Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Übungsblatt 2

Vorlesung Algorithmen II im WS 12/13

Ausgabe 30. Oktober 2012
Besprechung 13. November 2012

Problem 1: Algorithmus von Goldberg und Tarjan – Maximale Flüsse

Die Grundidee des Algorithmus von Goldberg und Tarjan basiert auf Präflüssen und PUSH- und RELABEL-Operationen. Algorithmen, die dieses Grundkonzept nutzen, nennt man daher auch *Preflow-Push-* oder *Push-Relabel-*Algorithmen.

Gegeben sei nun ein Flussnetzwerk $(D; s, t; c)$ bzw. dessen Erweiterung $(D'; s, t; c')$ und ein darin mit einem *Push-Relabel-*Algorithmus berechneter maximaler Fluss (d.h. die Abbildungen f, r_f, e und dist stehen bezüglich des fertig berechneten Flusses zur Verfügung).

Entwickeln Sie einen möglichst schnellen Algorithmus um die Knotenpartition eines zugehörigen minimalen s-t-Schnitts $(S, V \backslash S)$ zu berechnen. Geben Sie Ihren Algorithmus in Pseudocode an. Begünden Sie die Laufzeit und die Korrektheit Ihres Algorithmus

Hinweis: Es gibt einen Algorithmus Laufzeit $O(|V|)$.

Problem 2: Flüsse mit komplizierten Kosten

In der Vorlesung wurde bisher angenommen, dass die Kostenfunktion $\text{cost}\colon E \to \mathbb{R}_0^+$ eines Flussnetzwerks $(D = (V, E); \text{cost}; b; c)$ jeder Kante konstante Kosten pro Flusseinheit zuweist, d.h. ein Fluss f verursacht auf einer Kante e die Kosten $f(e) \cdot \text{cost}(e)$. Wir betrachten nun den Fall, dass jeder Kante $e \in E$ eine beliebige Kostenfunktion $\text{cost}_e\colon \mathbb{N}_0 \to \mathbb{R}$ zugewiesen wird. Ein ganzzahliger Fluss f verursacht auf der Kante e dann gerade $\text{cost}_e(f(e))$ Kosten.

Mit $\text{cost}_e(x) = k \cdot x$ für eine Konstante k erhält man für e eine Kostenfunktion, die wie zuvor konstante Kosten pro Flusseinheit veranschlagt. Man kann allerdings auch kompliziertere Kostenfunktionen zuweisen, wie das folgende Beispiel einer Stufenfunktion zeigt.

$$\text{cost}_e(x) = \begin{cases} 0 & x < 5 \\ 1 & \text{sonst} \end{cases}$$

Zeigen Sie:

(a) Wenn die Funktion cost_e für alle $e \in E$ konvex ist, dann kann MINCOSTFLOW effizient (d.h. in polynomieller Zeit) gelöst werden (unter der Voraussetzung, dass die Kapazitäten polynomiell beschränkt sind).

Hinweis: Modellieren sie ein äquivalentes Flussnetzwerk mit herkömmlichen Kosten. Sie dürfen Mehrfachkanten benutzen.

(b) Für potentiell nicht konvexe Kostenfunktionen ist das Entscheidungsproblem von MinCost-Flow $\mathcal{NP}$-vollständig.

Hinweis: Sie dürfen verwenden, dass 3-Dimensional-Matching $\mathcal{NP}$-vollständig ist.

3-Dimensional-Matching
Geg.: Endliche, disjunkte Mengen X, Y und Z mit einer Menge von Tripeln $T \subseteq X \times Y \times Z$, sowie Zahl $k \in \mathbb{N}$.
Ges.: Teilmenge $M \subseteq T$ mit $|M| \geq k$, sodass jedes Element aus X, Y bzw. Z in maximal einem Tripel in M auftaucht.

Problem 3: Flüsse mit Mindestfluss

Sein $D = (V, E)$ ein Flussnetzwerk wobei zusätzlich auf jeder Kante ein *Mindestfluss* gefordert ist. Das heißt, gegeben ist eine Abbildung $\ell \colon E \to \mathbb{R}_0^+$ und ein Fluss f muss zusätzlich zur Kapazitäts- und Flusserhaltungsbedingung die Ungleichung $\ell(e) \leq f(e)$ (für alle $e \in E$) erfüllen.

Zeigen Sie, dass ein Flussproblem mit gefordertem Mindestfluss in ein Flussproblem ohne diese zusätzliche Bedingung transformiert werden kann.

Hinweis: Sie dürfen zusätzliche Quellen und Senken erstellen.

Problem 4: Matchings – Erhöhende Wege

In der Vorlesung wurde ein Algorithmus für die Berechnung eines bipartiten Matchings vorgestellt, der auf der Idee basiert einen maximalen Fluss in einem geeignet gewählten Netzwerk zu berechnen. Hierzu wird das Prinzip des Ford-Fulkerson-Algorithmus angewendet, dass der maximale Fluss schrittweise mithilfe von erhöhenden Wegen berechnet werden kann. In jedem dieser Schritte induziert der aktuelle Fluss f ein (nicht notwendigerweise maximales) Matching. Erklären Sie inwiefern dieses Matching sich im Verlauf der Berechnung von f ändert. Gehen Sie hierzu insbesondere auf die Rolle der erhöhenden Wege ein.

Problem 5: Kreuzende Schnitte – Schnitte in Graphen

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A := S \cap T$, $B := S \setminus T$, $C := T \setminus S$ und $D := V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

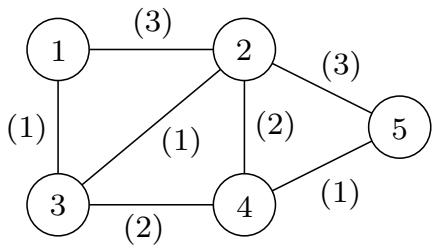
(a) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: Es gilt $s \in B$ und $t \in C$.

(b) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende minimale s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind zwei kreuzungsfreie minimale s-t-Schnitte.

Problem 6: Algorithmus von Stoer & Wagner – Schnitte in Graphen

(a) Wenden Sie auf den unten abgebildeten Graphen (Kantengewichte in Klammern) den Algorithmus von Stoer & Wagner an. Geben Sie nach jeder Phase die Knoten s und t, den Schnitt der Phase und dessen Gewicht an und zeichnen Sie den nach dem Verschmelzen resultierenden Graphen (mit Kantengewichten). **Verwenden Sie in Phase i den Knoten als Startkno-**

ten, der Knoten i des Originalgraphen enthält. Geben Sie zum Schluss den minimalen Schnitt S_{min} an.



(b) Liefert der Algorithmus von Stoer & Wagner auch für negative Kantengewichte einen global minimalen, nichttrivialen Schnitt? Begründen Sie Ihre Antwort.