Technische Universität München | SS 1999
Institut für Informatik | Merkblatt 3
Prof. Dr. Angelika Steger | 11. Mai 1999

Praktikum Diskrete Optimierung

Best-First-Suche

1 Allgemeines zur Schnittstelle

Bei allen folgenden Praktikumsaufgaben sollen Ihre Lösungen dieselbe Schnittstelle zur Verfügung stellen. Dem Programm `foo` werden an der Kommandozeile zwei Argumente übergeben:

```
foo <max. execution time in min> <basename>
```

Im Folgenden nehmen wir an, daß der *Basename* gleich `data` sei. Das Programm liest in diesem Fall die Eingabedatei `data.def` und erzeugt eine Datei `data.sol` mit einer (möglichst guten) Lösung. Wenn die angegebene maximale Ausführungszeit erreicht ist, sollte sich das Programm beenden (ggf. mit einer suboptimalen Zwischenlösung).
Die Datei `data.def` könnte z. B. folgendermaßen aussehen:

```
3
100   200
30.5  120
0     73
143.3 18.1 200
18.1  0    74.3
200   74.3 0
```

Diese Datei definiert ein Netzwerk mit drei Knoten (siehe erste Zeile). Danach werden die *Koordinaten* der Knoten angegeben. Hierauf folgt eine 3×3-Matrix mit dem *Verkehrsaufkommen*. Sie können davon ausgehen, daß diese Matrix symmetrisch ist. Die Einträge in der Diagonale sind für unser Problem irrelevant.
Eine zugehörige Lösung `data.sol` wäre

```
0 1 2 0 1
0 2 2 0 2
1 2 3 1 0 2
```

Jede Zeile entspricht hierbei einem Pfad. Die ersten zwei Zahlen geben die Indizes der Knoten an, zwischen denen der Pfad verläuft. Dabei sollte der kleinere Index

immer zuerst kommen. Danach folgt die Länge des Pfades und eine Liste aller Knoten, die der Pfad enthält, also auch Start- und Endknoten. Die letzte Zeile gibt also beispielsweise an, daß zwischen Knoten 1 und 2 ein Pfad mit 3 Knoten verläuft, nämlich $(1, 0, 2)$.
Ihre Einleseroutinen sollten so stabil sein, daß sie nicht von der Anzahl der Leerzeichen zwischen den Einträgen oder von der exakten Zeilenstruktur abhängen.

2 Best-First-Suche

Die *Best-First-Suche* (auch *Hill-Climbing* genannt) ist mit das einfachste Suchverfahren. Ausgehend von einer aktuellen Position im Suchraum – in unserem Fall eine bestimmte Belegung der Pfade – werden alle lokalen Veränderungen betrachtet und diejenige mit der größtmöglichen Verbesserung übernommen. Dies wird wiederholt bis es in der lokalen Nachbarschaft keine Verbesserungsmöglichkeiten mehr gibt. Es handelt sich bei Best-First also um eine sogenannte *lokale Suche*. Diese zeichnet sich dadurch aus, daß der Suchraum mit kleinen, möglichst zielgerichteten Schritten abgesucht wird.
Die Suche kann bespielsweise mit einem Netzwerk gestartet werden in dem alle Pfade nur die beiden entsprechenden Endknoten enthalten. Im nächsten Abschnitt wird eine mögliche Nachbarschaft dargestellt.
Die Geschwindigkeit und Güte von Best-First ist stark abhängig von der gewählten Nachbarschaftsrelation. Ein allgemeines Problem von Best-First ist allerdings, daß es meist in sogenannten *lokalen Minima* steckenbleibt. Bei einem lokalen Minimum ist in der direkten Nachbarschaft keine Verbesserung zu erzielen, es existieren aber andere, bessere Lösungen. Man ist natürlich daran interessiert, daß globale Minimum zu finden. Besonders bei großen Problemen ist die Zielfunktion meist sehr „zerklüftet“, d.h. es gibt sehr viele lokale Minima.

3 Mögliche lokale Nachbarschaft

In unserem Fall könnte ein lokaler Nachbar folgendermaßen definiert werden: Bei genau einem Pfad des aktuellen Netzwerkes wird entweder ein Knoten ...

- ... gelöscht,
- ... verschoben,
- ... oder eingefügt.

Die lokale Nachbarschaft eines Netzwerkes wären also sämtliche Netzwerke die durch jeweils eine solche Änderung eines Pfades entstehen.
Die Nachbarschaft eines Knotenpaares könnte man im einfachsten Fall als LEDA-Liste von Pfaden speichern.