Entwurf und Analyse von Algorithmen

WS 1999/2000
Prof. Dr. K. Madlener
Übungsblatt 9

Aufgabe 9.1. Wir wollen uns mit dem Problem der Größenabschätzung von Backtracking-Bäumen auseinandersetzen.

Dazu betrachten wir eine algorithmische Methode, die schnell eine grobe Abschätzung der Anzahl $|T|$ der Knoten eines Zustandsbaumes T liefert, ohne den vollständigen Backtracking-Algorithmus laufen zu lassen.

Wir betrachten zunächst einen Spezialfall. Seien dazu n sowie $c_0, \ldots, c_{n-1}$ positive ganze Zahlen. Wir nehmen an, dass T ein Baum der Höhe n sei, für den jeder Knoten der Tiefe i genau c_i Söhne habe $(0 \leq i \leq n-1)$. Mit anderen Worten: Für $0 \leq i \leq n-1$ und jede partielle Lösung $X = (x_0, \ldots, x_{i-1})$ gebe es c_i Möglichkeiten für x_i.

a) Wieviele Knoten der Tiefe i hat ein solcher Baum T? Wieviele Knoten hat T insgesamt?

Im Allgemeinen wird T keine solche reguläre Struktur besitzen. Wir können eine Schätzung der Knotenanzahl von T erhalten, indem wir einen zufälligen Pfad durch T von der Wurzel bis zu einem Blatt verfolgen. Dies tut der Algorithmus EstimateBacktrack (dabei bezeichne $\mathcal{C}_\ell$ die Menge der möglichen Kandidaten für x_ℓ):

procedure Probe(ℓ, m, N)

Berechne $\mathcal{C}_\ell$ für den Knoten $(x_0, \ldots, x_{\ell-1})$
$c := |\mathcal{C}_\ell|$
if $c \neq 0$ **then**
  $m := m \cdot c$
  $N := N + m$
  $x_\ell :=$ ein zufälliges Element aus $\mathcal{C}_\ell$
  Probe$(\ell + 1, m, N)$

procedure EstimateBacktrack

$m := 1$
$N := 1$
Probe$(0, m, N)$ # m und N werden durch Probe verändert!
return N

Für einen Zustandsbaum T berechnet der Algorithmus den Wert $N = N(P)$, der von dem Pfad P, der durchschritten wird, abhängt. Dabei kann N kleiner oder größer als $|T|$ sein. Wir wollen zeigen, dass der Erwartungswert von N genau $|T|$ ist. Dazu definieren wir folgende Funktion auf den Knoten von T:

$$S((x_0, \ldots, x_{\ell-1})) = \begin{cases} 1 & \text{falls } \ell = 0 \\ |\mathcal{C}_{\ell-1}((x_0, \ldots, x_{\ell-2}))| \cdot S((x_0, \ldots, x_{\ell-2})) & \text{falls } \ell \geq 1. \end{cases}$$

Wenn also X die Wurzel ist, ist $S(X) = 1$, und wenn es c Möglichkeiten zur Erweiterung eines Knotens X gibt, ist $S(Y) = c \cdot S(X)$ für alle Söhne Y von X.

b) Zeigen Sie, dass $N(P) = \sum_{Y \in P} S(Y)$ ist, falls P der Pfad ist, den der Algorithmus durchschreitet.

c) Zeigen Sie nun, dass der Erwartungswert für den Wert, den der Algorithmus liefert, genau $|T|$ ist. Berechnen Sie dazu für einen beliebigen Pfad von der Wurzel zu einem Blatt im Zustandsbaum die Wahrscheinlichkeit, dass genau er durchschritten wird, und verwenden Sie dann die Gleichung aus Teil b).

Aufgabe 9.2. Betrachten Sie die TSP-Instanz, die durch die folgende Kostenmatrix gegeben ist:

$$\begin{pmatrix} \infty & 7 & 3 & 12 \\ 3 & \infty & 6 & 14 \\ 5 & 8 & \infty & 6 \\ 9 & 3 & 5 & \infty \end{pmatrix}$$

a) Geben Sie die reduzierte Kostenmatrix an.

b) Zeichnen Sie den Teil des Zustandsbaumes, der von dem in der Vorlesung behandelten Branch-And-Bound-Algorithmus erzeugt wird, wobei die statische Organisation verwendet werde. Geben Sie zu jedem Knoten seinen $\hat{c}$-Wert sowie die korrespondierende reduzierte Matrix an.

c) Wiederholen Sie Teil b) für die dynamische Organisation des Zustandsbaumes.

Aufgabe 9.3. Zeigen Sie folgende Aussage aus der Vorlesung für das 15-Puzzle: Die Zielkonfiguration ist von einer gegebenen Konfiguration des Puzzles aus genau dann erreichbar, wenn in dieser Konfiguration gilt: $\gamma + \sum_{1 \leq i \leq 16} \mathrm{LESS}(i)$ ist gerade. (Zu den Bezeichnungen siehe Vorlesung.)

Aufgabe 9.4. Wir betrachten den generischen Branch-And-Bound-Algorithmus LC1 aus der Vorlesung und Näherungen $\hat{c}$ der Kostenfunktion c mit $\hat{c}(x) \leq c(x)$ für alle Knoten x und $\hat{c}(x) = c(x)$ für alle Ergebnisknoten x. Beweisen oder widerlegen Sie:

a) Die Anzahl der erzeugten Knoten eines Zustandsbaumes kann durch die Expansion eines Knotens x mit $\hat{c}(x) > u$ (wobei u die aktuelle obere Schranke für eine minimale Lösung bezeichne) nicht verringert werden.

b) Seien u_1 und u_2 zwei obere Schranken für die Kosten einer minimalen Lösung mit $u_1 \leq u_2$. Dann werden mit u_1 als initialer Schranke die selben Knoten aktiv wie mit u_2 als initialer Schranke, und die nach $\hat{c}$ sortierte Liste der lebendigen Knoten mit u_1 als initialer Schranke ist immer Anfangsliste derjenigen mit u_2 als initialer Schranke.

c) Sind $\hat{c}_1$ und $\hat{c}_2$ Rangfunktionen mit $\hat{c}_1(x) \leq \hat{c}_2(x) \leq c(x)$ für alle Knoten x, so werden bei Verwendung von $\hat{c}_2$ nicht mehr Knoten aktiv als bei Verwendung von $\hat{c}_1$.

d) Sind $\hat{c}_1$ und $\hat{c}_2$ Rangfunktionen mit $\hat{c}_1(x) \leq \hat{c}_2(x) \leq c(x)$ für alle Knoten x, so werden bei Verwendung von $\hat{c}_2$ nicht mehr Knoten erzeugt als bei Verwendung von $\hat{c}_1$.

Informationen zur Vorlesung:
http://www-madlener.informatik.uni-kl.de/ag-madlener/lectures/eaa/ws1999-2000/eaa.html