Symmetrisk traveling salesman problem

Dat2A godkendelsesopgave 2

Jens Kristian Jensen, David Pisinger og Martin Zachariasen

13. april 2003

1 Formalia

Dette er den anden af to godkendelsesopgaver på kurset Dat2A, efterår 2002 og forår 2003. Opgaven skal løses i grupper på 2–3 personer (enkeltmandsgrupper kræver begrundelse og skal godkendes af instruktor).

Opgaven bliver stillet mandag d. 14/4 2003 kl. 11.30 og skal afleveres i førstedelsadministrationen i 2 eksemplarer inden **onsdag d. 30/4 2003 kl. 11.30**. Bedømmelserne foreligger i førstedelsadministrationen senest torsdag d. 15/5 2003.

Opgaven vil blive bedømt enten *godkendt*, *ikke godkendt* eller *dumpet*. Bedømmelsen *ikke godkendt* giver mulighed for genaflevering senest d. 22/5 2003, mens bedømmelsen *dumpet* ikke giver mulighed for genaflevering. Opgavebesvarelser hvori der ikke er gjort et *væsentligt forsøg* på at besvare alle spørgsmål bliver betragtet som dumpet.

Rapporten skal være på højst 12 sider, excl. programudskrifter og udskrifter fra kørsler (der også vedlægges). Desuden skal alle kildetekstfiler — herunder en README-fil der beskriver hvordan filerne oversættes til et færdigt program — gøres tilgængelige på en 1. delskonto (`di`-konto) ejet af en af deltagerne; filerne lægges i et katalog med navnet `G22A` og må ikke rettes efter rapportens aflevering.

2 Formål

Formålet med opgaven er at

- få praktisk erfaring med implementation af grafalgoritmer
- forstå betydningen af kvaliteten af grænseværdier i branch-and-bound algoritmer
- få et indblik i teknikker til løsning af NP-hårde problemer

Opgavens formål er at analysere og implementere en række grænseværdier for det symmetriske traveling salesman problem (se afsnit 3), samt eksperimentelt at sammenligne deres fordele og ulemper. Målet skulle være optimal løsning af problemer med op til 25–30 knuder (byer).

Der skal konstrueres en generel algoritme som kan løse problemet. Algoritmen skal implementeres i C++ og gerne ved anvendelse af LEDA. Besvarelsen skal bestå af et tekstafsnit, hvor nedenstående opgaver besvares, samt udskrift af det **kommenterede** program. Ved bedømmelse af opgaven vil der blive lagt vægt på at alle spørgsmål (hver for sig) er besvaret tilfredsstillende.

3 Symmetrisk traveling salesman problemet

Den handelsrejsendes problem, eller traveling salesman problemet (TSP), har altid fascineret forskere: Problemet er simpelt at formulere og alligevel meget svært at løse i praksis. Der er skrevet utallige forskningsartikler om emnet, og flere lærebøger betragter TSP som hovedemne. TSP er ikke blot fascinerende i matematisk betydning, men det er et særdeles vigtigt problem i praksis. En lang række problemer i transportsektoren, produktionsplanlægning og endda arkæologi kan formuleres ved brug af TSP. Hertil kommer at teorien om NP-fuldstændige problemer udsprang af forskeres manglende succes med løsning af TSP.

Mange års forskning har flyttet grænserne for, hvor store TSP problemer vi kan løse. Selv om der altid vil eksistere probleminstanser, som kræver eksponentiel løsningstid, udnytter moderne algoritmer strukturen i problemet til konstruktion af stramme grænseværdier. Problemer med flere hundrede knuder (byer) kan nu løses rimelig let.

Formelt kan det symmetriske TSP defineres som følgende optimeringsproblem (se også afsnit 34.5.4 i CLRS [1]): Lad $G = (V, E)$ være en vægtet (komplet) graf med $n = |V|$ knuder, hvor $c(i, j)$ for $(i, j) \in E$ angiver afstanden mellem

knuderne i og j. Afstanden antages at være symmetrisk, således at $c(i,j) = c(j,i)$ for alle par af knuder i og j. Problemet er da at finde en Hamilton kreds i grafen som har en minimal total længde med hensyn til c.

I det følgende antages at alle afstande $c(i,j)$ er heltal. Herunder er et eksempel som angiver afstandene mellem 8 byer på Bornholm.

i \ j	1	2	3	4	5	6	7	8
1	0	11	24	25	30	29	15	15
2	11	0	13	20	32	37	17	17
3	24	13	0	16	30	39	29	22
4	25	20	16	0	15	23	18	12
5	30	32	30	15	0	9	23	15
6	29	37	39	23	9	0	14	21
7	15	17	29	18	23	14	0	7
8	15	17	22	12	15	21	7	0

Den optimale løsning at besøge knuderne (byerne) i rækkefølgen: $1, 8, 7, 6, 5, 4, 3, 2, 1$, hvilket giver en samlet længde af Hamilton kredsen på 100.

4 Grådig løsning

Det grådige princip er en velkendt metode til konstruktion af heuristiske løsninger. For TSP fungerer den grådige algoritme som følger: Sorter alle kanter efter vægt. Tilføj en kant til turen, hvis den ikke danner en knude med grad større end 2 eller danner en deltur med færre end n knuder.

Opgave 1 Vis at den grådige algoritme kan implementeres til at køre i $O(n^2 \log n)$ tid, og implementer denne algoritme. ■

5 Nedre grænseværdier

Vi vil i denne opgave betragte to forskellige nedre grænseværdier for TSP:

- *1-Tree*. Et mindste udspændende træ (MST) konstrueres på grafen, hvor knude 1 midlertidigt er fjernet. Herefter tilføjes de to billigste kanter incidente med knude 1 til det udspændende træ. I ovenstående eksempel finder vi et MST blandt knuderne $2, \ldots, n$ som kanterne $(2,3)$, $(3,4)$, $(4,8)$, $(5,6)$, $(6,7)$ og $(7,8)$. De to billigste kanter fra knude 1 er $(1,2)$ og $(1,7)$. Samlet fås en nedre grænseværdi på 97.

- *Itereret 1-Tree*. Princippet er her at omformulere problemet, så en strammere 1-tree grænseværdi opnås. Kursusbog 1, del 1, side 13 nederst beskriver dette princip i detaljer. Lad d_i være antallet af valgte kanter incidente med knude i. Vi ønsker at "straffe" knuder med grad større end 2, mens knuder med grad mindre end 2 "belønnes". Derfor modificeres afstandsmatricen c på følgende vis:

  $$c'(i,j) = c(i,j) + (d_i - 2) + (d_j - 2)$$

  Processen kan gentages et antal gange. I praksis er 10–100 iterationer passende. Eksempelvis kan man benytte 100 iterationer i rod-knuden af branch-and-bound søgetræet, og gradvist mindske antallet af iterationer i forhold til knudens dybde i søgetræet. Knuder, som har en dybde over 10 i søgetræet, kan passende nøjes med 10 iterationer. Bemærk, at en knude i søgetræet ikke bør starte med at iterere afstandsmatricen c fra grunden af, men i stedet bør tage udgangspunkt i den bedste modificerede afstandsmatrix c' fra forældre-knuden.

  I ovenstående eksempel får vi følgende modificerede afstandsmatrice c':

i \ j	1	2	3	4	5	6	7	8
1	0	11	24	25	29	29	16	15
2	11	0	13	20	31	37	18	17
3	24	13	0	16	29	39	30	22
4	25	20	16	0	14	23	19	12
5	29	31	29	14	0	8	23	14
6	29	37	39	23	8	0	15	21
7	16	18	30	19	23	15	0	8
8	15	17	22	12	14	21	8	0

  Ved løsning af 1-tree for det modificerede problem findes kanterne $(2,3)$, $(3,4)$, $(4,8)$, $(5,6)$, $(5,8)$, $(7,8)$, $(1,2)$, og $(1,8)$, hvor de første seks kanter angiver MSTet blandt knuderne $2,\ldots,n$. Samlet fås en nedre grænseværdi på 97. De efterfølgende iterationer giver grænseværdierne $98, 99, 99, 100$.

Opgave 2 Bevis at grænseværdierne er lovlige nedre grænser. Vurder deres beregningskompleksitet ved brug af store-O notation. Det antages at Kruskal's algoritme bruges til konstruktion af MSTer. ■

Opgave 3 Implementer algoritmer til udregning af grænseværdierne, og eksperimenter med antallet af iterationer i den *itererede 1-tree* grænseværdi. ■

6 Branch-and-bound algoritme

Ved konstruktion af en branch-and-bound algoritme skal man overveje om der skal bruges dybde-først eller bedste-først søgning, samt hvilken forgreningsstrategi, der skal benyttes. I det følgende angiver z^* længden af den hidtil bedste løsning.

Dybde-først søgning er nemmest at implementere, idet algoritmen da blot bliver en rekursiv procedure som skitseret nedenfor:

```
TSP_BRANCH(n, c)
find en nedre grænseværdi ℓ for problemet.
if ℓ >= z* then return
if løsningen i grænseværdiberegningen er en Hamilton kreds then
   gem løsningen, opdater z*.
else
   vælg en knude som spolerer Hamilton kredsen
      for alle valgte kanter incidente med knuden
         forbyd kanten, kald TSP_BRANCH(n, c)
```

Håndteringen af "levende knuder" i branch-and-bound træet foretages automatisk af rekursionen, idet tidligere problemer gemmes på stakken. I ovenstående skitse er forgreningstrategien fra kursusbog 1, del 1, side 18, blevet benyttet: En knude udvælges, som har flere end to kanter, og branch-and-bound algoritmen forbyder på skift hver af kanterne ud fra knuden.

Opgave 4 Implementer branch-and-bound algoritmen. Benyt gerne det udleverede rammeprogram (se kursets hjemmeside). Beskriv fordele og ulemper ved den valgte forgreningsstrategi og dybde-først søgningen. Hvis tiden tillader det, kan der eksperimenteres med andre søge-strategier og forgrenings-strategier, omend dette ikke er et krav. (Vær opmærksom på at *hele* løsningsrummet skal dækkes; knude 1 indgår ikke nødvendigvis i en optimal løsning med de to mindste incidente kanter.) ■

7 Eksperimenter

Afprøv den konstruerede algoritme med de to grænseværdier på følgende probleminstanser, som findes på kursets hjemmeside.

n	Instans	Beskrivelse
8	bornholm	afstande mellem otte byer på Bornholm
10	rand10	tilfældigt genererede kantvægte
15	rand15	tilfældigt genererede kantvægte
20	rand20	tilfældigt genererede kantvægte
25	rand25	tilfældigt genererede kantvægte
30	rand30	tilfældigt genererede kantvægte
17	gr17	17 byer i Tyskland
21	gr21	21 byer i Tyskland
24	gr24	24 byer i Tyskland

Opgave 5 Rapporter: øvre og nedre grænseværdier i rodknuden, antal knuder i branch-and-bound træet, beregningstid pr. knude i branch-and-bound træet, samt total beregningstid. Kommenter resultaterne. ■

Opgave 6 Forsøg at konstruere probleminstanser, hvor ovenstående grænseværdier er svage, f.eks. når MSTet er stjerneformet og dermed langt fra at være en Hamilton kreds. ■

8 Forbedret Kruskal

For de to grænseværdier baseret på 1-tree beregninger viser det sig, at langt den største del af CPU-tiden bruges til sortering af kanter i Kruskal's algoritme. For at reducere denne beregningstid, foreslås det at benytte den forbedrede MST-algoritme fra G1-opgaven.

Opgave 7 Kør tests med alle probleminstanser for den forbedrede MST-algoritme. Kommenter resultaterne. Kan det gøres endnu bedre? (Bemærk at kanternes rækkefølge i sorteringen ofte ikke ændrer sig drastisk.) ■

9 Gode råd

Tidspunkter for opgavehjælp er annonceret på kursets hjemmeside. Desuden vil svar på spørgsmål af generel interesse løbende blive lagt ud på kursets hjemmeside.

Et rammeprogram til løsning af opgaven samt ovennævnte probleminstanser findes på kursets hjemmeside. Hvis det tager mere end et minut at løse en probleminstans, så rapporter at algoritmen ikke kunne løse dette problem. Specielt vil den første grænseværdi baseret på 1-tree beregning næppe kunne løse instanser med mere end 10 knuder.

Litteratur

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein: Introduction to Algorithms (2nd ed.), MIT Press, 2001.

Noter

Nogle af de udleverede probleminstanser stammer fra TSPLIB, som findes på internettet på adressen

`www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

hvor I er velkomne til at kigge efter flere/andre instanser.

Som pauselæsning har vi fundet følgende artikel fra *New York Times*, 27. november 1979, omhandlende betydningen af en polynomiel algoritme til lineær programmering for løsning af bl.a. traveling salesman problemet.

An Approach to Difficult Problems

Mathematicians disagree as to the ultimate practical value of Leonid Khachiyan's new technique, but concur that in any case it is an important theoretical accomplishment.

Mr. Khachiyan's method is believed to offer an approach for the linear programming of computers to solve so-called "travelling salesman" problems. Such problems are among the most intractable in mathematics. They involve, for instance, finding the shortest route by which a salesman could visit a number of cities without his path touching the same city twice.

Each time a new city is added to the route, the problem becomes very much more complex. Very large numbers of variables must be calculated from large numbers of equations using a system of linear programming. At a certain point, the complexity becomes so great that a computer would require billions of years to find a solution.

In the past, "traveling salesman" problems, including the efficient scheduling of airline crews or hospital nursing staffs, have been solved on computers using the "simplex method" invented by George B. Dantzig of Stanford University.

As a rule, the simplex method works well, but it offers no guarantee that after a certain number of computer steps it will always find an answer. Mr. Khachiyan's approach offers a way of telling right from the start whether or not a problem will be soluble in a given number of steps.

Two mathematicians conducting research at Stanford already have applied the Khachiyan method to develop a program for a pocket calculator, which has solved problems that would not have been possible with a pocket calculator using the simplex method.

Mathematically, the Khachiyan approach uses equations to create imaginary ellipsoids that encapsulate the answer, unlike the simplex method, in which the answer is represented by the intersections of the sides of polyhedrons. As the ellipsoids are made smaller and smaller, the answer is known with greater precision.

MALCOLM W. BROWNE