Online

Traveling Salesman Problem

Proseminar
Online-Algorithmen
bei Prof. Dr. Rolf Klein
WS 00/01

Vortrag gehalten am
20. Dezember 2000

Tobias Honecker

Quelle: Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, Maurizio Talamo.
Competitive Algorithms for the On-line Traveling Salesman *in:* Proc. 4th Workshop Algorithms Data Struct., Springer Verlag 1995

Inhaltsverzeichnis

1. Problemvorstellung

Traveling Salesman Problem (TSP)

Das Traveling Salesman Problem (TSP) ist ein Vehicle Routing Problem. Ausgehend von einem Startpunkt (Ursprung, o oder 0 genannt) müssen verschiedene Punkte (requests) besucht werden. Danach muß zum Ursprung zurückgekehrt werden.

Online Traveling Salesman Problem (OLTSP)

Bei der Online-Variante erfährt der Handlungsreisende die requests erst unterwegs.

Anwendungsbereich

Die beiden Probleme haben mannigfaltige Bezüge in der Realität, das OLTSP ist beispielsweise im Logistikbereich ein grundlegendes Problem. So kann man sich den Handlungsreisenden beispielsweise als ein Kurier-Fahrzeug vorstellen, ebenso kann man natürlich an einen Handwerker-Service oder etwas Vergleichbares denken.

Festlegungen

Wir führen ein:

- Wir nennen den Salesman „*Server*“.
- Wir vergleichen einen „Online-Server“ mit einem „Offline-Server“.
- Beide müssen die gleichen Punkte (*requests*) besuchen.
- Der „Offline-Server“ kennt alle requests im Voraus,
  der „Online-Server“ erfährt seine Ziele teilweise erst unterwegs.
- Die requests dürfen erst nach ihrer Bekanntgabe besucht werden.
- Beide Server bewegen sich mit fester und gleicher Geschwindigkeit.

Als Vergleich für das Online-Problem benutzen wir das Offline-Problem “Vehicle Routing with release times”. Dem Offline-Server sind sämtliche requests bereits beim Start bekannt, er darf sie aber erst nach der Bekanntgabe, der so genannten „release time“ besuchen. Unabhängig davon, wie gut sich das Offline-Problem in der Realität lösen läßt, gehen wir davon aus, daß der Offline-Server optimal ist.

Es ergibt sich folgende Aufgabe für beide Server:

- im Ursprung starten und enden
- alle Orte besuchen, jedoch nicht vor der release time
- möglichst kurze Route
- „Spielfeld“ ist komplett bekannt

2. Untere Schranke für c-kompetitive Algorithmen

Zunächst wollen wir herausfinden, wie gut ein Online-Algorithmus im Vergleich zum optimalen Offline-Algorithmus überhaupt sein kann.
Um einen sinnvollen Vergleich überhaupt zu ermöglichen, muß der angenommene Online-Algorithmus c-kompetitiv sein.
Wir suchen also eine Schranke für c, so daß jeder OLTSP-Algorithmus mindestens c-kompetitiv oder schlechter ist.

Fragestellung:

Wie gut kann ein c-kompetitiver OLTSP-Algorithmus sein?

Annahme:

Wenn ein Online-Algorithmus c-kompetitiv ist, dann $c \geq \frac{9+\sqrt{17}}{8}$

Beweisführung:

Wir führen diesen Beweis in einem besonderen metric space, nämlich auf der real line. (Anschaulich: wie die „Zahlengerade“)

Beide Server starten bei t=0 in 0 und müssen hier ihren Rundgang wieder beenden.

Die requests erhalten die beiden Server wie in der Tabelle rechts.

Zielvorgaben:

Zeit t	request
0	
1	-1 ; +1
2	
3	+1
4	

Der Offline-Server würde folgende optimale Lösung finden:

Zeit t	request	Standort Offline-S.
0		0
1	-1 ; +1	-1
2		0
3	+1	+1
4		0

Den Zeitbedarf für den optimalen Offline-Server vom Start bis zum Ziel bezeichnen wir mit Z*.
Offensichtlicher Zeitbedarf für diese Folge von requests: Z*=4

Einführung Bezeichnungen:

t	Zeit
Z*	Laufzeit Offline-Server
Z^{OL}	Laufzeit Online-Server
p*(t)	Position Offline-Server zum Zeitpunkt t
$p^{OL}(t)$	Position Online-Server zum Zeitpunkt t

Ein Algorithmus heißt c-kompetitiv, wenn gilt: $Z^{OL} \leq c\, Z^*$

Annahme:

Wenn ein Online-TSP-Algorithmus c-kompetitiv ist, dann gilt: $c \geq \frac{9+\sqrt{17}}{8}$

Um diese Kompetitivitäts-Schranke zu beweisen, betrachten wir die Position des Online-Servers zu verschiedenen Zeitpunkten.

Welche Position $p^{OL}(1)$ hat der Online-Server bei t=1?

Behauptung: $p^{OL}(1) \in [\,-(2c-3)\,,\,(2c-3)\,]$

Beweis:

[Anmerkung: Im folgenden wird nur bewiesen, daß sich der Online-Server nicht in positiver Richtung aus dem Intervall heraus beweg t haben kann. Der Beweis in negativer Richtung wird hier nicht gezeigt, er funktioniert aber analog zu dem hier gezeigten.]

Annahme: $p^{OL}(1) > (2c-3)$ im Zeitpunkt t=1

Jetzt geben wir dem Online-Server als einzigen request –1 im Zeitpunkt t=1.

Dann: $Z^{OL} > 1 + (2c-3) + 2 = 2c$

Erläuterung der einzelnen Teile der Ungleichung: Der Online-Server braucht zur Beendigung des Parcours mindestens die Zeit, die bereits verstrichen ist (1), da er sich aus dem Intervall heraus nach rechts bewegt haben soll, muß er, um zu –1 zu kommen, mindestens (2c-3) bis zum Ursprung zurück laufen. Dann muß er noch vom Ursprung aus nach –1 und zurück laufen (also insgesamt 2).

Der optimale Offline-Server benötigt nur: Z* = 2

Durch Einsetzen erhalten wir also: $Z^{OL} > c\, Z^*$ ↯ Widerspruch zur Kompetitivität ↯

Also kann sich der Online-Server bis zum Zeitpunkt t=1 nicht aus dem Intervall heraus bewegt haben:

$\Rightarrow p^{OL}(1) \in [\,-(2c-3)\,,\,(2c-3)\,]$

Als nächstes betrachten wir, wo sich der Online-Server zum Zeitpunkt t=3 befindet, wenn er den Parcours durchläuft, wie er in der Tabelle vorgegeben ist.

Welche Position $p^{OL}(3)$ hat der Online-Server bei t=3?

Behauptung: $p^{OL}(3) \notin [\,-(7-4c)\,,\,(7-4c)\,]$

Beweis:

[Anmerkung: Es wird wieder nur eine Richtung des Beweises gezeigt. Die andere erfolgt wiederum analog der gezeigten. Im folgenden werde ich ohne einen weiteren Hinweis darauf verzichten, alle symmetrischen Beweisrichtungen zu zeigen.]

Zeit t	request
0	
1	-1 ; +1
2	
3	+1
4	

Annahme: $p^{OL}(3) \in [\,-(7-4c)\,,\,(7-4c)\,]$

Wir gehen davon aus, daß der Online-Server den bei t=1 gegebenen request in +1 erledigt hat, den in –1 aber noch nicht oder genau erst bei t=3. Mit dem bei t=3 hinzukommenden request in +1 haben wir also:

noch nicht erledigte requests: -1 und +1

Also: $Z^{OL} > 3 + 1 - (7-4c) + 3 = 4c$

Erläuterung: Der Online-Server benötigt mindestens die bereits verstrichene Zeit (3), von dem dann aktuellen Standpunkt aus müßte er 1-(7-4c) bis zu dem einen Extrem (nach der Annahme ist dies –1) laufen, wenn er sich innerhalb des in der Behauptung genannten Intervalls befinden würde. Von –1 aus wiederum müßte er nach +1 und zurück zum Urpsprung laufen, was wiederum 3 Zeiteinheiten kostet.

Der optimale Offline-Server benötigt nur: $Z^* = 4$

Durch Einsetzen erhalten wir also: $Z^{OL} > c\,Z^*$ ↯ Widerspruch zur Kompetitivität ↯

Also muß sich der Online-Server bei t=3 außerhalb dieses Intervalls befinden.

$\Rightarrow p^{OL}(3) \notin [\,-(7-4c)\,,\,(7-4c)\,]$

Es muß bei t=3 aber immer noch gelten: $p^{OL}(3) \in [\,-(2c-3)\,,\,(2c-3)\,]$

Begründung:
Der Online-Server hat sich nach t=1 zum request +1 bewegt und von da aus in Richtung -1. Daher muß er immer noch im Intervall $[\,-(2c-3)\,,\,(2c-3)\,]$ sein.

Also:

$p^{OL}(3) \in [\,-(2c-3)\,,\,(2c-3)\,]$
$p^{OL}(3) \notin [\,-(7-4c)\,,\,(7-4c)\,]$

Bemerkung

Wenn $c \geq \frac{9+\sqrt{17}}{8} \approx 1{,}64$ dann folgt die Behauptung.

Wenn $c < \frac{9+\sqrt{17}}{8} \approx 1{,}64$ dann wäre das Intervall [-(2c-3) , (2c-3)]
im Intervall [-(7-4c) , (7-4c)] enthalten. Widerspruch!

Aber: Wie kommt man auf diese Schranke?

Feststellung:
Der Online-Server befindet sich bei t=3 im Intervall $(7-4c) \leq p(3) \leq 1$

Begründung:
Er kann sich, wenn er +1 besucht hat, nicht links von (7-4c) befinden.
Da er sich von +1 in Richtung –1 bewegt, muß er links von +1 sein.

[Der andere Fall ist symmetrisch!]

Jetzt muß der Online-Server noch –1 besuchen.
Um –1 besuchen zu können, muß 0 passiert werden!
Annahme: Der Online-Server passiert 0 bei t=(3+q)

Um c-kompetitiv zu sein, muß dies spätestens bei (4c-2) passieren
Erläuterung: Z*=4 ist die Laufzeit des Offline-Algorithmus, um c-kompetitiv zu sein, muß also $Z^{OL} \leq 4c$ gelten. Da der Online-Server zum betrachteten Zeitpunkt noch vom Ursprung aus zu –1 und zurück gehen muß, muß er spätestens bei 4c-2 im Ursprung sein.

Also:

$$3+q \leq 4c-2$$
$$\Leftrightarrow \quad q \leq 4c-5$$

Zum Zeitpunkt t=(3+q) befindet sich der Online-Server also in 0.

Der Offline-Server befand sich bei t=3 in +1. Er kann sich danach weiter vom Ursprung entfernt haben, er kann sich also zum Zeitpunkt t=(3+q) im Punkt (1+q) befinden.

Jetzt geben wir den beiden Servern einen neuen request bei 1+q (aktueller Standort des Offline-Servers).

Zeit t	request
0	
1	-1 ; +1
2	
3	+1

Der Offline-Server kann also von dort aus direkt zurück nach 0 gehen:

$\Rightarrow Z^* = (3+q) + (1+q) = 4+2q$

Der Online-Server muß noch erledigen:

zuerst zu –1, dann zurück zum Ursprung 0, dann nach (1+q), dann zurück zu 0
$\Rightarrow Z^{OL} = (3+q) + 2 + 2(1+q) = 7+3q$

Erläuterung : Die Zeit (3+q) ist bereits verstrichen, vom Ursprung aus zu –1 und zurück kostet 2, danach vom Ursprung aus zu (1+q) und zurück kostet 2(1+q).

nach Vorauss. : $Z^{OL} \leq c\, Z^*$

$\Leftrightarrow \quad c \geq Z^{OL} / Z^*$

$\Leftrightarrow \quad c \geq (7+3q) / (4+2q)$

Diese Funktion von q ist monoton steigend. Also können wir für q einsetzen: $q \leq 4c-5$:

$$c \geq (7+3(4c-5)) / (4+2(4c-5))$$

$$\Leftrightarrow c \geq \frac{9+\sqrt{17}}{8} \approx 1{,}64$$

q.e.d.

Das heißt:

Jeder beliebige Online-TSP Algorithmus, der c-kompetitiv ist, ist mindestens 1,64-kompetitiv oder schlechter!

3. Vorbemerkung Algorithmus „Plan at Home“ (PAH)

Kein OLTSP-Algorithmus kann also besser als 1,64-kompetitiv sein.

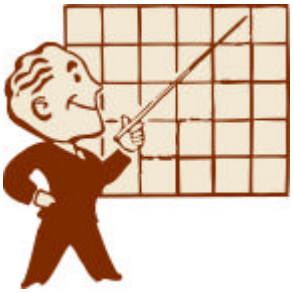
Welche guten Algorithmen gibt es also?

Bereits naive Algorithmen sind 2,5-kompetitiv.

Beispiel:
Nimm immer den kürzesten Weg vom aktuellen Standpunkt aus.
= 2,5-kompetitiv

Gibt es einen einfachen 2-kompetitiven Algorithmus?

Plan at Home (PAH)



Wir betrachten den folgenden Algorithmus im *metric space*.

Definitionen:

Z^{PAH}	Laufzeit PAH-Algorithmus
$p^{PAH}(t)$	Position des PAH-Servers im Zeitpunkt t
d(x,y)	Abstand zweier Punkte
T	Pfad
T*	optimaler Pfad von 0 aus für alle bisher bekannten requests
$\|T\|$	Länge des Pfades T

Arbeitsweise des PAH

1. Wenn der Server im Ursprung ist, folgt er der optimalen Route.
   (genau wie der Offline-Server)
2. Der PAH-Server befindet sich im Punkt p und erhält einen neuen request im Punkt x:
   a) Wenn d(x,0) > d(p,0): Server geht direkt zurück zu 0 und kommt zu Fall 1.
   b) Wenn d(x,0) ≤ d(p,0): Server ignoriert den neuen request komplett, bis er wieder in 0, also in Fall 1, ist.

Behauptung:

PAH ist 2-kompetitiv.

Beweis:

Wenn PAH in allen 3 Fällen 2-kompetitiv ist, so ist er es auch insgesamt.

4. Beweis 2-Kompetitivität PAH

Fall 1

Zum Zeitpunkt t wird ein beliebiger neuer request vorgegeben.
Der Server befindet sich in 0.

Offensichtlich gilt: $Z^* \geq t$
und: $Z^* \geq |T^*|$

Weiter gilt offensichtlich: $Z^{PAH} \leq t + |T^*| \leq 2\, Z^*$

⇒ Fall 1 ist 2-kompetitiv

Fall 2a $d(x,0) > d(p,0)$

Nach t geht der Server zurück zum Ursprung und von da aus eine optimale Tour:
$Z^{PAH} < t + d(x,0) + |T^*|$

Es gilt: $Z^* \geq |T^*|$
und: $Z^* \geq t + d(x,0)$

⇒ $Z^{PAH} < 2\, Z^*$
⇒ Fall 2a ist ebenfalls 2-kompetitiv

Fall 2b $d(x,0) \leq d(p,0)$

1. Möglichkeit

Der PAH-Server ist bereits auf dem Weg zum Ursprung wegen eines vorherigen Auftretens des Falles 2a.

2. Möglichkeit

PAH geht eine Route R der beim letzten Besuch in 0 bekannten requests.

S	Menge der ignorierten requests,
s	der erste request aus S, der vom Offline-Server besucht wird,
t_s	der Zeitpunkt, zu dem der request s gestellt wird,
P_s^*	der kürzeste Pfad von s durch alle Punkte in S zu 0.

Offline-Server: $Z^* \geq t_s + |P_s^*|$

PAH im Zeitpunkt t_s: muß höchstens noch $|R| - d(0,s)$ von R gehen
$\Rightarrow$ Ankunft des PAH in 0 spätestens: $t_s + |R| - d(0,s)$
Optimale Tour von 0 durch alle Punkte in S zu 0: T_s^*

$\Rightarrow Z^{PAH} \leq t_s + |R| - d(0,s) + |T_s^*|$

$|T_s^*| \leq d(0,s) + |P_s^*|$
$\Rightarrow Z^{PAH} \leq t_s + |R| - d(0,s) + d(0,s) + |P_s^*| = t_s + |R| + |P_s^*|$

$Z^* \geq |R|$ und $Z^* \geq t_s + |P_s^*|$
$\Rightarrow Z^{PAH} \leq 2\, Z^*$

$\Rightarrow$ Fall 2b ist 2-kompetitiv!

$\Rightarrow$ PAH ist 2-kompetitiv!

q.e.d.

5. Zusammenfassung

Im Kapitel 2 haben wir gesehen, daß jeder Online-TSP-Algorithmus bestenfalls 1,64-kompetitiv sein kann.

Im Kapitel 3 haben wir dann eine relativ einfache Lösungsstrategie kennengelernt. Im Anschluß daran haben wir in Kapitel 4 gezeigt, daß diese Strategie 2-kompetitiv ist.
Es handelt sich also nicht nur um eine verhältnismäßig einfache Strategie, sie erzielt sogar ein gutes Ergebnis.