

DATV: Introduktion til optimering og operationsanalyse

Asymmetric Traveling Salesman Problem

David Pisinger, Efterår 2004

Dette er den anden obligatoriske projektopgave på kurset "DATV: Introduktion til optimering og operationsanalyse". Opgaven skal afleveres senest 7. december 2004 kl. 12.00 i DIKU's 1. delsadministration. Besvarelsen skal udarbejdes i grupper på en til tre deltagere. Læs venligst hele opgaveformuleringen igennem inden du går igang.

Indledning

Et IP-problem kan ofte formuleres på mange måder. Selv om de matematisk set er ækvivalente, kan en "stærk" formulering være at foretrække, idet den er tættere på det konvekse hylster indeholdende alle IP-løsninger til problemet [8].

Traveling Salesman Problemet (TSP) er et klassisk, svært optimeringsproblem som vi kun kan løse takket være stærke formuleringer af problemet [2, 3, 4, 5]. Opgavens formål er at eksperimentere med forskellige formuleringer af TSP samt at konstruere en simpel branch-and-cut algoritme.

Traveling Salesman Problemet

Lad der være givet en komplet graf $G = (V, E)$ hvor hver kant $(i, j) \in E$ har en tilhørende omkostning c_{ij} . Vi antager at problemet ikke nødvendigvis er symmetrisk, dvs. der kan være situationer hvor $c_{ij} \neq c_{ji}$. Traveling Salesman Problemet har til opgave at finde den korteste Hamilton kreds i grafen, dvs. en kreds som besøger alle knuder netop een gang, og som minimerer den tilhørende omkostning af kanterne.

Lad n betegne antallet af knuder i V . For at formulere problemet som et IP problem, kan man indføre beslutningsvariablene

$$x_{ij} = \begin{cases} 1 & \text{hvis kant } (i, j) \text{ indgår i kredsen} \\ 0 & \text{ellers} \end{cases}$$

for $i, j = 1, \dots, n$. For at gøre skrivemåden nemmere vil vi i det følgende antage at $x_{ii} = 0$ for $i = 1, \dots, n$.

Idet man skal ankomme til hver knude netop een gang, og skal forlade hver knude een gang, vil en naiv formulering af TSP være

$$\begin{aligned}
 & \text{minimize} && \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \\
 & \text{subject to} && \sum_{i=1}^n x_{ij} = 1, && j = 1, \dots, n \\
 & && \sum_{j=1}^n x_{ij} = 1, && i = 1, \dots, n \\
 & && x_{ij} \in \{0, 1\}, && i, j = 1, \dots, n
 \end{aligned} \tag{1}$$

Ovenstående begrænsninger kaldes “assignment” begrænsninger.

Opgave 1 Vis at en LP-optimal løsning til ovenstående problem automatisk er heltallig. ■

Desværre er formuleringen (1) ikke tilstrækkelig til at løse TSP, idet en optimal løsning til (1) kan indeholde delture. For at forhindre disse foreslog Dantzig, Fulkerson og Johnson [1] at man tilføjer følgende delturbegrænsninger

$$\sum_{i \in S} \sum_{j \in V \setminus S} (x_{ij} + x_{ji}) \geq 2 \tag{2}$$

for alle $S \subset V$ hvor $2 \leq |S| \leq n/2$. Disse begrænsninger kaldes ofte DFJ-begrænsninger.

Opgave 2 Bevis at ulighederne (2) eliminerer alle delture (også delture der er længere end $n/2$), og angiv hvor mange uligheder der vil være. ■

Miller, Tucker og Zemlin [6] foreslog en formulering som forhindrer delture, men som kun er polynomielt stor. Ideen er at indføre nogle nye variable u_i for hver knude, som angiver rækkefølgen af knuden i turen. Knude 1 har altid $u_1 = 1$ og hvis $u_i = k$ betyder det at knude i er den k 'te besøgte knude på turen. Uden tab af generalitet kan vi sætte $u_1 = 1$, og dermed må $2 \leq u_i \leq n$ for alle øvrige knuder $i \in V \setminus \{1\}$. Endvidere skal vi kræve at

$$\text{hvis kant } (i, j) \text{ benyttes i turen, så skal } u_j \geq u_i + 1 \tag{3}$$

for alle $i = 1, \dots, n$ og $j = 2, \dots, n$ hvor $i \neq j$. Disse begrænsninger kaldes MTZ-begrænsninger.

Opgave 3 Formuler MTZ begrænsningerne som IP-model. Angiv hvor mange uligheder der vil være. Tilføj en rutine som genererer ulighederne i programmet. ■

Opgave 4 Løs problemet burma14 med begge formuleringer (DFJ og MTZ), ved brug af rammeprogrammet beskrevet sidst i opgaven. Angiv modellens størrelse (i bytes), samt løsestid for CPLEX. ■

Trods MTZ formuleringens polynomielle størrelse, er der i litteraturen ikke rapporteret anvendelser af denne model som kunne løse TSP problemer med mere end 50 knuder. Dette skyldes at MTZ formuleringen er svagere end DFJ formuleringen som vi vil anskueliggøre i næste opgave.

Opgave 5 Lad C være en kreds i grafen, og $S_c \subset V$ mængden af knuder der indgår i kredsen. Vis at begrænsningerne (2) medfører at

$$\sum_{i \in S_c} \sum_{j \in S_c} x_{ij} \leq |S_c| - 1 \quad (4)$$

Summer begrænsningerne (3) formuleret som IP-model over alle kanter $(i, j) \in C$. Vis at man dermed får uligheden

$$\sum_{(i,j) \in C} x_{ij} \leq |S_c| - \frac{|S_c|}{n-1} \quad (5)$$

Hvilken af ulighederne er stærkest, og hvilken af formuleringerne ville man derfor foretrække? ■

Opgave 6 Find den nedre grænseværdi for alle instanser af typen $ftvXX$ ved brug af:

- “assignment”-begrænsningerne alene
- “assignment”-begrænsningerne og MTZ-begrænsningerne

Begge relaxeringer løses til LP-optimalitet. Rapportør køretid og nedre grænseværdi for hver instans og for hver af de relaxeringer. ■

Vi vil nu udvikle en simpel “cutting plane” algoritme til at finde en nedre grænseværdi til TSP.

```

1 Start med en simpel formulering, der kun indeholder “assignment”-begrænsningerne
2 repeat
3   Løs modellen til LP-optimalitet med CPLEX
4   Separer den mest overtrådte DFJ-begrænsning
5   Såfremt ingen DFJ-begrænsninger er overtrådt, break
6   Tilføj den separerede ulighed til modellen
7 forever
8 Udskriv den nedre grænseværdi

```

Opgave 7 Beskriv og implementer en polynomiell algoritme, som i linie 4 kan separere den mest overtrådte DFJ-begrænsning (2). Angiv algoritmens køretid i store-O notation. ■

Opgave 8 For alle instanser af typen $ftvXX$ find en nedre grænseværdi ved at bruge ovenstående cutting plane algoritme. Rapportør køretid og nedre grænseværdi for hver instans. ■

For at finde en IP-optimal løsning til TSP problemet må ovenstående cutting plane algoritme kombineres med en branch-and-bound algoritme. Den resulterende branch-and-cut algoritme kan i princippet konstrueres på to måder:

- 1 Man separerer samtlige overtrådte DFJ-begrænsninger inden branch-and-bound algoritmen forgrener på en variabel.
- 2 Branch-and-bound algoritmen løser problemet til IP-optimalitet inden næste DFJ-begrænsning separeres.

Opgave 9 Løs så store problemer af typen $ftvXX$ som muligt med begge varianter af branch-and-cut algoritmen. Rapportér køretid, antal cuts genereret, og antal branch-and-bound knuder besøgt. ■

Opgave 10 Diskuter forskellen på de to branch-and-cut algoritmer, og foreslå forbedringer. ■

Instanser

Følgende instanser er (med få undtagelser) hentet fra TSPLIB hjemmesiden [9] og konverteret til et format der er nemt at indlæse. Alle instanser findes på kursets hjemmeside.

n	instans	beskrivelse
5	rand5	tilfældigt genererede kantvægte
10	rand10	tilfældigt genererede kantvægte
8	bornholm	afstande mellem otte byer på Bornholm
14	burma14	14 byer i Burma, geografisk afstand
17	gr17	17 byer i Tyskland
21	gr21	21 byer i Tyskland
24	gr24	24 byer i Tyskland
48	gr48	48 byer i Tyskland
120	gr120	120 byer i Tyskland
29	bays29	29 byer i Bayern (street distance)
29	bayg29	29 byer i Bayern (geographic distance)
42	swiss42	42 byer i Schweiz (Fricker)
17	br17	asymmetrisk TSP (Repetto)
34	ftv33	asymmetrisk TSP (Fischetti)
36	ftv35	asymmetrisk TSP (Fischetti)
39	ftv38	asymmetrisk TSP (Fischetti)
45	ftv44	asymmetrisk TSP (Fischetti)
48	ftv47	asymmetrisk TSP (Fischetti)
56	ftv55	asymmetrisk TSP (Fischetti)
71	ftv70	asymmetrisk TSP (Fischetti)
171	ftv170	asymmetrisk TSP (Fischetti)
48	ry48p	asymmetrisk TSP (Fischetti)
323	rbg323	Stacker crane application (Ascheuer)
358	rbg358	Stacker crane application (Ascheuer)
403	rbg403	Stacker crane application (Ascheuer)
443	rbg443	Stacker crane application (Ascheuer)
535	si535	TSP (M. Hofmeister)
1032	si1032	TSP (M. Hofmeister)

Den optimale løsningsværdi er angivet i hovedet af de fleste instanser. De nederste instanser kræver at programmets tabeller udvides.

Noter

Til opgaven benyttes et rammeprogram `tsp.c` som er skrevet i C og som varetager kommunikationen med CPLEX. Da der kun er nogle få CPLEX-licenser til rådighed på DIKU, vil rammeprogrammet højst bruge CPLEX i 60 sekunder, hvorpå licensen frigives. CPLEX licenser er tilgængelige på Linux pc'er samt SUN maskiner.

For at benytte CPLEX er det nødvendigt at tilføje følgende linie i sin `.tcshrc` fil.

```
setenv ILOG_LICENSE_FILE /usr/local/stow/etc/cplex.ilm
```

eller på de nye maskiner:

```
setenv ILOG_LICENSE_FILE /opt/cplex/cplex70/etc/cplex.ilm
```

Rammeprogrammet findes på kursets hjemmeside sammen med en makefil til at oversætte det.

Der benyttes et meget simpelt interface til CPLEX: IP-modellen skrives til en fil `/tmp/infile.lp`, hvorpå rammeprogrammet kalder CPLEX med filen som inddata. Det kan være hensigtsmæssigt at tilføje sit eget login-navn til denne fil, for at undgå konflikt med andre brugere.

Det simple interface gør det nemt at finde fejl i IP-modellen, idet man kan anvende CPLEX interaktivt med den genererede input fil: Skriv `cplex` i kommandolinien, og indlæs datafilen med `read /tmp/infile.lp`. Såfremt der er syntaxfejl i IP-modellen vil CPLEX rapportere disse. Ellers kaldes `optimize` og CPLEX vil rapportere om modellen er ubegrænset (“unbounded”), har et tomt løsningsrum (“infeasible”), eller lignende.

Bemærk at edb-afdelingen er ved at installere nye førstedelsmaskiner, hvilket måske kan give problemer med afvikling af CPLEX på disse. Start derfor på opgaven i god tid. En liste over tilgængelige 1-dels maskiner kan fås med kommandoen `ng2h dell1-linux`.

Litteratur

- [1] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson, “Solution of a Large-Scale Traveling Salesman Problem,” *Operations Research*, 2 (1954) 393–410.
- [2] <http://www.caam.rice.edu/~bico/>, home page of Bill Cook at Rice University.
- [3] <http://www.keck.caam.rice.edu/concorde.html>.
- [4] E. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, eds., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, UK, 1985
- [5] A. Langevin, F. Soumis, and J. Desrosiers, Classification of travelling salesman formulations, *Oper. Res. Lett.*, 9 (1990), pp. 127-132.
- [6] C. E. Miller, A. W. Tucker, and R. A. Zemlin, Integer programming formulations and traveling salesman problems, *J. ACM*, 7 (1960), pp. 326-329.
- [7] M. Padberg, and T.-Y. Sung, An analytical comparison of different formulations of the travelling salesman problem, *Math. Programming*, 52 (1991), pp. 315-357.
- [8] L. A. Wolsey, *Integer Programming*, Wiley, Chichester, UK, 1999.
- [9] <http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/>
- [10] <http://rodin.wustl.edu/~kevin/dissert/node11.html>