OPERATIONS RESEARCH
KACKERTSTR. 7
52072 AACHEN

Prof. Dr. Marco Lübbecke

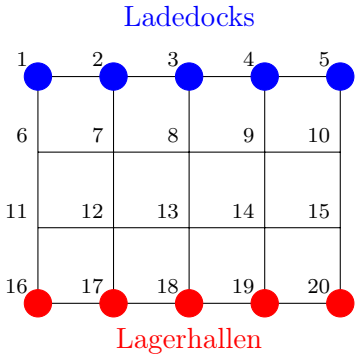
Michael Bastubbe
Dr. Matthias Walter

Praktische Optimierung mit Modellierungssprachen
Aufgabenblatt 6

Abgabe: für eine 1,0 bis spätestens Freitag, den 15.07.2016 um 23:59 Uhr,
über das Online-Abgabesystem)

Aufgabe 6 (Kollisionsfreies Routen)

In einem Hafen sollen Container von den Ladedocks zu den Lagerhallen bzw. von den Lagerhallen zu den Ladedocks transportiert werden. Der Transport erfolgt mithilfe von ferngesteuerten Vehikeln, die sich zwischen Ladedocks und Lagerhallen entlang der Kanten eines Gittergraphen bewegen bzw. auf dessen Knoten warten dürfen. Von diesem Gittergraphen sind sowohl die „Gitterbreite“ `width` als auch die „Gitterhöhe“ `height` bekannt. Die Knoten sind konsekutiv benannt, wie aus der folgenden Abbildung eines Gittergraphen mit Gitterbreite 5 und Gitterhöhe 4 hervorgeht:



Es sind n Transportaufträge gegeben jeweils durch einen Startknoten s_i, einen Zielknoten t_i, eine Startzeit r_i und eine Zielzeit d_i, $i = 1, \dots n$. Zur jeweiligen Startzeit erscheint der jeweilige Container auf einem Transportvehikel im jeweiligen Startknoten. Es darf in Knoten, aber nicht entlang von Kanten gewartet werden. Jeder Container muss sich genau zur Zielzeit im Zielknoten befinden (wenn ein Container den Zielknoten beispielsweise eine Zeiteinheit zu früh erreicht, müsste er noch eine Zeiteinheit im Zielknoten warten). Nach der Zielzeit verschwindet der Container (inklusive Transportvehikel). Es darf zu keinen Kollisionen kommen, d.h. zu keinem Zeitpunkt darf sich in einem Knoten mehr als ein Container befinden. Außerdem darf während der Befahrung einer Kante die Gegenrichtung nicht befahren werden.

Weiterhin gibt es auf den Kanten Fahrzeiten und Kosten, die als 1 angenommen werden. Die Kosten entstehen sobald ein Container über die jeweilige Kante transportiert wird. Das Warten in einem Knoten ist kostenlos. Es sollen nun für alle Transportaufträge Routen bestimmt werden, so dass die obigen Bedingungen eingehalten und die Gesamtfahrkosten minimal sind. Betrachtet werden sollen die Zeitpunkte $1, \dots, T$ mit $T = \max_{i=1,\dots,n} d_i$.

Im L^2P findet Ihr die beiden Instanzen `routingdata1.py` und `routingdata2.py`, deren optimale Zielfunktionswerte 4 bzw. 16 betragen. In den Instanzdateien sind jeweils n, `height` und `width`, sowie die Listen `startNodes`, `targetNodes`, `startTimes` und `endTimes` gegeben, an denen jeweils an der i-ten Stelle s_{i+1}, t_{i+1}, r_{i+1} bzw. d_{i+1} stehen, $i = 1, \dots n$.

Entwickelt ein IP, welches das beschriebene Problem modelliert. Entwickelt python-Code, der die Instanzen einlesen und euer IP als gurobi-Modell baut. Hochzuladen ist eine Datei `routingmodel.py`, in der eine Methode `solve`(n, `width`, `height`, `startNodes`, `targetNodes`, `startTimes`, `endTimes`) implementiert ist, die euer gurobi-Modellobjekt **zurückgibt!**. Dabei ist weiterhin zu beachten:

- das Modell soll mindestens die folgenden Binärvariablen beinhalten, deren Namen folgendermaßen gewählt werden müssen: "$x\_i\_j\_t$" wobei $i \in \{1, \dots, n\}$ (Indizes der Transportaufträge), $j \in \{1, \dots, \texttt{height} \cdot \texttt{width}\}$ (Knotenindizes) und $t \in \{1, \dots, T\}$ (Zeitpunkte) sind, dabei soll der Wert von $x\_i\_j\_t$ genau dann den Wert eins besitzen, wenn sich das Vehikel von Auftrag i zum Zeitpunkt t im Knoten j befindet

Ihr sollt Eure gefunden Lösung anschließend visualisieren. Dabei habt ihr freie Hand, wie eure Visualisierung im Detail aussieht, bis auf die folgenden Punkte:

- der Gittergraph soll grafisch dargestellt sein; wie das prinzipiell aussehen **könnte**, habt ihr oben gesehen
- jeder Transportauftrag soll eine andere Farbe bekommen ($n = 10$ sollte möglich sein)
- die Transportwege der Aufträge sollen nachvollziehbar sein
- wenn sich die Knotenfolgen zweier Transportaufträge schneiden, soll ersichtlich sein, warum es sich um keine Kollision handelt (z.B. indem ihr die Zeiten mit darstellt oder indem ihr für jeden Zeitpunkt eine Momentaufnahme plottet oder indem ihr eine kleine Animation baut oder ...)
- die Darstellung soll übersichtlich und eindeutig sein, legt auf jeden Fall eine Legende an, die eure Darstellung leicht verständlich macht
- als pdf vorliegen, falls Ihr andere Wünsche habt, meldet euch bitte bei uns
- ladet die Visualisierung der zweiten Instanz bitte über das Abgabesystem hoch
- der Code in `routingmodel.py` soll keine Visualisierung erzeugen, sondern nur die oben beschriebene `solve()`-Methode implementieren

Als optimale Zielfunktionswerte solltet Ihr 4 bzw. 16 als optimale Zielfunktionswerte herausbekommen. Bitte ladet Eure Datei und die Visualisierung der zweiten Instanz über das Online-Abgabesystem bis spätestens Freitag, den 15.07., um 23:59 Uhr hoch.

Viel Erfolg!