Programmierpraktikum Diskrete Optimierung

Kürzeste Wege zwischen Hindernissen

SoSe 2010

Homepage der Veranstaltung:
`http://www.or.uni-bonn.de/lectures/ss10/proprakt_ss10.html`

Kürzeste-Wege-Probleme gehören zu den grundlegenden Problemen der Optimierung. Meistens werden dabei Wege in einem vorgegebenen Graphen gesucht. In diesem Programmierpraktikum wird dagegen das Problem betrachtet, zwischen zwei Punkten in der Ebene einen kürzesten Weg zu berechnen, wobei man bestimmten rechteckigen Hindernissen ausweichen muß. Dieses Problem muß z.B. beim Entwurf von modernen Computerchips millionenfach gelöst werden, daher werden sehr schnelle Algorithmen und effiziente Implementierungen benötigt. In diesem Praktikum sollen einige Verfahren zur Lösung dieses Problems implementiert und auf großen Instanzen aus der Praxis getestet werden.

Allgemeine Bemerkungen:

Wir betrachten rectilineare Pfade und Hindernisse. Die Rechtecke, die wir betrachten, sind offene[1] achsenparallele Rechtecke im $\mathbb{R}^2$, d.h. ein Rechteck c ist gegeben als Punktmenge

$$c =]x_{\min}(c), x_{\max}(c)[\times]y_{\min}(c), y_{\max}(c)[.$$

Seien n Hindernisrechtecke $c_i, i = 1, \ldots, n$, gegeben, dann definiert

$$\mathcal{B} := \bigcup_{i=1}^{n} c_i$$

die Hindernisregion. Wir nennen einen Punkt $p \in \mathbb{R}^2$ *legal*, wenn er $\mathcal{B}$ nicht schneidet.

Ein rectilinearer Pfad besteht nur aus achsenparallelen Segmenten, wir nennen einen Punkt, an dem ein Pfad von horizontal auf vertikal wechselt (oder umgekehrt) einen *Knick*. Ein Pfad kann also eindeutig durch die Angabe seiner Knicke beschrieben werden. Ein Pfad ist legal, wenn er kein Segment $\mathcal{B}$ schneidet.

Ziel wird es nun sein, gegeben $\mathcal{B}$ und zwei Punkte $s, t \in \mathbb{Z}^2$, einen kürzesten legalen Pfad von s nach t zu finden. Für die Rechtecke werden nur ganzzahlige Koordinaten angenommen.

Die Programme des Praktikums sind in ISO-C oder C++ zu schreiben, und sollen mit aktuellen gnu-Tools kompilierbar sein.

Aufgaben:

Ganz allgemein soll das rectilineare kürzeste Wege Problem zwischen rectilinearen Blockaden gelöst werden. Hierfür gibt es 3 unterschiedliche Verfahren, die von je einer Gruppe gelöst werden sollen. Die Implementierungen sollen natürlich auch die in den Arbeiten dazu angegebenen asymptotischen Laufzeiten haben.

Für jede der Aufgaben ist zunächst eine Einarbeitungsaufgabe zu bearbeiten, für die es eine Zwischenbesprechung geben wird. Hier sollen auch Fragen zur Bearbeitung und Probleme besprochen werden. Sie können sich

[1] Es ist also erlaubt, am Rand eines Rechtecks einen Pfad zu erstellen, sofern man nicht durch das Innere eines weiteren Rechtecks l¨auft.

natürlich auch jederzeit mit Fragen und Problemen melden. Die Einarbeitungsaufgabe soll bis zum 26. April 2010 abgegeben werden. Ein Termin für die endgültige Abgabe und eine entsprechende Präsentation wird dann auch festgelegt.

Gruppe 1: Wave Front Approach

Implementieren Sie den Wave-Front Ansatz von Mitchell.

Einstiegsaufgabe

Implementieren Sie einen Kürzesten-Wege-Algorithmus mit dem Maze-Running-Ansatz. Da alle Koordinaten der Rechtecke ganzzahlig sind, dürfen Sie annehmen, dass Sie entsprechend $\mathbb{Z}^2$ als zugrundeliegendes Gitter haben.

Gruppe 2: Visibility Graph

Implementieren Sie eine Pfadsuche auf dem Visibility-Graph von Clarkson, Kapoor und Vaidya. Benutzen Sie für die Pfadsuche auf diesem Graph den Ansatz von Fredman und Tarjan, der auf diesem Graph effizient ist.

Einstiegsaufgabe

Erstellen Sie ein Hanangitter und lösen Sie das Kürzeste-Wege-Problem auf diesem Graphen.

Gruppe 3: Track Graph

Implementieren Sie eine Pfadsuche basierend auf dem Track-Graphen von Yang, Lee und Wong.

Einstiegsaufgabe

Erstellen Sie ein Hanangitter und lösen Sie das Kürzeste-Wege-Problem auf diesem Graphen.

Datenformat und Testinstanzen:

Testinstanzen finden Sie auf der Homepage der Veranstaltung. Die Eingabe der Start- und Zielpunkte sowie der Blockaden wird als Textdatei gegeben.

Die erste Zeile kodiert den Startpunkt, die zweite Zeile den Zielpunkt. In der dritten Zeile steht die Anzahl an Rechtecken, die dann angegeben werden. Jede dieser Zeilen kodiert genau ein Rechteck $c \subset \mathbb{R}^2$ in der Form $x_{\min}(c)x_{\max}(c)y_{\min}(c)y_{\max}(c)$. Die Daten sind durch Leerzeichen getrennt. Sie dürfen ferner annehmen, dass alle Koordinaten der Rechtecke ganzzahlig sind, und es keine 1-dimensionalen Rechtecke (Linien) in der Instanz gibt.

Das Programm soll so aufgerufen werden, dass man den Namen der Instanzdatei mit übergibt. Wenn also `program` ein Programm zur Lösung einer (Teil-)Aufgabe ist, und `datei` die Instanz kodiert, so ist der Aufruf:
`program datei`