Optimierung für Studierende der Informatik

Thomas Andreae

Wintersemester 2017/18
Blatt 11

B: Hausaufgaben zum 22./23. Januar 2018

1. Wir knüpfen an Präsenzaufgabe 1 an und betrachten das dort formulierte Problem WEIGHTED k-HITTING SET. Die Bezeichnungen (ILP) und (LP) verwenden wir wie in dieser Präsenzaufgabe.
   a) Geben Sie basierend auf (LP) einen (polynomiellen) Approximationsalgorithmus für WEIGHTED k-HITTING SET an, bei dem es sich um einen k-Approximationsalgorithmus handelt.
   b) Weisen Sie nach, dass es sich bei dem von Ihnen angegebenen Algorithmus tatsächlich um einen k-Approximationsalgorithmus handelt.

a) Wir betrachten die Integer Programming Formulierung (ILP) des Problems WEIGHTED k-HITTING SET:

minimiere $w_1x_1 + \ldots + w_nx_n$
unter den Nebenbedingungen

$$\begin{aligned} \sum_{s_i \in T_j} x_i &\geq 1 \qquad j = 1, \ldots, m \\ 0 \leq x_i &\leq 1 \qquad i = 1, \ldots, n \\ x_i &\in \mathbb{Z} \qquad i = 1, \ldots, n \end{aligned} \qquad \text{(ILP)}$$

Die LP-Relaxation dieses Problems lautet:

minimiere $w_1x_1 + \ldots + w_nx_n$
unter den Nebenbedingungen

$$\begin{aligned} \sum_{s_i \in T_j} x_i &\geq 1 \qquad j = 1, \ldots, m \\ 0 \leq x_i &\leq 1 \qquad i = 1, \ldots, n \end{aligned} \qquad \text{(LP)}$$

Beschreibung des Algorithmus, von dem in b) nachgewiesen wird, dass es sich um einen k-Approximationsalgorithmus handelt:

---

1. Man löse die LP-Relaxation (LP) mit einem polynomiellen Verfahren (Ellipsoid-Methode oder ein Inneres Punkte Verfahren). Es sei

$$x^* = (x_1^*, \ldots, x_n^*)$$

die auf diese Art erhaltene optimale Lösung von (LP).

2. Das vom Algorithmus gelieferte Ergebnis sei die folgende Teilmenge H von S:

$$H = \left\{ s_i : 1 \leq i \leq n \text{ und } x_i^* \geq \frac{1}{k} \right\}.$$

---

b) <u>Behauptung 1:</u> H ist ein Hitting Set.

<u>Beweis:</u> Wäre H kein Hitting Set, so würde es ein T_j geben mit $H \cap T_j = \emptyset$. Für alle $s_i \in T_j$ würde demnach $x_i^* < \frac{1}{k}$ gelten, woraus man erhalten würde:

$$\sum_{s_i \in T_j} x_i^* < |T_j| \cdot \frac{1}{k} = 1.$$

Dies widerspricht der Tatsache, dass $x^* = (x_1^*, \ldots, x_n^*)$ eine zulässige Lösung von (LP) ist. $\square$

Es sei H^* ein Hitting Set mit Minimalgewicht (optimale Lösung unseres Hitting Set Problems). Wie üblich setzen wir

$$w(H^*) = \sum_{s_i \in H^*} w_i$$

sowie (für H wie oben).

$$w(H) = \sum_{s_i \in H} w_i$$

<u>Behauptung 2:</u> $w(H) \leq k \cdot w(H^*)$.

<u>Beweis</u>: Da $w(H^*)$ der Wert einer optimalen Lösung von (ILP) ist, während $\sum_{i=1}^{n} w_i x_i^*$ der Wert einer optimalen Lösung von (LP) ist, gilt

$$w(H^*) \geq \sum_{i=1}^{n} w_i x_i^* . \qquad (1)$$

Außerdem gilt

$$\sum_{i=1}^{n} w_i x_i^* \geq \sum_{s_i \in H} w_i x_i^* \geq \frac{1}{k} \sum_{s_i \in H} w_i = \frac{1}{k} w(H). \qquad (2)$$

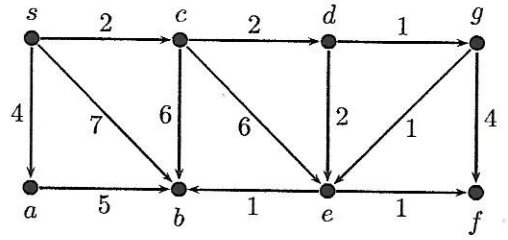
wegen $x_i^* \geq \frac{1}{k}$ für alle $s_i \in H$

Aus (1) und (2) folgt $w(H) \leq k \cdot w(H^*)$. $\square$

Zusammengenommen zeigen die Behauptungen 1 und 2, dass es sich bei dem Algorithmus aus a) um einen k-Approximationsalgorithmus handelt.

L.77

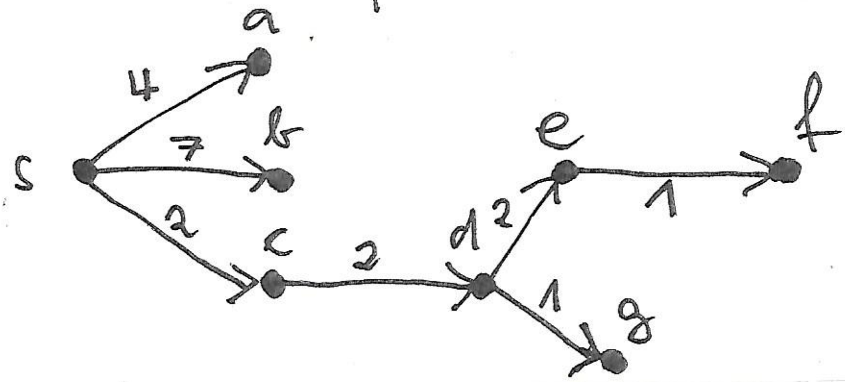
2. a) Der Graph $G = (V, E)$ mit Längenfunktion ℓ sei durch die folgende Zeichnung gegeben:



Verwenden Sie den Algorithmus von Dijkstra (Skript, Seite 181) um für alle $v \in V$ die Länge $d(v)$ eines kürzesten s, v-Pfades zu berechnen. Legen Sie eine Tabelle an, an der man zusätzlich kürzeste s, v-Pfade ablesen kann. Bestimmen Sie auch einen kürzeste-Pfade-Baum.

	s	a	b	c	d	e	f	g
0	0	4 s	7 s	2 s	∞	∞	∞	∞
1	0	4 s	7 s	2 s	4 c	8 c	∞	∞
2	0	4 s	7 s	2 s	4 c	8 c	∞	∞
3	0	4 s	7 s	2 s	4 c	6 d	∞	5 d
4	0	4 s	7 s	2 s	4 c	6 d	9 g	5 d
5	0	4 s	7 s	2 s	4 c	6 d	7 e	5 d
6	0	4 s	7 s	2 s	4 c	6 d	7 e	5 d
7	0	4 s	7 s	2 s	4 c	6 d	7 e	5 d

Anhand der letzten Zeile kann man an jedem $u \neq s$ einen Vorgänger von u auf einem kürzesten s, u-Pfad ablesen. Es ergibt sich der folgende kürzeste-Pfade-Baum:



L.78

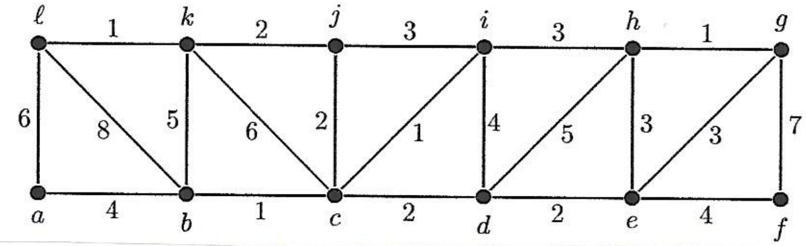
b) Für den folgenden Graphen bestimme man einen minimalen aufspannenden Baum auf drei Arten:

(i) mit dem Algorithmus von Prim (mit Startknoten a);

(ii) mit dem Algorithmus von Kruskal;

(iii) mit dem Reverse-Delete-Algorithmus.

Geben Sie jeweils die Kanten in der Reihenfolge an, in der sie hinzugefügt bzw. weggelassen wurden. (Kommen mehrere Kanten infrage, so wähle man willkürlich eine aus.)



Es wird jeweils nur eine der möglichen Reihenfolgen angegeben; Kanten werden in der Form xy angegeben (anstelle von $\{x, y\}$).

(i) <u>Prim</u>: ab, bc, ci, cj, jk, kl, cd, de, eg, gh, ef.

(ii) <u>Kruskal</u>: kl, bc, ci, gh, cj, jk, cd, de, ih, ab, ef.

(iii) <u>Reverse-Delete</u>: bl, fg, al, ck, bk, dh, id, ij, eh, eg.