Verzweigen und Beschränken

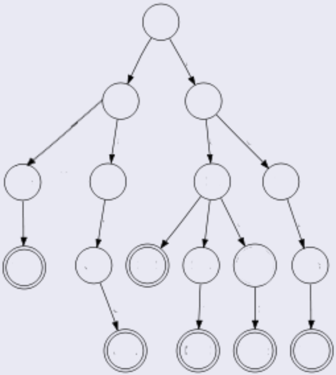
Algorithmen und Komplexität

Sebastian Rudolf, Stefan Bradl

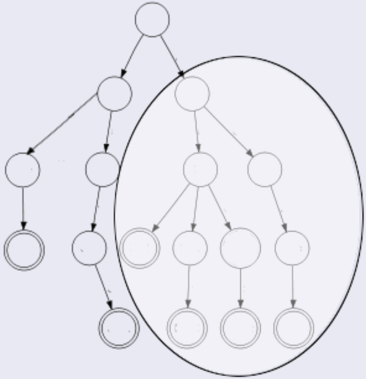
4. Mai 2009

Gliederung

Beispiel



Nichtbenötigter Teilbaum



Annahme

Angenommen jede Kante und jeder Knoten beansprucht eine Grundoperation, n ist Anzahl der Knoten $\Rightarrow$ Anzahl der Grundoperationen $= n + (n-1)$ bei vollständiger Suche

Bezogen auf Beispielbaum

$n = 17 \Rightarrow$ Anzahl der Grundoperationen $= 33$, wenn der komplette unnötige Teilbaum weggelassen werden könnte, so würden nur noch 13 Grundoperationen benötigt

Aufwandsveringerung

“Ausästen” ändert nichts am exponentiellen Aufwand, hat aber trotzdem praktischen Nutzen wie man gesehen hat

Aufwand für vollständige Suche

$T(n) = a * k^n + b$

angestrebter Aufwand

$T(n) = c * k^n + d$, mit $c < a$ und/oder $d < b$

Ansatz

Entfernen von Teilbäumen, die nicht zum gewünschten Ergebnis führen

Ausästen

Um das Ausästen durchzuführen gibt es das Hilfsmittel “Verzweigen und Beschränken” (engl: branch and bound)

Idee

- jeder Knoten bekommt eine Bewertung
- je nach Bewertung wird der Knoten mit der besten Erfolgsaussicht verfolgt, es werden also die Teilbäume entfernt, die definitiv nicht zur Lösung führen können

Wie kommt man zu diesen Bewertungen?

- Bewertungsfunktion dient zur Bewertung der Knoten
- Bewertung richtet sich danach, welche Teilbäume nicht zum Ziel führen können

Eigenschaften der Bewertungsfunktion

1. Muss gewährleisten, dass das Optimum gefunden wird
2. sollte möglichst viele Teilbäume, die nicht zur Lösung führe, entfernen

Boundoptimierung

Schlechte Bewertungsfunktion führt nicht zum Ziel, zu komplexe Bewertungsfunktion hebt Zeitersparnis durch Ausästen wieder auf

Minimierungsproblem

- Gesucht ist der Knoten im Lösungsbaum mit der kleinsten Bewertung
- Knoten mit der niedrigsten Bewertung wird verfolgt
- Schranken werden als untere Schranke bezeichnet

Maximierungsproblem

- Gesucht ist der Knoten im Lösungsbaum mit der größten Bewertung
- Knoten mit der höchsten Bewertung wird verfolgt
- Schranken werden als obere Schranke bezeichnet

priority queue

- Werteliste mit Schlüsseln, die nach Priorität geordnet sind
- zwei Operationen:
  1. Entfernen des Elements mit höchster Priorität
  2. Einfügen eines neuen Elementes

Expandierung

- beginnt bei Bewertung des Wurzelknotens, dieser wird in priority queue eingefügt
- Kindknoten wird erzeugt und dessen Bewertung ermittelt
- Kindknoten kommt ebenfalls in priority queue
- Erstes Element der priority queue wird expandiert

Anwendung

z.B. Optimale Beladung von LKW's oder Frachtschiffen

Gesucht:

- $\vec{x} = (x_1, x_2, \ldots, x_n)$ mit $x_i \in \{0, 1\}$, sodass $\sum_{i=1}^{n} x_i g_i \leq K$ und $\sum_{i=1}^{n} x_i w_i \longrightarrow max$

Spezifischer Wert

- für Bewertung wird spezifischer Wert benötigt
- berechnet sich aus $\frac{w_i}{g_i}$
- Liste der Elemente wird absteigend nach diesem Wert geordnet

Beispiel

- Kapazität: 15
- Elemente: `{(1 . 3) (3 . 55) (9 . 10) (7 . 11)}`
- Bewertungsfunktion:
  $ws \ldots$ aktuelle Wertsumme
  $gs \ldots$ aktuelle Gewichtssumme
  $sw \ldots$ spezifischer Wert des nächsten Objekts

$$Knotenbewertung = ws + (K - gs) * sw$$

Problem

- Minimierungsproblem
- Bestimmung der kürzesten Rundreise wobei jede Stadt genau ein mal besucht werden muss

Anwendung

- z.B. Optimierung des Weges bei Plottern
- Routenplanung
- Maschinensteuerung
- ...

Frage

Worin könnte der Ansatz eines naiven Algorithmus bestehen?

Naiver Algorithmus

- offensichtlichster Ansatz zur Lösung des Problems
- zuerst werden alle möglichen Rundreisen ermittelt
- Kombinatorik $\rightarrow$ Permutation $\rightarrow$ $(n-1)!$ mögliche Rundreisen
- durch STIRLINGsche Formel ergibt sich somit für Ermittlung der möglichen Rundreisen ein exponentieller Aufwand
- Aufwand für Berechnung der besten Rundreise ist hier noch nicht inbegriffen!
- Dazu kommt noch ein linearer Aufwand für die Bestimmung der kürzesten Rundreise

Was bedeutet exponentiell konkret?

Anzahl der Städte	mögliche Rundreisen $(n-1)!$
3	2
5	24
10	362.880
15	87.178.291.200
20	121.645.100.408.832.000
50	$608,281 * 10^{60}$

Mögliche Bewertungsfunktionen

Summe der Teilweglängen bis zum nächsten unbesuchten Knoten

Frage

Was ist an dieser Bewertungsfunktion zu kritisieren?

Antwort

Auch wenn der aktuelle Weg kurz ist, kann der Restweg extrem lang werden.

Mögliche Bewertungsfunktionen

Nutzung der Summe aller hinführenden und wegführenden Kanten und anschließendes durch zwei Teilen

Frage

Was ist an dieser Bewertungsfunktion zu kritisieren?

Antwort

Man bekommt nur eine Näherungslösung

Mögliche Bewertungsfunktionen

Nutzung von Entfernungsmatrizen

von\nach	1	2	3	4
1	∞	20	15	10
2	8	∞	9	8
3	6	12	∞	13
4	5	10	9	∞

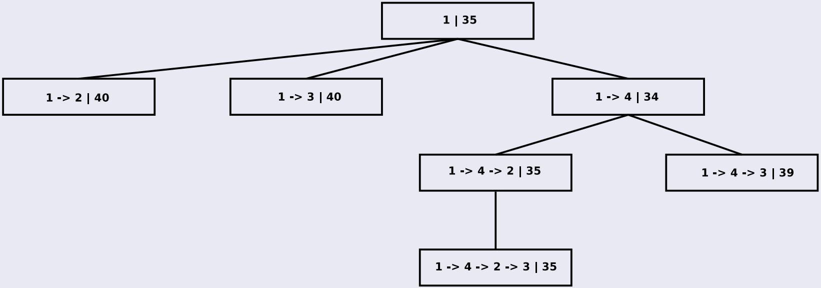
Wozu reduzierte Matrizen?

- zur Normierung
- dadurch erhält man am Ende die Länge des optimalen Weges

reduzierte Entfernungsmatrix

von\nach	1	2	3	4	
1	∞	5	4	0	10
2	0	∞	0	0	8
3	0	1	∞	7	6
4	0	0	3	∞	5
	0	5	1	0	**35**

Lösung



Übung

- priority queue
- Rucksackproblem am Beispiel selber rechnen
- Überprüfung, dass Verzweigen und Beschränken das korrekte Ergebnis bei TSP liefert
- TSP am Beispiel selber rechnen