Programmierpraktikum Diskrete Optimierung
Das Time-Cost Tradeoff Problem

Stephan Held
held@or.uni-bonn.de

SoSe 2013

1 Das Diskrete Time-Cost-Tradeoff-Problem

Das Diskrete Time-Cost-Tradeoff-Problem (DTCTP) ist wie folgt definiert. Gegeben sind ein gerichteter azyklischer Graph (DAG) G und zwei Knoten $s, t \in V(G)$ mit $\delta^-(s) = \delta^+(t) = 0$. Der Graph modelliert ein Scheduling-Projekt. Die Kanten stellen Teilaktivitäten des Projekts dar, und der Graph definiert eine Ausführungsreihenfolge. D.h. eine Aktivität $(v, w) \in E(G)$ muss vor $(v', w') \in E(G)$ ausgeführt werden, wenn es einen w-v'-Pfad in G gibt.

Für jede Kante $e \in E(G)$ gibt es $l_e \in \mathbb{N}$ Paare $(\tau_e^1, c_e^1), \ldots, (\tau_e^{l_e}, c_e^{l_e}) \in \mathbb{R}^2$ $(1 \leq i \leq l_e)$, bestehend aus einer Ausführungszeit τ_e^i und Kosten c_e^i.

Häufig wird das DTCTP zunächst über eine partiell geordnete Menge von Aktivitäten definiert. Dieses Problem lässt sich aber auf einen oben beschriebenen Graphen reduzieren. Der Graph wird häufig auch als **Activity-On-Edge-Diagram** bezeichnet.

1.1 Die Gesamtlaufzeit einer Lösung

Für einen Lösungsvektor $(i_e)_{e \in E(G)}$ $(i_e \in \{1, \ldots, l_e\})$ kann die kürzest mögliche Gesamtlaufzeit des Projektes wie folgt bestimmt werden. Für jeden Knoten $v \in V(G)$ bestimmen wir eine frühest mögliche Zeit $\pi(v)$, zu der die Schritte $(v, w) \in E(G)$ in v gestartet werden dürfen, wobei $\pi(s) = 0$. Um die Ausführungsreihenfolge einzuhalten, muss offensichtlich folgende Ungleichung erfüllt sein:

$$\pi(v) + \tau_e^{i_e} \leq \pi(w) \qquad (v, w) \in E(G).$$

Die kürzest mögliche Gesamtlaufzeit ist daher durch den minimalen Wert $\pi(t)$ eines zulässigen Längste-Wege-Potentials gegeben. Sie kann in linearer Zeit bestimmt werden, da der Graph azyklisch ist.

2 Das Optimierungsziel

In der Regel werden zwei Optimierungs-Probleme betrachtet. Beim Deadline-Problem soll für eine maximale Gesamtlaufzeit T eine kostenminimale Lösung $(i_e)_{e \in E(G)}$ $(i_e \in$

$\{1, \ldots, l_e\}$) bestimmt werden, deren Gesamtlaufzeit kleiner oder gleich T ist. Beim BUDGET-PROBLEM soll für maximale Gesamtkosten B eine Lösung $(i_e)_{e \in E(G)}$ ($i_e \in \{1, \ldots, l_e\}$) mit minimaler Gesamtlaufzeit bestimmt werden, deren Gesamtkosten kleiner oder gleich B sind.

Im Rahmen dieses Programmierpraktikums betrachten wir das folgende Problem. **Zunächst wird die kleinste mögliche Gesamtlaufzeit $T^{\min}$ (wenn alle Kanten schnellstmöglich ausgeführt werden) und anschließend eine (approximative) Lösung für das DEADLINE PROBLEM mit $T^{\min}$ bestimmt.**

Das DCTP ist NP-schwer und vermutlich schwer zu approximieren. Genauer gibt es keinen Algorithmus mit konstantem Approximationsfaktor, falls die sogenannte Unique-Games-Conjecture gilt (O. Svensson: Hardness of Vertex Deletion and Project Scheduling. arxiv (2012)). Das Ziel dieses Praktikums ist es, im Chip Design auftretende Instanzen "möglichst" gut zu lösen und ggf. a posteriori eine Güte der Lösung anzugeben.

3 Streng monoton fallende Kosten

O.B.d.A. können wir davon ausgehen, dass eine Alternative einer Kante e, die schneller ist als eine andere Alternative für e, auch teurer ist. Ansonsten kann die langsamere Alternative gestrichen werden.

4 Lineare Relaxierung

Um die Güte einer Lösung abzuschätzen, müssen ihre Kosten mit einer unteren Schranke für die Kosten verglichen werden. Dabei ist es wünschenswert, möglichst hohe untere Schranken zu haben.

Eine untere Schranke ergibt sich aus der linearen Relaxierung, die durch eine Minimum-Cost-Flow Berechnung gelöst werden kann. Dazu reduzieren wir das Problem zunächst auf zwei Alternativen pro Kante.

4.1 Reduktion auf 2 Alternativen

Es gibt verschiedene Möglichkeiten, das DTCTP auf 2 Alternativen pro Kante zu reduzieren, entweder durch Einführung paralleler Kanten oder Ersetzung einer Kante durch einen Pfad. Wir wählen erstere.

Sei e eine Kante mit einer endlichen Anzahl $l_e > 2$ zulässiger Ausführungszeiten $\tau_e^1 < \tau_e^2 < ... < \tau_e^{l_e}$ gegeben. O.B.d.A. gelte $c(\tau_e^{l_e}) = 0$.

Wir ersetzen e durch l_e parallele Kanten $e_1, \ldots, e_{l_e}$ Die Kanten e_1 hat nur eine Alternative $(\tau_e^1, 0)$, die sicherstellt, dass die minimale Ausführungszeit niemals unterschritten wird. Für $i \in \{2, \ldots, l_e\}$ fügen wir jeweils eine Kante e_i mit den zwei Alternativen $(0, c_e^{i-1} - c_e^i)$ und $(\tau_e^i, 0)$ ein. Hierdurch kann die Aktivität der Originalkante e nur dann schneller als τ_e^i ausgeführt werden, wenn die Kostendifferenz zur nächstschnelleren Alternative gezahlt wird.

4.2 Minimum-Cost Flow Relaxierung

Wir gehen davon aus, dass jede Kante e 2 Alternativen $(\tau_e^u, 0), (\tau_e^l, \tilde{c}_e)$ mit $\tau_e^u \geq \tau_e^l$ hat. Mit

$$c_e := \begin{cases} \frac{\tilde{c}_e}{\tau_e^u - \tau_e^l}, & \text{falls } \tau_e^u > \tau_e^l \\ 0, & \text{falls } \tau_e^u = \tau_e^l (e \text{ hat nur eine Realisierung}) \end{cases}$$

läßt sich das DTCTP wie folgt beschreiben:

$$\begin{aligned} & & \min \quad & \sum_{e \in E(G)} c_e(\tau_e^u - d_e) & & \\ s.d. \quad & \pi(v) - \pi(w) + d_{(v,w)} & \leq 0 & & (v,w) \in E(G) \\ & \pi(t) - \pi(s) & \leq T^{\min} & & \\ & d_e & \in \{\tau_e^u, \tau_e^u\} & & (e \in E(G), \end{aligned} \tag{1}$$

wobei die Variablen d_e $(e \in E(G))$ das gewählte Delay der Kante e angeben.

Relaxieren wir die Bedingunen $d_e \in \{\tau_e^u, \tau_e^u\}$ zu $\tau_e^l \leq d_e \leq \tau_e^u$, so erhalten wir ein sogenanntes lineares Programm (LP). Dieses ist äquivalent zu folgendem Minimum-Cost-Flow-Problem mit stückweise linearen Kosten:

$$\begin{aligned} & & \min \quad & \sum_{e \in E(G)} T_e(x_e) + T^{\min} f & & \\ s.d. \quad & \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e & = \begin{cases} f, & \text{falls } v = t \\ -f, & \text{falls } v = s \\ 0, & \text{sonst.} \end{cases} & & (v \in V(G)) \\ & x_e & \geq 0 & & (e \in E(G)), \end{aligned} \tag{2}$$

wobei die Kostenfunktion $T_e : \mathbb{R}_+ \to \mathbb{R}$ durch

$$T_e(x_e) = \begin{cases} -\tau_e^u \cdot x_e, & \text{falls } x_e \leq c_e \\ -\tau_e^l \cdot x_e - c_e(\tau_e^u - \tau_e^l), & \text{sonst} \end{cases}$$

definiert ist. Dieses kann wiederum auf ein Minimum-Cost-Flow mit linearen Kosten reduziert werden, indem jede Kante durch zwei parallele Kanten mit geeigneten Kapazitäten und Kosten ersetzt wird.

Details zu dieser Reduktion finden sich bei *E. Lawler: Combinatorial Optimization, Networks and Matroids. Holt, Rinehart, and Winston (1976), 165–168.*

5 Rundung

Eine dual optimale Lösung π der Minimum-Cost-Flow-Instanz können wir wie folgt runden. Traversiere die Knoten in topologischer Ordnung startend in s. Für jeden besuchten Knoten v setze die Delays der eingehenden Kanten auf den größten/billigsten Wert, so dass die Länge eines längsten s-v-Pfades kleiner gleich $\pi(v)$ ist.

Durch dieses Schema wird zum einen sichergestellt, dass die Projektdauer $T^{\min}$ eingehalten wird. Zum anderen wird ausgenutzt, dass in der diskreten Lösung Kanten billiger gewählt werden können als in der linearen Relaxierung, falls vorhergehende Kanten schneller gewählt wurden.

6 Eingabe-Daten

Das Format für die Time-Cost-Tradeoff-Instanzen ist eine Erweiterung des DIMACS-Formats http://dimacs.rutgers.edu/Challenges/.

Es gibt folgende Zeilen

Kommentare Kommentarzeilen beginnen mit einem 'c' als erstes Token und können an beliebigen Stellen der Eingabedatei auftreten.

```
c This is a comment
```

Problemzeile Die Problemzeile kommt genau einmal vor und muss als erste Nicht-Kommentarzeile auftreten. Die Zeile hat folgendes Format, wobei n und m die Anzahl der Knoten und Kanten im Graphen sind. Die Knoten des Graphen werden mit 1 bis n indiziert und die Kanten mit 1 bis m.

```
p arc n m
```

Kanten-Zeilen Kanten-Zeilen haben die folgende Form, wobei U und V die Knoten-Ids von Anfangs- und Endpunkt sind. Die Kanten werden implizit in der Reihenfolge ihrer Definition nummeriert.

```
e U V
```

Delay-Realisierungs-Zeilen Kantendelays werden durch Kanten der folgenden Form spezifiziert.

```
i 1 E D C
```

Hierbei sind E die Kanten-Id, D das Delay und C die Kosten dieser Realisierung für E. Das zweite Token wird für ein allgemeineres Problem benötigt und ist für unsere Zwecke immer gleich 1. Jede Delay-Alternative einer Kante wird durch eine neue Zeile angegeben. Die Delay Alternativen einer Kanten dürfen erst spezifiziert werden, wenn E durch eine Kanten-Zeile erzeugt wurde.

s-t-**Zeile** Die dedizierten Knoten s und t werden durch eine Zeile der folgenden Form angegeben, wobei S die Startknoten-Id und T die Endknoten-Id sind.

```
t T S
```

Die Reihenfolge ist so gewählt, da (t, s) als Kante aufgefasst werden kann, die

$$\pi(t) - T \leq \pi(s)$$

erfüllen muss.

Testinstanzen finden sich auf der Webseite zum Programmierpraktikum http://www.or.uni-bonn.de/lectures/ss13/praktikum_ss13.html

Programmiersprache

Die Implementierungen müssen in C/C++ ausgeführt werden und unter Linux mit dem gcc/g++ kompilierbar sein, d.h. sie müssen dem ANSI C99/C++98-Standard genügen. Außer den Standard-Libraries, insbesondere der STL, dürfen keine Hilfsbibliotheken verwendet werden.

Der Code muss verständlich programmiert und kommentiert werden. Selbstverständlich müssen die Programme fehlerfrei funktionieren. Insbesondere werden sie mit dem Programm valgrind (http://valgrind.org) überprüft und valgrind-Fehler führen zu Abzügen.

Einführungsaufgabe

Als Einführungsaufgabe sollte ein Programm implementiert werden, das eine Time-Cost-Tradeoff-Instanz einliest und die minimale mögliche Projektdauer $T^{\min}$ bestimmt.

Spätester Abgabetermin für die Einführungsaufgabe ist der 30.04.2013

7 Aufgaben

7.1 Approximations-Algorithmus

Aufgabe 1: Modifizierter Bar-Yehuda-Even-Algorithmus + Lokale Suche

Startend mit einer der billigsten Alternative für jede Kante wird sukzessive die billigste Kante e auf einem derzeit längsten s-t-Pfad P beschleunigt und die Kosten der anderen Kanten auf P um die Kosten von e reduziert. Details sind in der Bachelorarbeit von *Sonja Wittke: Diskrete Time-Cost-Tradeoff-Probleme. Forschungsinstitut für Diskrete Mathematik (2012)*, Abschnitt 3.2, zu finden.

Da diese Lösung voraussichtlich relativ teuer ist, soll anschließend versucht werden, sie durch lokale Suche zu verbessern, z.B. indem das Problem auf zusammenhängenden Teilproblemen mit sehr wenigen Kanten optimal gelöst wird. Ideen hierfür können aus *De et al.: Complexity of the Discrete Time-Cost Tradeoff Problem for Project Networks. Operations Research, Vol. 45, No. 2 (1997), 302–306*, Abschnitt 1, gewonnen werden.

7.2 Minimum-Cost Flow

Aufgabe 2: Cost-Scaling (Preflow-Push)

Implementieren Sie die Lösung der Relaxierung und deren Rundung unter Verwendung des Preflow-Push-Cost-Scaling-Algorithmus von Goldberg und Tarjan zum Lösen des Minimum-Cost-Flow-Problems.

Aufgabe 3: Netzwerk-Simplex
Implementieren Sie die Lösung der Relaxierung und deren Rundung unter Verwendung des Netzwerk-Simplex-Algorithmus zum lösen des Minimum-Cost-Flow-Problems.
(siehe Korte und Vygen, 2012, Kapitel 9.6)

Aufgabe 4: Capacity Scaling (Successive Shortest Path) Implementieren Sie die Lösung der Relaxierung und deren Rundung unter Verwendung des Capacity Scaling Algorithmus zum Lösen des Minimum-Cost-Flow-Problems.
(siehe Korte und Vygen, 2012, Kapitel 9.4)

Aufgabe 5: Cycle Canceling
Implementieren Sie die Lösung der Relaxierung und deren Rundung unter Verwendung des Cycle-Cancelling-Algorithmus von Golderg und Tarjan zum Lösen des Minimum-Cost-Flow-Problems.

7.3 Minimum-s-t-Cut

Aufgabe 6:
Lösen Sie die lineare Relaxierung mit dem Algorithmus von Phillips und Dessouky, d.h. durch eine Folge von minimalen Schnitten und runden Sie diese am Ende. Hierzu muss ein Maximum-Flow-Algorithmus, z.B. Push-Relabel implementiert werden. *Phillips, S.J. and Dessouky, M. I.: Solving the project time/cost tradeoff problem using the minimal cut concept. In: Management Science 24 (1977), Nr. 4, S. 393–400.*

Aufgabe 7:
Wie Aufgabe 6, nur dass der minimale Schnitt in jeder Iteration diskret realisiert wird. Das heißt startend mit einer langsamsten Lösung wird iterativ ein Minimum-Cut bestimmt, anhand dessen zu einer schnelleren Lösung gerundet wird.

7.4 Backup

Aufgabe 8: Ähnlich zu Aufgabe7b, nur dass mit einer teuresten Lösung begonnen wird und diese entlang von maximal gewichteten Antiketten verbilligt wird.
(siehe Korte und Vygen, 2012, Kapitel 8, Aufgabe 28)

Spätester Abgabetermin für das Programmierpraktikum ist der 12.07.2013.