OPERATIONS RESEARCH
KACKERTSTR. 7
52072 AACHEN

Prof. Dr. Marco Lübbecke

Michael Bastubbe

Praktische Optimierung mit Modellierungssprachen
Aufgabenblatt 4

Abgabe (für eine 1,0): bis spätestens Freitag, den 24.06.2015 um 23:59 Uhr,
über das Online-Abgabesystem

Aufgabe 4 (Scheduling von Flugzeuglandungen)

In dieser Programmieraufgabe soll das Problem des Planens von Flugzeuglandungen betrachtet werden. Gegeben seien eine Menge von Flugzeugen $\{1, \dots, n\}$, die auf einem Flugplatz mit einer Landebahn landen sollen. Für jedes Flugzeug $i \in \{1, \dots, n\}$ gibt es einen frühestmöglichen Landezeitpunkt r_i, einen bestmöglichen Landezeitpunkt b_i, und einen spätestmöglichen Landezeitpunkt d_i. Des Weiteren gibt es Strafkosten g_i für jede Zeiteinheit, die Flugzeug i vor dem bestmöglichen Zeitpunkt b_i landet und Strafkosten h_i für jede Zeitenheit, die es später als b_i landet. Außerdem müssen zwischen den Landungen gewisse Pufferzeiten eingehalten werden. Genauer gesagt, ist für jedes Paar von Flugzeugen $i, j \in \{1, \dots, n\}$ eine Pufferzeit s_{ij} gegeben, die (sofern i vor j landet) zwischen der Landung von i und der Landung von j verstreichen muss. Ziel ist es für jedes der Flugzeuge einen Landezeitpunkt festzulegen, so dass der Landezeitpunkt von Flugzeug i innerhalb des Intervalls $[r_i, d_i]$ liegt, die Pufferzeiten eingehalten werden und die Summe der Strafkosten minimal ist.

Die Instanzen für dieses Problem findet Ihr unter

`http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/` (stellt euch vor es wäre der Flughafenbetreiber).

Wir interessieren uns für `airland`$\{1, \dots, 8\}$`.txt` .

Das Format der Dateien wird euch vom Flughafenbetreiber wie folgt beschrieben:

```
n T
for every plane i (i=1,...,n):
  a_i r_i b_i d_i g_i h_i
  for every plane j (j=1,...,n): s_{ij}
```

wobei auf Eure Nachfrage der Flughafenbetreiber T als Ortstemperatur am Flughafen und a_i als Turbinentyp des Flugzeugs vorstellt.

Entwickelt ein IP, welches das beschriebene Problem modelliert. Entwickelt python-Code, der die Instanzen einlesen und euer IP als gurobi-Modell baut. Hochzuladen ist Code, der Instanzdateien einlesen kann, sowie das gurobi-Modell baut (mit dem ihr das Problem gelöst habt) und zurückgibt. Genauer gesagt ein Zip-archiv, das (mindestens) eine Datei `runscheduling.py` enthält, in der eine Methode `solve(full_path_instance)` implementiert ist, die euer gurobi-Modellobjekt **zurückgibt!**, wobei `full_path_instance` der Pfad zur Instanzdatei (als String) ist. Dabei ist weiterhin zu beachten:

- das Modell soll mindestens die folgenden Variablen beinhalten, deren Namen folgendermaßen gewählt werden müssen: "x_i" wobei $i \in \{1, \ldots, n\}$ (Indizes der Flugzeuge) sind, dabei soll der Wert von x_i genau dem Landezeitpunkt von Flugzeug i entsprechen
- stellt sicher, dass die Dateien im Archiv ausreichen um durch Aufruf der Methode `solve(full_path_instance)` die Instanz einzulesen und euer Modell zu returnen

Als optimale Zielfunktionswerte solltet Ihr für die ersten vier Instanzen 700, 1480, 820 bzw. 2520 erhalten. Bitte ladet Euer Archiv über das Online-Abgabesystem bis spätestens Freitag, den 24.06., um 23:59 Uhr hoch.

Viel Erfolg!