Schubert Julian, Gruppe 4
Philipp Wahl, Gruppe 4

Abgabe zum 3. Übungsblatt (AGT 21)

Aufgabe 1:

Wir lösen die Aufgabe mit CPLEX:

```
$ cat task.mod
dvar float+ x;
dvar float+ y;
dvar float+ z;

maximize 3.6 * x - 1.7 * y + 2.2 * z;

subject to {

        250 * x - 480 * y + 150 * z <= 18000;
        475 * x           + 300 * z <= 84000;
        180 * x + 75  * y           <= 25000;
        400 * x + 250 * y + 260 * z <= 60000;
}

execute {
        writeln("x: ", x);
        writeln("y: ", y);
        writeln("z: ", z);
}
$ oplrun task.mod

<<< setup


<<< generate

Tried aggregator 1 time.
No LP presolve or aggregator reductions.
Presolve time = 0,00 sec. (0,00 ticks)

Iteration log . . .
Iteration:     1   Scaled dual infeas =      0,000000
Iteration:     2   Dual objective     =      543,802632

<<< solve


OBJECTIVE: 418.4825
```

```
x: 126.179
y: 30.5046
z: 7.31691

<<< post process


<<< done
```

Die optimale Lösung lautet also:
x: 126.179
y: 30.5046
z: 7.31691

Aufgabe 2:

a) Zunächst muss für alle Städte gelten, dass die Summe aller Kosten die eine Stadt bezahl nicht größer sein darf als das Budget dieser Stadt.

$$\forall v \in V : \sum_{\text{Zahlungen der Stadt v}} \leq \text{Budget der Stadt v}$$

Zusätzlich muss aber für jede Straße gelten, dass der Teil den die eine Stadt an Kosten trägt addiert mit den Kosten die die andere Stadt trägt gleich den Reperaturkosten für die jeweilige Straße ist

$\forall s \in E$: Teil den Stadt1 bezahlt+Teil den Stadt2 bezahlt = Kosten Reperatur für s

Wir minimieren nun einfach nach der Summe der Zahlungen aller Städte und erhalten unsere Zielfunktion

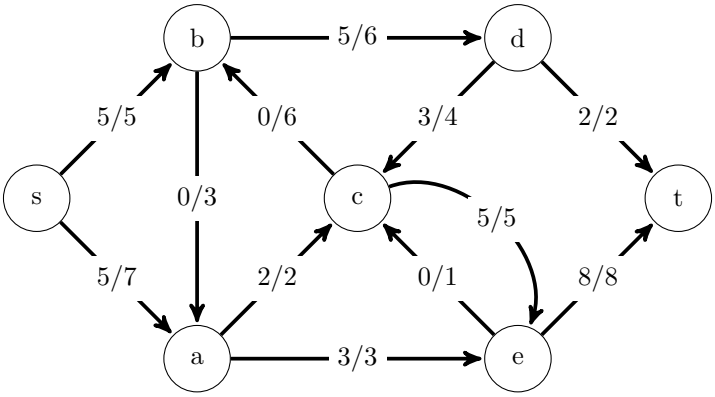
$$\text{minimize} \sum_{v \in V} \sum_{\text{Zahlungen der Stadt v}}$$

Unsere Lösung ist korrekt da unsere erste Nebenbedingung dafür sorgt, dass keine Stadt mehr bezahlt als ihr budget. Die zweite Nebenbedingung sorgt dafür, dass die Kosten von für jede Straße gedeckt sind.

b) Wir verwenden hier den selben Aufbau wie in Teilaufgabe a, allerdings Ändern wie die zweite Bedingung wie folgt ab:
Die Kosten für die Reperatur der Straße s wird nun gleichgesetzt mit x multipliziert mit den Kosten die Stadt 1 trägt und dazu y mal die Kosten die Stadt zwei trägt hinzuaddiert. Wir fordern zusätzlich das x und y natürliche Zahlen (inklusive Null) sind und das x + y = 1 ergibt. Das stellt sicher das entweder x = 0 y = 1 oder x = 1 y = 0 gelten muss, die Kosten für die Reperatur der Straße kann nun also nur noch von einer der beiden Städte getragen werden.
Unsere Lösung ist ein ganzzahliges lineares Programm da wir keine Kommazahlen benötigen.

c) Wir zeigen zunächst wie aus unserer Lösung ein Flussnetzwerk modelliert werden kann:
Zunächst erstellen wir für jede Stadt einen Knoten, dann verbinden wir die Städte mit den Straßen durch ungerichtete Kanten. Das Gewicht der Kanten ist dabei die Kosten die Reperatur der jeweiligen Straße.
Dann Fügen wir einen Startknoten s hinzu, dieser hat zu jeder Stadt eine gerichtete Kante (Start: s, Ende: Stadt). Das Gewicht dieser Kante ist dabei das Budget der Jeweiligen Stadt.
Wir erstellen noch einen Endknoten t, der noch keine Kanten besitzt.
Nun ersetzen wir die ungerichteten Katnen zwischen zwei Städten nach folgendem Schema: Wir erstellen jede die ungerichtete Kante zwischen zwei Städten einen Hilfsknoten h. Anschließend erstellen wir gerichtete Kanten von beiden ursprünglichen Städten zum Hilfsknoten h, die kapazität der Kante ist dabei beide male die Kosten für die Reperatur der Straße. Dann erstellen wir noch eine gerichtete Kante vom Hilfspunkt h zum Endknoten t, die Kapazität der Kante ist dabei wieder die Kosten der Reperatur der Straße.
Nun berechnet man den maximalen s-t-Fluss. Ist der Wert dieses Flusses gleich der Summe aller Straßenreperaturen existiert eine zulässige Verteilung (die genau dem Fluss entspricht). Die Rückrichtung ist hier sehr einfach indem man alle Schritte Rückwährts ausführt, man also die gerichteten Kanten durch die entsprechenden Ungerichteten ersetzt und s, t sowie alle Hilfspunkte entfernt.
Unsere Lösung ist korrekt da durch die Kanten ausgehend von s zu den Städten sicher gestellt wird, dass keine Stadt mehr als ihr Budget zulässt bezahlen kann. Die Kosten für alle Reperaturen werden ebenfalls korrekt erfasst da diese sich im maximalen s-t-Fluss befinden.

Aufgabe 3:



Der Maximale Wert des s-t-Flusses ist hierbei 10.

b) Da die Kanten die in den Knoten t gehen eine maximale Kapazität von 10 haben, kann es keinen s-t Fluss mit einer größeren Kapazität als der von uns angegebene Fluss haben (unser Fluss hat bereits den höchsten Möglichen Wert)

Aufgabe 4

a) Hat man einen einfachen Weg der länge k kann man daraus wie folgt einen s-t-Weg in G' bilden:
Man Startet am Knoten s (den wir zu G hinzu gefügt haben), und gehen dann zum ersten Knoten des einfachen Weges der länge k in G. Diese Verbindung muss existieren da s (und t) jeweils zu jedem Knoten in G adjazent sind. Dann geht man den Weg weiter (wie in G), am letzen Knoten geht man dann noch zu t. Auch diese Verbindung muss existieren da auch t adjazent zu allen Knoten in G ist. So kann man also aus einem Weg der Länge k in G einen s-t-Weg der Länge k + 2 in G' bilden.
Rückrichtung: Man entfernt einfach s und t aus dem s-t-Weg (inklusive der Kanten zu s bzw t) und erhält somit direkt den Weg der länge k - 2 in G aus dem orginalen s-t-Weg in G'

b) Hinrichtung:
Angenommen es gibt einen einfachen s-t-Weg der länge n+1 in G', dann kann man daraus einen Hamiltonweg bilden. Dies ist gegeben da ein Hamiltonweg ein einfacher s-t-Weg (wobei s der Start des Weges und t das ende des weges ist) ist, der die Länge n - 1 hat. Wie man dann aus dem s-t-Weg den Hamiltonweg konstruieren kann haben wir bereits in Aufgabe 4a gezeigt.
Rückrichtung
Aus einem Hamiltonkreis lässt sich ein s-t-Weg der Länge n+1 in G' bilden. Auch hier nutzen wir die Eigenschaft das ein Hamiltonweg ein einfacher Weg (bzw s-t-weg mit start s und ende t des Weges) mit länge n - 1 ist. Auch hierzu haben wir in Aufgabe 4a gezeigt wie sich aus dem s-t-weg der Länge n -1 ein s-tWeg der Länge n+1 in G' bilden lässt. Dies funktioniert da ein s-t-weg der länge n + 1 alle Knoten im Graphen besucht (wenn n die anzahl der Knoten in G ist)

c) Wenn das längste Pfad Problem in polynomialzeit lösbar wäre, könnte man zu jedem ungerichtetem graphen zwei knoten s und t hinzufügen, welceh mit jedem Knoten verbunden wären, und dann einen längsten pfad berechnet, der mit s startet und mit t endet. Diese könnten nun wieder entfernt werden um einen hamilton weg in polynomialziet zu finden. Da jedoch hamilton np-schwer ist, kann dies nicht sein woraus resultiert das gegebenes Problem mindestens np schwer ist. Da ein hamilton weg in np berechenbar ist ist es offensichtlich möglich einen längsten weg in np zu berechnen, da ein hamitlon kreis bereits ein längster einfacher weg ist. Also muss das längste wege Problem genau np-schwer sein.