\}) - f(Y_{i-1}) = b_i.$$

If $e_i \notin OPT_{i-1}$, then $f(OPT_{i-1}) - f(OPT_{i-1} \cup \{e_i\}) = 0$ and $X_{i-1} \subseteq ((OPT \cup X_{i-1}) \cap Y_{i-1}) \setminus \{e_i\} = OPT_{i-1} \setminus \{e_i\}$. By submodularity

$$f(OPT_{i-1}) - f(OPT_{i-1} \setminus \{e_i\}) \leq f(X_{i-1} \cup \{e_i\}) - f(X_{i-1}) = a_i,$$

proving (4.9).

Substituting (4.8) and (4.9) in to (4.7), we need to show

$$\frac{a_i b_i}{a_i + b_i} \leq \frac{1}{2} \frac{a_i^2 + b_i^2}{a_i + b_i},$$

which follows from the binomial formula. □

Similar to the deterministic case, we can now conclude the main result:

Theorem 4.54 (Buchbinder et al. 2012)**.** *Algorithm 13 returns a solution S with $\mathbb{E}[f(S)] \geq f(OPT)/2$.*

Proof. We sum up (4.7) for all $1 \leq i \leq n$ and collapse the telescopic sum to obtain

$$\mathbb{E}[f(OPT_0) - f(OPT_n)] \leq \frac{1}{2}\mathbb{E}[f(X_n) - f(X_{i-1}) + f(Y_n) - f(Y_{i-1})] \leq \frac{\mathbb{E}[f(X_n) + f(Y_n)]}{2}.$$

With $OPT_0 = OPT$ and $OPT_n = X_n = Y_n = S$, we obtain the desired result $\mathbb{E}[f(S)] \geq f(OPT)/2$. □

The algorithm is in a sense best possible:

Remark 4.55. *Feige et al. have shown that any algorithm for (USM) based on oracle calls, providing a $(1/2 + \epsilon)$-approximation for any $\epsilon > 0$ requires an exponential number of oracle calls [2].*

4.7 Submodular Function Minimization

SUBMODULAR FUNCTION MINIMIZATION PROBLEM	
Input:	A finite set U, a submodular function $f : s^U \to \mathbb{Z}$ (given by an oracle)
Task:	Find a subset $X \subseteq U$ with $f(X)$ minimum

Lemma 4.56. *Let U be a finite set and $f : 2^U \to \mathbb{Z}$. Then f is submodular if and only if*

$$f(S+e) - f(S) \geq f(T+e) - f(T) \text{ for all } S \subset T \subset T + e.$$

4.7.1 Schrijvers Algorithm

Definition 4.57. *For a finite set U and a submodular function $f : 2^U \to \mathbb{Z}$, the* **base polyhedron** *is defined by*

$$B(f) := \left\{ x \in \mathbb{R}^U : \sum_{u \in A} x(u) \leq f(A) \text{ for all } A \subseteq U, \sum_{u \in U} x(u) = f(U) \right\}.$$

By a slight modification of the polymatroid greedy algorithm (Algorithm 11) we can derive the following result.

Theorem 4.58. *The vertices of the base polyhedron is precisely the set of vectors $b^\prec$ for all total orders $\prec$ of U, where*

$$b^\prec(u) := f(\{v \in U \ : \ v \preceq u\}) - f(\{v \in U \ : \ v \prec u\})$$

for $u \in U$.

Schrijver's algorithm maintains a point $x \in B(f)$ that is represented by an explicit convex combination $x = \lambda_1 b^{\prec_1} + \cdots + \lambda_k b^{\prec_k}$ of vertices of $B(f)$. At termination x will provide a certificate of optimality, also showing the following theorem, which for now will motivate the algorithm.

Theorem 4.59. *Let $f : s^U \to \mathbb{Z}$ be a submodular function such that $f(\emptyset) = 0$. Then*

$$\min_{S \subseteq U} f(S) = \max\{x^-(U) \ : \ x \in B(f)\},$$

where $x^-(U) = \sum_{u \in U} x^-(u) = \sum_{u \in U} \min(0, x(u))$.

The algorithm starts with $k = 1$ and $x = b^{\prec_1}$ and an arbitrary order $\prec_1$. On the fly the algorithm will create new total orders applying the following **operation**. For a total order $\prec$ and $s, u \in U$ we denote by $\prec^{s,u}$ the total order that results from $\prec$ by moving u just before s. By Ξ^U we denote the incedence vector of $u \in U$.

Theorem 4.60. *(Schrijver [2000])* SCHRIJVER'S ALGORITHM *works correctly.*

Algorithm 14 Schrijver's Algorithm

Instance: A finite set $U = \{1, \dots, n\}$ and a submodular function $f : 2^U \to \mathbb{Z}$ with $f(\emptyset) = 0$.
Output: A subset $X \subseteq U$ with $f(X)$ minimum.

Set $k := 1$, let $\prec_1$ be any total order on U, and set $x := b^{\prec_1}$.

BUILD GRAPH:
Set $D := (U, A)$, where $A = \{(u, v) \mid u \prec_i v \text{ for some } i \in \{1, \dots, k\}\}$.
Let $P := \{v \in U \mid x(v) > 0\}$ and $N := \{v \in U \mid x(v) < 0\}$.
Let X be the set of vertices not reachable from P in the digraph D.
if $N \subseteq X$ **then**
  stop
else
  let $d(v)$ denote the distance from P to v in D.
end if
Choose the vertex $t \in N$ reachable from P with $(d(t), t)$ lexicographically maximum.
Choose the maximal vertex s with $(s, t) \in A$ and $d(s) = d(t) - 1$.
Let $i \in \{1, \dots, k\}$ such that $\alpha := |\{v \mid s \prec_i v \preceq_i t\}|$ is maximum
  ▷ the number of indices attaining this maximum will be denoted by β.

CHANGE SOLUTION:
Compute a number ϵ with $0 \leq \epsilon \leq -x(t)$ and write $x' := x + \epsilon(\chi^t - \chi^s)$ as an explicit convex combination of at most n vectors, chosen among $b^{\prec_1}, \dots, b^{\prec_k}$ and $b^{\prec_i^{s,u}}$ for all $u \in U$ with $s \prec_i u \preceq_i t$, with the additional property that $b^{\prec_i}$ does not occur if $x'(t) < 0$.
Set $x := x'$, rename the vectors in the convex combination of x as $b^{\prec_1}, \dots, b^{\prec_{k'}}$, set $k := k'$, **go to** BUILD GRAPH.

Theorem 4.61. *(Schrijver [2000]) Each iteration can be performed in $O(n^3+\gamma n^2)$ time, where $\gamma$is the time for an oracle call.*

Lemma 4.62. *(Vygen 2003)* SCHRIJVER'S ALGORITHM *terminates after $O(n^5)$ iterations.*

Theorem 4.63. *The* SUBMODULAR FUNCTION MINIMIZATION PROBLEM *can be solved in $O(n^8 + \gamma n^7)$ time, where γ is the time for an oracle call.*

Corollary 4.64. *Linear functions over the intersection of two polymatroids can be optimized in polynomial time.*

4.8 Symmetric Submodular Functions

Definition 4.65. *A submodular function $F : 2^U \to \mathbb{R}$ is called* **symmetric** *if*

$$f(A) = f(U \setminus A) \text{ for all } A \subseteq U.$$

Minimizing symmetric submodular functions is trivial since by symmetrie, submodularity, and again symmetrie

$$2f(\emptyset) = f(\emptyset) + f(U) \leq f(A) + f(U \setminus A) = 2f(A)$$

for all $A \subseteq U$. Thus, we are looking for a non-empty proper subset $A \subset U$ minimizing $f(A)$.

Lemma 4.66. *Given a symmetric submodular function $f : 2^U \to \mathbb{R}$ with $n := |U| \geq 2$, we can find two elements $x, y \in U$ with $x \neq y$ and $f(\{x\}) = \min\{f(X) : x \in X \subseteq U \setminus \{y\}$ in $\mathcal{O}(n^2\theta)$ time, where θ is the time bound for the oracle for f.*

Theorem 4.67. *(Queranne [1998]) Given a symmetric submodular function $f : 2^U \to \mathbb{R}$, we can find in $\mathcal{O}(n^3\theta)$ time a nonempty proper subset A of U such that $f(A)$ is minimum, where θ is the time bound for the oracle for f.*

5 Survivable Network Design

SURVIVABLE NETWORK DESIGN PROBLEM	
Input:	An undirected graph G with weights $c : E(G) \to \mathbb{R}_+$, and a connectivity requirement $r_{xy} \in \mathbb{Z}_+$ for each $x, y \in V(G)$.
Task:	Find a minimum weight subgraph H of G such that for each x, y there are at least $r_{x,y}$ edge disjoint paths from x to y in H.

Example 5.1. *The* STEINER TREE PROBLEM *is an important special case. In this problem there is a set $T \subseteq V(G)$ of terminals and*

$$r_{xy} = \begin{cases} 1 & \text{if } x, y \in T. \\ 0 & \text{otherwise.} \end{cases}$$

The GENERALIZED STEINER TREE PROBLEM *is a* SURVIVABLE NETWORK DESIGN PROBLEM*, where $r_{xy} \in \{0, 1\}$ for all $x, y \in V(G)$.*

Example 5.2. *Solutions to the* SURVIVABLE NETWORK DESIGN PROBLEM*, where $r_{xy} = k$ for all $x, y \in V(G)$ are k-edge-connected subgraphs.*

Given a SURVIVABLE NETWORK DESIGN PROBLEM, we define a function $f : 2^{V(G)} \to \mathbb{Z}_+$ by $f(\emptyset) := f(V(G)) := 0$ and

$$f(S) := \max_{x \in S, y \in V(G) \setminus S} r_{xy} \quad \text{for } \emptyset \neq S \subset V(G).$$

Now that the SURVIVABLE NETWORK DESIGN PROBLEM can be formulated as an integer linear program:

$$\begin{array}{rrll} & \min & \sum_{e \in E(G)} c(e) x_e & \\ s.t. & x_e & \in \{0, 1\} & (e \in E(G)) \\ & \sum_{e \in \delta(S)} x_e & \geq f(S) & (S \subseteq V(G)). \end{array} \tag{SNDILP}$$

Definition 5.3. *A function $f : 2^U \to \mathbb{Z}_+$ is called* **proper** *if it satisfies the following three conditions:*

1. *$f(S) = f(U \setminus S)$ for all $S \subseteq U$ (symmetry)*
2. *$f(A \cup B) \leq \max\{f(A), f(B)\}$ for all $A, B \subseteq U$ with $A \cap B = \emptyset$.*
3. *$f(\emptyset) = 0$.*

The function f arising from the SURVIVABLE NETWORK DESIGN PROBLEM, as above is obviously proper.

Definition 5.4. *A function $f : 2^U \to \mathbb{Z}_+$ is called* **weakly supermodular** *if at least one of the following conditions hold for $A, B \subseteq U$:*

1. $f(A) + f(B) \leq f(A \cup B) + f(A \cap B)$ *(supermodularity)*
2. $f(A) + f(B) \leq f(A \setminus B) + f(B \setminus A)$

Proposition 5.5. *A proper function $f : 2^U \to \mathbb{Z}_+$ is weakly supermodular.*

We now show how the SEPARATION PROBLEM for the LP-relaxation of (SNDILP) can be solved using Gomory-Hu trees. Unfortunately, we cannot hope to find an optimal integral solution in polynomial time, because they would be solutions for *NP*-hard problems, e.g. the minimum Steiner tree problem, and ¶ = *NP* has not been proven yet.

Lemma 5.6. *Let G be a undirected graph with edge capacities $u : E(G) \to \mathbb{R}_+$, $f : 2^{V(G)} \to \mathbb{Z}_+$ be a proper function, and H be a Gomory-Hu tree for (G, u).*
Then for every $\emptyset \neq S \subset V(G)$ we have

1. $\sum_{e' \in \delta_G(S)} u(e') \geq \max_{e \in \delta_H(S)} \sum_{e' \in \delta_G(C_e)} u(e')$ *and*
2. $f(S) \leq \max_{e \in \delta_H(S)} f(C_e)$,

where C_e and $V(H) \setminus C_e$ are the connected components of $H - e$.

Theorem 5.7. *Let G be an undirected graph $x \in \mathbb{R}_+^{E(G)}$ and $f : 2^{V(G)} \to \mathbb{Z}_+$ be a proper function given by an oracle. The we can find in time $O(n^4 + n\theta)$ a set $S \subseteq V(G)$ with $\sum_{e \in \delta_G(S)} x_e < f(S)$ or decide, that no such set exists. Here $n = |V(G)|$ and θ is the time needed to query the oracle.*

This theorem allows us also to check whether (SNDILP) has feasible solutions.

5.1 A primal dual approxmation algorithm

In this section it we will develop a primal-dual algorithm for finding an edge set F, whose incidence vector fulfills (SNDILP). The algorithm runs in

$$k := \max_{S \subseteq V(G)} f(S) = \max_{x \in V(G)} f(\{x\})$$

phases, where the second equality holds, because f is proper. In iteration $1 \leq p \leq k$ we consider the proper function f_p defined by

$$f_p(S) := \max\{f(S) + p - k, 0\}.$$

The function values of f_1 are either zero or one and the empty set is almost feasible. In phase p, the incidence vector of the current edge set F will be modified to such that it satisfies (SNDILP) w.r.t. f_p. Thus, it will be almost satisfied w.r.t. f_{p+1} at the beginning of phase $p + 1$.

We will present an approximation algorithm that works well if k is not too large, e.g. for generalized Steiner tree problems.

Definition 5.8. *Let g be some proper function, $F \subseteq E(G)$ and $X \subseteq V(G)$. We say that X is* **violated** *with respect to (g, F) if $|\delta_F(X)| < g(X)$.*
The minimal violated sets with respect the (g, F) are the **active** *sets with respect to (g, F).*
$F \subseteq E(G)$ **satisfies** *g if no set is violated with respect to (g, F).*
$F \subseteq E(G)$ **almost satisfies** *g if $|\delta_F(X)| \geq g(X) - 1$ for all $X \subseteq V(G)$.*

The following lemma shows that active sets are pairwise disjoint and, thus, their number is linearly bounded.

Lemma 5.9. *Given a proper function g, a set $F \subseteq E(G)$ almost satisfying g, and two violated sets $A, B \subseteq V(G)$. Then either $A \setminus B$ and $B \setminus A$ are both violated or $A \cup B$ and $A \cap B$ are both violated. The active sets w.r.t. (g, F) are pairwise disjoint.*

The active sets can be computed using Gomory-Hu trees.

Theorem 5.10. *(Gabow, Goemans, Williamson [1998]) Given a proper (oracle) function g and a set $F \subseteq E(G)$ almost satisfying g. Then the active sets with respect to (g, F) can be computed in $O(n^4 + n^2\theta)$ time, where $n = |V(G)|$ and θ is the time needed to query the oracle for g.*

By F_p we denote the set F at the end of phase p and $F_0 := \emptyset$.

Algorithm 15 Primal-Dual Algorithm for Network Design

Input: An undirected graph G, weights $c : E(G) \to \mathbb{R}_+$, and an oracle for a proper function $f : 2^{V(G)} \to \mathbb{Z}_+$.
Output: A set $F \subseteq E(G)$ satisfying f.

if $E(G)$ does not satisfy f **then**
  stop ▷ the problem is infeasible.
end if
Set $F := \emptyset$, $k := \max_{v \in V(G)} f(\{v\})$, and $p := 1$.

START PHASE:
Set $i := 0$.
Set $\pi(v) := 0$ for all $v \in V(G)$.
Let $\mathcal{A}$ be the family of active sets with respect to (F, f_p), where f_p is defined by
  $f_p(S) := \max\{f(S) + p - k, 0\}$ for all $S \subseteq V(G)$.
while $\mathcal{A} \neq \emptyset$ **do**
  Set $i := i + 1$.
  Set $\epsilon := \min \left\{ \dfrac{c(e) - \pi(v) - \pi(w)}{|\{A \in \mathcal{A} : \ e \in \delta_G(A)\}|} \mid e = \{v, w\} \in \bigcup_{A \in \mathcal{A}} \delta_G(A) \setminus F \right\}$,
   and let e_i be some edge attaining this minimum.
  Increase $\pi(v)$ by ϵ for all $v \in \bigcup_{A \in \mathcal{A}} A$.
  Set $F := F \cup \{e_i\}$.
  Update $\mathcal{A}$.
end while
for $j := i$ down to 1 **do**
  if $F \setminus \{e_j\}$ satisfies f_p **then**
    set $F := F \setminus \{e_j\}$.
  end if
end for
if $p = k$ **then**
  stop,
else
  set $p := p + 1$ and
  go to START PHASE:.
end if

Lemma 5.11. *At each stage of phase p the set F almost satisfies f_p and $F \setminus F_{p-1}$ is a forest. At the end of phase p, F_p satisfies f_p.*

Lemma 5.12. *Determining ϵ and e_i in the* **While***-loop of Algorithm 15 can be done in $O(mn)$.*

Theorem 5.13. *(Goemans et al. [1994]) Algorithm 15 returns a set F satisfying f in $O(kn^5 + kn^3\theta)$ time, where $k = \max_{S \subseteq V(G)} f(S), n = |V(G)|$, and θ is the time for querying the oracle.*

To proove an approximation ratio, we implicitly construct a dual feasible solution of the LP relaxation and bound the difference of the cost of the solution constructed by Algorithm 15 and the dual solution. Let us take a look at the LP relaxation

$$\begin{aligned} & \min \sum_{e \in E(G)} c(e) x_e \\ s.t. \sum_{e \in \delta(S)} x_e & \geq f(S) && (S \subseteq V(G)). \\ x_e & \leq 1 && (e \in E(G)) \\ x_e & \geq 0 && (e \in E(G)). \end{aligned} \tag{LP}$$

and its dual

$$\begin{aligned} & \max \sum_{S \subseteq V(G)} f(S) y_S - \sum_{e \in E(G)} z_e \\ s.t. \sum_{S: e \in \delta(S)} y_S & \leq c(e) + z_e && (e \in E(G)). \\ y_S & \geq 0 && (S \subset V(G)) \\ z_e & \geq 0 && (e \in E(G)). \end{aligned} \tag{DP}$$

We implicitly construct dual solutions $(y^{(p)}, z^{(p)})$ during the course of Algorithm 15. At the beginning of phase p, we initialize $y^{(p)} = 0$. Then at each iteration $y_A^{(p)}$ is increased by ϵ for each active set $A \in \mathcal{A}$. Furthermore, we set

$$z_e^{(p)} := \begin{cases} \sum_{S: e \in \delta(S)} y_S^{(p)} & \text{if } e \in F_{p-1}. \\ 0 & otherwise. \end{cases}$$

In the algorithm, the dual solution is represented by the π variables. Note that $\pi(v) = \sum_{S: v \in S} y_S$ for all $v \in V(G)$. By our definition of $(y^{(p)}, z^{(p)})$ we get.

Lemma 5.14. *For each phase $p \in \{1, \dots, k\}$, $(y^{(p)}, z^{(p)})$ is a feasible solution of (**??**).*

Lemma 5.15. *For each phase $p \in \{1, \dots, k\}$,*

$$\sum_{S \subseteq V(G)} y_S^{(p)} \leq \frac{1}{k - p + 1} OPT(G, c, f),$$

where $OPT(G, c, f)$ is the value of an optimum integral solution.

Lemma 5.16. *At each iteration of any phase $p \in \{1, \dots, k\}$,*

$$\sum_{A \in \mathcal{A}} |\delta F_p \setminus F_{p-1}(A)| \leq 2|\mathcal{A}|.$$

where $OPT(G, c, f)$ is the value of an optimum integral solution.

To prove this lemma we use tree representations:

Definition 5.17. *Let T be a digraph such that the underlying undirected graph is a tree. Let U be a finite set and $\varphi : U \to V(T)$. Let $\mathcal{F} := \{S_e \mid e \in E(T)\}$, where for $e = (x, y)$ we define*

$$S_e := \{s \in U \mid \varphi(s) \text{ is in the same connected component of } T - e \text{ as } y\}.$$

Then (T, φ) is called a **tree-representation** *of $(U, \mathcal{F})$.*

Proposition 5.18. *Let $(U, \mathcal{F})$ be a set system with a tree-representation (T, φ). Then $(U, \mathcal{F})$ is cross-free. If T is an arborescence, then $(U, \mathcal{F})$ is laminar. Moreover, every cross-free family has a tree-representation, and for laminar families, an arborescence can be chosen as T.*

Now, we can bound the cost increase in a phase.

Lemma 5.19. *(Williamson [1995]) For each $p \in \{1, \dots, k\}$,*

$$\sum_{e \in F_p \setminus F_{p-s}} c(e) \leq 2 \sum_{S \subseteq V(G)} y_S^{(p)}.$$

Finally, we obtain the following approximation result.

Theorem 5.20. *Algorithm 15 returns a set F which satisfies f and whose weight is at most $2H(k)OPT(G, c, f)$ in $O\left(kn^5 + kn^3\theta\right)$ time, where $n = |V(G)|$, $k = \max_{S \subseteq V(G)} f(S)$, $H(k) = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$, and θ is the time required by the oracle for f.*

5.2 Iterative LP-Rounding

- Kamal Jain: A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem. Combinatorica, Volume 21, Issue 1, pp 39-60. 2001.
- Section 23 of book by Vijay Vazirani: Approximation Algorithms, Springer, 2003.

5.3 Degree Bounded Network Design Problems

- Lap Chi Lau, Joseph (Seffi) Naor, Mohammad R. Salavatipour, and Mohit Singh: Survivable Network Design with Degree or Order Constraints. SIAM J. Comput., 39(3), 1062–1087, 2009.

Bibliography

[1] Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz, *A tight linear time (1/2)-approximation for unconstrained submodular maximization*, Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (Washington, DC, USA), FOCS '12, IEEE Computer Society, 2012, pp. 649–658.

[2] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák, *Maximizing non-monotone submodular functions*, SIAM J. Comput. **40** (2011), no. 4, 1133–1153.