

Präsenzübung 11

Besprechung: 21.01.19 – 25.01.19

Aufgabe 1: Branch & Bound (keine Punkte)

In der Vorlesung haben Sie das TSP Problem auf gerichteten Graphen kennengelernt und mit Branch & Bound gelöst. Betrachten Sie folgende Funktion als untere Schranke für TSP für einen Graphen $G = (V, E, c)$ mit $c: E \rightarrow \mathbb{N}$:

$$l(G) = \sum_{v \in V} \min_{w \in V} c(v, w)$$

1. Begründen Sie weshalb $l(G)$ eine untere Schranke für TSP ist.

Betrachten Sie nun den vollständigen Graphen $G = (\{0, 1, 2, 3\}, E, c)$ mit folgender Kantenkostenmatrix:

c	0	1	2	3
0	∞	2	5	8
1	3	∞	4	1
2	2	1	∞	1
3	6	6	1	∞

2. Verwenden Sie Branch & Bound mit der unteren Schranke $l(G)$. Gehen Sie in so wenige *Branches* wie möglich, d.h. benutzen Sie so oft wie möglich den Vorteil der unteren Schranke. Das erreichen Sie, indem sie die Reihenfolge der hinzuzufügenden Knoten (bzw. Kanten) geschickt festlegen.
3. Verwenden Sie Branch & Bound nun so, dass möglichst viele *Branches* besucht und abgelaufen werden.

Aufgabe 2: Minimale Knotenüberdeckung (keine Punkte)

Betrachten Sie das Problem der minimalen Knotenüberdeckung: Für einen ungerichteten Graphen $G = (V, E)$ ist eine in ihrer Größe minimale Teilmenge von Knoten $S \subseteq V$ gesucht, sodass für jede Kante $(u, v) \in E$ mindestens einer der beiden Knoten u, v in S ist. Dazu sei folgender Algorithmus gegeben.

```
function APPROX-VERTEX-COVER(G)
   $C = \emptyset$ 
   $E' = G.E$ 
  while  $E' \neq \emptyset$  do
    Sei  $(u, v)$  eine beliebige Kante aus  $E'$ .
     $C = C \cup \{u, v\}$ 
    Entferne jede zu  $u$  oder  $v$  inzidente Kante aus  $E'$ .
  end while
  return  $C$ 
end function
```

- a) Geben Sie einen Beispielgraphen an, für den APPROX-VERTEX-COVER immer eine suboptimale Lösung liefert.
- b) Beweisen Sie, dass die Kanten, die in Zeile 4 von APPROX-VERTEX-COVER gewählt werden, ein maximales Matching im Graph G bilden. Ein Matching eines Graphen $G = (V, E)$ ist eine Teilmenge der Kanten $M \subseteq E$, sodass je zwei Kanten in M keinen gemeinsamen Knoten enthalten.

Aufgabe 3: Online Algorithmen (keine Punkte)

Das Paging Problem ist ein online Minimierungsproblem. Sei k die Anzahl der Internetseiten, die im Zwischenspeicher gespeichert werden können. Zu Beginn ist der Zwischenspeicher mit den ersten k Seiten $B_0 = \{p_1, \dots, p_k\}$ befüllt. Es gebe m verschiedene Seiten. Die Inputsequenz sei $I = \{x_1, \dots, x_n\}$, wobei die Seite $x_i \in \{p_1, \dots, p_m\}$ im i -ten Schritt aufgerufen wird. Falls eine Seite x_i im Schritt i aufgerufen wird und die Seite sich nicht im momentanen Zwischenspeicher befindet, muss der Algorithmus eine Seite p_j mit der angefragten Seite x_i im Zwischenspeicher ersetzen. Das Ersetzen einer Seite erzeugt die Kosten $y_i = 1$, ansonsten ist $y_i = 0$. Das Ziel ist, die Anzahl der Tausch-Operationen im Zwischenspeicher zu minimieren. Wir minimieren also $\text{cost} = \sum_{i=1}^n y_i$.

- a) Nehmen Sie an, Ihnen wird das Paging Problem als offline Problem gestellt. Sie kennen also alle Anfragen I vorher. Beschreiben Sie, wie Ihr Algorithmus die optimale Lösung findet. Wir bezeichnen die Kosten der optimalen Lösung als OPT.
- b) Der Zwischenspeicher sei auf drei verschiedene Weise realisiert: A1) als Stack, A2) als Queue, A3) als Algorithmus, der diejenige Seite ersetzt, die am seltensten benutzt wurde. Leiten Sie für alle drei Fälle Instanzen eines Paging Problems mit $m \geq 5$ her, sodass $\text{cost}(A_i) \geq k \cdot \text{OPT}$ gilt.