Algorithmendesign
Lösungen zu Übungsblatt 6
WS 13/14

Gruppe 2
Max Bannach

Markus Richter (614027)

Aufgabe 6.1 Binäre Zuweisung

Teilaufgabe 1)

Gegeben sei ein ungerichteter Graph $G = (V, E)$. Weiterhin sei G ein bipartiter Graph, d. h. die Knotenmenge V kann in $X \cup Y$ partitioniert werden. X und Y sind disjunkt und alle Kanten $e \in E$ verlaufen zwischen X und Y. Darüber hinaus hat jeder Knoten v mindestens eine inzidente Kante. Ein Matching in einem bipartiten Graphen G ist ein Subgraph $G' = (V', E'), V' \subseteq V, E' \subseteq E$, für dessen Menge von Kanten $e' \in E'$ gilt, dass e' $u \in V$ und $v \in V$ zu einem Paar (u, v) verbindet. Die Knoten dieses Paares werden in V' aufgenommen. Für jedes Paar aus V' muss aber gelten:

- $d(u, v) = 1$, wenn eine Kante $e' \in E'$ existiert, die u' und v' verbindet
- $d(u, v) = 0$, wenn nicht

Außerdem gilt für $u' \in V'$ und $v' \in V'$: $c(u') = c(v') = 1$. Maximum Matching beschreibt nun den Subgraph G' mit den meisten Kanten.

Teilaufgabe 2)

Um das Binary Assignment Problem zu lösen kann die Ford-Fulkerson-Methode verwendet werden. Die Idee ist nun die, dass man versucht ein Flussnetzwerk $G' = (V', E')$ zu konstruieren, sodass der Fluss dem Matching entspricht. Dazu muss aber vorerst eine Quelle q und eine Senke s hinzugefügt werden, sodass gilt $V' = V \cup \{q, s\}$. Die gerichteten Kanten E' von G' bestehen aus den Kanten von E, die von X nach Y gerichtet sind und den $|V|$ neuen gerichteten Kanten (von der Quelle weg und zur Senke hin):
$E' = \{(s, u) | u \in X\} \cup \{(u, v) | (u, v) \in E\} \cup \{(v, s) | v \in Y\}$
Für die Kapazitäten gilt: $\gamma(q, u_i) = c(u_i)$, für $\gamma(v_i, s) = c(v_i)$ und für $\gamma(u_i, v_j) = d(u_i, v_j)$.
Mit diesem Flussnetzwerk G' können wir nun die Ford-Fulkerson-Methode anwenden. Aus dem daraus resultierenden maximalen Fluss f lässt sich nun die Anzahl der Tupel (u_i, v_i) direkt aus dem Fluss über der Kante (u_i, v_i) ablesen. Zum Schluss wird also jede Kante (u_i, v_i) durchlaufen.

Korrektheit: Zum einen ist die verwendete Ford-Fulkerson-Methode korrekt. Nach Anwendung der Ford-Fulkerson-Methode wird der resultierende Restgraph durchlaufen und die Tupel, sowie deren Anzahl, aus dem Fluss ermittelt. Da der Restgraph einem korrekten Ergebnis entspricht ist das Ablesen des Flusses und das Zuordnen dessen Werte ebenfalls korrekt. Damit ist der Algorithmus korrekt.

Laufzeit: Das nachträgliche Durchlaufen der Kanten, um die Tupel zu bestimmten benötigt $\mathcal{O}(E)$. Dazu kommt die Laufzeit der Ford-Fulkerson-Methode, speziell eine Implementierung mit Breitensuche, z. B. der Algorithmus von Edmonds und Karp, also $\mathcal{O}(|V| \cdot |E|^2)$. Letztere Laufzeit dominiert, daher ergibt sich für für den Algorithmus eine Laufzeit von $\mathcal{O}(|V| \cdot |E|^2)$.

Aufgabe 6.2 Kantenänderungen

Teilaufgabe 1)

Sei $r(e)$, $e \in E$ oder $r(u,v)$, $u,v \in V$ die Kapazitätsfunktion aus dem Skript. $G' = (V', E')$ sei der Restgraph, der aus dem Graphen G resultiert. Weiterhin sei e_i die betroffene Kante. Wenn e_i bekannt ist, dann ist auch u und v aus $e_i(u,v)$ bekannt. Wenn die Kapazität einer einzelnen Kante e_i um 1 inkrementiert wird, gibt es für den maximalen Fluss die folgenden Möglichkeiten:

1. Gibt es im Restgraphen G' einen aufsteigenden Weg $\pi = q, e_1, \ldots, e_k, s$ von der Quelle q zur Senke s, so dass $e_i \in \pi$, dann wird der ehemals maximale Fluss f um 1 erhöht. e_i war damit der Engpass.
2. Gibt es diesen Weg π nicht, wird der maximale Fluss f nicht erhöht.

Teilaufgabe 2)

Es wird mit einer Breitensuche rückwärts von u zur Quelle der kürzeste Weg gesucht, der die zusätzliche Kapazität von 1 noch aufnehmen kann. Danach wird ebenfalls mit der Breitensuche der kürzeste Pfad von v zur Senke gesucht, der die zusätzliche Kapazität von 1 aufnehmen kann. Wird in beiden Fällen ein Pfad gefunden wird der ehemals maximale Fluss um 1 erhöht, andernfalls nicht.

Korrektheit: Der Algorithmus sucht mit einer Breitensuche vor- und rückwärts nach einem Weg, der die zusätzliche Kapazität von 1 aufnehmen kann und findet definitiv einen Weg, wenn es einen gibt. Die Suche ist also korrekt. Die Folgerung daraus, dass der maximale Fluss sich um 1 erhöht, wenn es einen solchen Weg gibt, ist korrekt, da die betroffene Kante e_i der Engpass in diesem Weg war. Andernfalls ändert sich nichts. Der Algorithmus ist also korrekt.

Laufzeit: im schlimmsten Fall werden alle Kanten betrachtet, die Laufzeit beträgt also $\mathcal{O}(|E|)$.

Teilaufgabe 3)

Analog zu Teilaufgabe 1 gibt es ebenfalls die Möglichkeit eines Weges π, mit dem Unterschied, dass der Fluss um 1 gesenkt wird, falls es einen Weg π gibt, aber nur, wenn kein paralleler Weg π' von der Quelle zu v existiert, der in seiner Kapazität um 1 erhöht werden kann. In dem Fall war nämlich nicht e_i der Engpass, sondern eine nachfolgende Kante.

Teilaufgabe 4)

Analog zu Teilaufgabe 2 wird vorwärts und rückwärts ein Weg gesucht und wenn in beiden Fällen einer gefunden wurde wird rückwärts mit Breitensuche von v aus der kürzeste Pfad zur Senke gesucht, der um die Kapazität 1 erweitert werden kann. Wird kein solcher Pfad gefunden, wird der maximale Fluss f um 1 gesenkt.

Korrektheit Analog zu Teilaufgabe 2 ist auch in diesem Fall die Korrektheit für die Breitensuche gegeben. Die Erweiterung besteht darin, dass ein Alternativpfad von der Quelle zum Knoten v gesucht wird, der die Dekrementierung der Kante e_i um 1 ausgleichen kann. Nur wenn dieser Alternativpfad nicht existiert dekrementiert der Algorithmus den ehemals maximalen Fluss um 1. Der Algorithmus ist also korrekt.

Laufzeit. Die Laufzeit beträgt zum einen wieder $\mathcal{O}(|E|)$, nur das jetzt noch nach einem Alternativpfad gesucht werden muss, was einem nochmaligen Durchlaufen der Kanten entspricht d. h. $2 \cdot |E|$, also $\mathcal{O}(|E|)$.