TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

TU Kaiserslautern
AG Algorithmen und Komplexität

Ausgabedatum: 17.05.2013
Version: 2013-05-19 16:12

5. Übungsblatt zur Vorlesung Kombinatorische Algorithmen, SS 13

Abgabe: *Bis Montag,* 03.06.2013, *12:00 Uhr,*
in den Abgabekasten vor dem AG-Flur oder per Email an `wild@cs.uni....`

6. Aufgabe

2 + 2 Punkte

Sei $G = (V, E)$ ein einfacher Graph mit Kapazität $c(e) \in N \subset \mathbb{N}$ auf Kante $e \in E$. (Wir nehmen an, dass es keine unteren Schranken $l(e)$ für die Kanten gibt, bzw. $l(e) = 0$.)

Zeigen oder **widerlegen** Sie:

a) Wenn alle Kapazitäten *gerade* Zahlen sind, d. h. $N = 2\mathbb{N} = \{2n \mid n \in \mathbb{N}_0\}$, so gibt es stets einen *maximalen* (s,t)-Fluss f^*, der nur gerade Flusswerte besitzt, d. h. $\forall e \in E : f^*(e) \in N$.

b) Wenn alle Kapazitäten *ungerade* Zahlen sind, d. h. $N = 2\mathbb{N} + 1 = \{2n + 1 \mid n \in \mathbb{N}_0\}$, so gibt es stets einen *maximalen* (s,t)-Fluss f^*, der nur ungerade Flusswerte besitzt, d. h. $\forall e \in E : f^*(e) \in N$.

7. Aufgabe

3 Punkte

In Anwendungen haben wir oft das Ziel, nicht einen *maximalen* Fluss zu bestimmen, der die Kapazitäten komplett auslastet, sondern vielmehr ist das Entscheidungsproblem interessant, ob das Netzwerk einen Fluss zulässt, der eine vorgegebene Menge von Fluss aus bestimmten Quellen zu bestimmten Senken transportieren kann. Als Beispiel mag ein Abwassernetz dienen, in das an bestimmten Knoten eine gewisse Menge Wasser pro Zeit einführt wird, die aus anderen Knoten wieder entnommen wird.

Formal modellieren wir diese Situation als *Feasible Flow Problem*: Gegeben ein einfacher Graph $G = (V, E)$ mit (oberen) Kapazitäten $c : E \to \mathbb{R}_{\geq 0}$ auf den Kanten, sowie *Überschuss* $b : V \to \mathbb{R}$ in den Knoten, gibt es einen *zulässigen* Fluss $f : E \to \mathbb{R}$ mit

$$\forall v \in V \qquad b(v) + \sum_{\substack{e \in E \\ e=(u,v)}} f(e) = \sum_{\substack{e \in E \\ e=(v,u)}} f(e) \quad \text{und} \tag{1}$$

$$\forall e \in E \qquad 0 \leq f(e) \leq c(e) \quad ? \tag{2}$$

Dabei bezeichnet man Knoten $v \in V$ mit positivem Überschuss $b(v) > 0$ als *Quellen*, und solche mit negativem Überschuss $b(v) < 0$ – also Knoten mit *Bedarf* – als *Senken*.

Zeigen Sie, dass man das *Feasible Flow Problem* auf das *Max-Flow Problem* reduzieren kann, d. h. geben Sie einen Algorithmus an, der das *Feasible Flow Problem* löst, indem er einen Max-Flow Algorithmus als Unterprogramm verwendet.

8. Aufgabe

4 Punkte

Wir betrachten das folgende *Scheduling Problem*: Gegeben m identische Maschinen $M_1, \ldots, M_m$, die wir parallel betreiben können, sowie n Aufträge (*Jobs*), die rechtzeitig abgearbeitet werden sollen.

Jeder Auftrag j hat eine Bearbeitungszeit $p_j \in \mathbb{N}$, d. h. eine Anzahl Zeitslots, die j eine Maschine belegt. Außerdem kann Job j erst nach seiner Eingangszeit (*release date*) $r_j \in \mathbb{N}$ begonnen werden und muss vor Ablauf seiner Deadline $d_j \in \mathbb{N}$, mit $d_j \geq r_j + p_j$, komplett bearbeitet worden sein. Dabei nehmen wir an, dass die Zeit in diskreten Zeitschritten gegeben ist (vgl. *Unix-Epoche*).

Jede Maschine kann zu einem Zeitpunkt höchstens einen Job bearbeiten und einzelne Jobs können nicht parallel auf mehreren Maschinen gleichzeitig bearbeitet werden. Wir erlauben aber, Jobs auf einer Maschine zu starten und vor vollständiger Bearbeitung zu pausieren. Der Job kann später – evtl. auf einer anderen Maschine – ohne zusätzliche Kosten fortgesetzt werden (*preemption*).

Die Aufgabe besteht darin zu entscheiden, ob es möglich ist, mit den gegebenen Maschinen alle Jobs vor Ablauf ihrer Deadline abzuschließen – und falls ja, einen solchen Fahrplan zu berechnen.

Geben Sie einen Algorithmus an, der dieses Problem mit Laufzeit polynomiell in der Eingabekodierung löst. Teilpunkte gibt es – wie immer – auch auf weniger effiziente Lösungen.

Tipp: Modellieren Sie das Problem als *Max-Flow Problem*.