Universität Bremen

Nicole Drechsler
Rüdiger Ebendt
Email: nd@tzi.de
ebendt@tzi.de

Bremen, 16. Dezember 2005

Weihnachtspäckchen für

ExPlayN

Aufgabe 1 – Eine A$^{\text{Weihnachts-}}$*geschichte

In der Vorweihnachtszeit ist es in unseren Landen üblich, viele kleinere oder auch größere Besorgungen zu tätigen. Um dabei stets erholt und fröhlich zu sein ist es unerlässlich, beim Einkaufsbummel die Wegstrecken, die zwischen den Geschäften zurückgelegt werden müssen, im Blick zu haben …
Das geschulte InformatikerInnenauge erkennt natürlich sofort: „Das ist doch das *Handlungsreisendenproblem*, kann man da nicht A* nehmen?“

Ja genau! Und wer die Problembeschreibung nicht mehr ganz präsent hat, der kann sie hier auspacken:
http://www.informatik.uni-bremen.de/agra/doc/lehrmat/wise0506/heuopt/2_optimierung.pdf

Jetzt benötigt ihr noch eure C++-Kenntnisse und los geht’s mit der Implementierung des A*-Algorithmus, der das Wegeproblem löst.

Probleminstanzen zum Testen eures A$^{\text{Weihnachts-}}$*-Algorithmus gibt es beispielsweise unter:
http://www.tsp.gatech.edu/ oder
http://www.sintef.no/static/am/opti/projects/top/vrp/benchmarks.html

Aufgabe 2 – Weihnachtsgraph®en: Ein Geschenk für den Bremer Weihnachtsmann

Vor einigen Jahren erregte eine Publikation über die Unmöglichkeit der Existenz eines Weihnachtsmannes Aufsehen. Es wurde argumentiert, dass der arme Kerl pro Sekunde 976,7 Bescherungen leisten müsse, daher mit 0,33 Prozent der Lichtgeschwindigkeit reisen müsse, was seinen sicheren Tod bedeuten würde.

Nun, wir wollen diese Diskussion nicht unbedingt vertiefen, so interessant sie an sich auch sein möge ☺ Als wahrscheinlich gilt allerdings die regionale Organisation einer ganzen Schar von Weihnachtsmännern, da es offenbar mehr als einen geben muss. Als sicher angesehen werden kann zudem, dass jeder Weihnachtsmann immer noch einen recht harten Job hat und sich über schnelle Algorithmen zur Planung einer optimalen Tour durch N Wohnungen mit zu

Beschenkenden in z.B. Bremen sicher freuen würde. Lasst uns also dem Bremer Weihnachtsmann unter die Arme greifen – unser Geschenk an den Weihnachtsmann!

Wie ihr sicher gemerkt habt, könnt ihr Euer Programm zur Aufgabe 1 hier wiederverwenden – es geht algorithmisch um das selbe Problem. Diesesmal wollen wir uns aber besondere Gedanken um die Wahl der Repräsentation im Zustandsraum machen und herausfinden, welchen Effekt das auf die Laufzeit unseres Verfahrens hat.

Dazu ein paar ***Anmerkungen***: Gesucht ist ja ein kürzester (billigster) Zykel, der o.B.d.A. bei der ersten Wohnung, also Wohnung 1 beginnt. Vielleicht hat ja bereits in Eurer Implementation in Aufgabe 1 ein Knoten des Zustandsraum einen *Teilweg* repräsentiert, der bei Geschäft 1 beginnt: (oder etwa nicht :-?). Ein Knoten *v* des Graphen *G=(V,E)* repräsentiert also eine geordnete *Folge* $Pv=(1, x_2, \ldots, x_k)$ und wird zu den Nachfolgern $\{P' | P' = (1, x_2, \ldots, x_k, x_{k+1}), \{x_k, x_{k+1}\} \in E)\}$ expandiert. Es fällt gleich auf, dass so verschiedene Teilwege zu verschiedenen Knoten im Suchgraphen führen. Der Zustandsraum entfaltet sich also in Form eines riesigen (Weihnachts-)baumes. Er umfasst hier die Menge aller Teilwege, und davon gibt es leider gar schröcklich viele…

Wie auch immer, wir wollen nun folgendes ausprobieren: Ein Knoten v repräsentiert eine Knoten***menge*** $M_v = \{1, x_2, \ldots, x_k\}$. Wir verzichten also auf jegliche Ordnungsinformation. Damit subsumiert eine dieser Mengen gleich eine große Anzahl der früheren Teilfolgen, folglich ist der Zustandsraum viel kleiner! Wie vorher werden die Längen der besten bisher bekannten Teilfolgen als *g(v)* gespeichert (und nun aktualisiert, falls ein besserer Teilweg von Knoten 1 über die Knoten der Menge M_v gefunden wurde). Da es mehrere Wege zum selben Knoten im Suchgraphen geben kann, arbeiten wir auf einer Art modernem „Weihnachtsgraph®en“ *G*.

Als heuristische Funktion kann im ersten Anlauf natürlich die Null-Funktion, später, wenn Ihr so wollt, auch komplexere Schätzfunktionen verwenden, bis hin zur Summe der Kosten des billigsten Spannbaumes von $G \setminus P_v$ und der Kosten der billigsten Kante von $G \setminus P_v$ zum ersten Knoten (Knoten 1). Aber das ist schon ziemlich “advanced“, ihr dürft es Euch auch wesentlich einfacher machen. Beachtet auch, dass die Funktion monoton sein sollte, sonst habt ihr, wie ihr ja inzwischen wisst, u.U. schlimme bis schlimmste Effizienz-Einbußen (die Nullfunktion ist übrigens immer monoton!).

Vergleicht die Laufzeit / Speicherbedarf des $\mathbf{A}^{\mathbf{Weihnachts\text{-}}*}$-Algorithmus’ mit Weihnachtsgraph® mit der ersten, einfachen Version, die noch „herkömmliche“ Weihnachtsbäume verwendet.

®: not (yet;) a trademark of ®üdiger