1. Programmieraufgabe

Abgabetermin: Zwischen dem 16.1.2015 und 23.1.2015

Das Ziel der Programmieraufgabe ist es den Netzwerk-Simplex-Algorithmus zu implementieren. Als Programmiersprachen sind PYTHON, JAVA, C++, C zugelassen. (Auf Anfrage kann die Liste gerne noch erweitert werden.)

Eine Menge von Beispielinstanzen werden auf der Website zur Vorlesung bereitgestellt. Dies ist eine Teilmenge der Testdaten von

`http://elib.zib.de/pub/Packages/mp-testdata/mincost/`.

Information zum Datenformat sind an gleicher Stelle unter

`http://elib.zib.de/pub/Packages/mp-testdata/mincost/netg/info`

zu finden.

Teilaufgaben

1. **Einlesen der Daten**
2. **Implementierung Netzwerksimplex** Es bietet sich an, die in der Übung vorgestellte Datenstruktur des 3er Arrays d, s, p für eine Baumlösung zu nutzen, siehe auch [Chv83, AMO93].

   Vergesst nicht die Implementierung einer geeigneten Strategie um Cycling zu vermeiden.
3. **Auswertung** Schreibt eine kurze Auswertung (**EINE A4 Seite**) zu eurem Programm. Diese soll beinhalten:

   - Umgebung der Testrechnungen (CPU, RAM, Beriebssytem, Programmiersprache),
   - Implementierungsdetails wie zum Beispiel Auswahlregeln der Bögen, spezielle Datenstrukturen oder genutzte Bibliotheken,
   - Ergebnistabelle die mindestens die Einträge: „erreichter Zielfunktionswert“ und „benötigte Rechenzeit“ für jede Instanz enthält.

Abgabe Für die Programmieraufgabe gibt es keine Punkte. Für die erfolgreiche Teilnahme an der Vorlesung muss sie erfolgreich bearbeitet werden. Die gegebenen Probleminstanzen müssen optimal gelöst oder die Unzulässigkeit erkannt werden. Jedes Gruppenmitglied muss in der Lage sein den Code zu erklären. Die A4 Seite zur Auswertung schickt ihr bitte bis zum 23.1.2015 an klug@zib.de. Vereinbart frühzeitig einen Termin zur Endabnahme zwischen dem 16.1 und 23.1.2015.

Tipps Nutzt ihr die in der Übung vorgestellte Datenstruktur, braucht ihr im Grunde keine extra Datenstruktur für Graphen. Um Fehler aufzuspüren, ist es jedoch sehr hilfreich, wenn man sich eine Baumlösung direkt anschauen kann. Für Python können zum Beispiel die Bibliotheken Networkx(`https://networkx.github.io`) und matplotlib (`http://matplotlib.org/`) genutzt werden. Programmiersprachenunabhängig können Graphen über eine Textdatei mit Graphviz `http://www.graphviz.org/` visualisiert werden.

Erstellt euch kleine Beispielinstanzen und fangt nicht gleich mit den größten an.

Viel Erfolg.

Literatur

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows, Theory, Algorithms and Applications*. Paramount Publishing International, Prentice Hall, New York, 1993.

[Chv83] Vasek Chvátal. *Linear Programming*. Freeman, 1983.