

DAT2A godkendelsesopgave O2

Heltalsprogrammering

David Pisinger, Forår 2002

*I opgaven skal der konstrueres og implementeres en generel algoritme som kan løse heltalsprogrammeringsproblemer. Algoritmen implementeres i C (eller C++). Besvarelsen består af et tekstafsnit, hvor nedenstående opgaver besvares, samt udskrift af det **kommenterede** program. Opgaven er individuel, hvorfor både tekstafsnit og program skal være selvstændige. Besvarelsen afleveres i to eksemplarer (tre forsider). Læs hele opgaveteksten inden implementeringen påbegyndes, og bemærk specielt kommentarerne sidst i teksten.*

Denne opgave stilles onsdag 24. april 2002 og skal afleveres onsdag 8. maj 2002 kl. 12.00 i førstedelen. De rettede opgaver kan afhentes i førstedelen 17. maj. Eventuel genaflevering sker senest 24. maj, således at de rettede opgaver herefter kan afhentes 31. maj.

Indledning

Heltalsprogrammering er en af de hyppigst anvendte metoder til formulering og løsning af \mathcal{NP} -hårde optimeringsproblemer. Dette beror på problemets generalitet, idet adskillige problemer kan formuleres som et heltalsprogrammeringsproblem. Igennem de seneste årtier er der lagt et stort arbejde i at udvikle heltalsprogrammeringsløsere, og kommercielle algoritmer er i stand til at løse problemer med flere hundrede (heltallige) beslutningsvariable.

I heltalsprogrammering (IP) arbejder man med en række beslutningsvariable $x_j \in \{0, 1, 2, \dots\}$ for $j = 1, \dots, n$. Prisen c_j svarende til beslutningsvariabel x_j angiver fortjenesten ved at producere hver enhed af x_j , og det antages at denne fortjeneste skal maksimeres. Løsningsmængden er begrænset af et antal lineære begrænsninger som angivet nedenfor

$$\begin{aligned} & \text{maximize} && \sum_{j=1}^n c_j x_j \\ & \text{subject to} && \sum_{j=1}^n a_{ij} x_j \leq b_i && \text{for } i = 1, \dots, m \\ & && x_j \in \{0, 1, 2, \dots\} && \text{for } j = 1, \dots, n. \end{aligned} \tag{1}$$

\mathcal{NP} -fuldstændighed

Opgave 1 Vis at generel heltalsprogrammering (1) er \mathcal{NP} -hårdt ved reduktion fra SUBSET-SUM. Det er tilstrækkeligt at vise at SUBSET-SUM i optimeringsversionen er indeholdt som specialtilfælde af IP. ■

Vi kan betragte generelt heltasprogrammering som et afgørlighedsproblem IP-DECISION ved at spørge om maximum til (1) er større end en given værdi.

Opgave 2 Vis at IP-DECISION er et \mathcal{NP} -fuldstændigt problem selv hvis alle koefficienter a_{ij}, b, c_j skal være 0, 1 eller -1. ■

Øvre grænseværdi for IP

Et lineært programmeringsproblem (LP) kan skrives på formen

$$\begin{aligned} & \text{maximize} && \sum_{j=1}^n c_j x_j \\ & \text{subject to} && \sum_{j=1}^n a_{ij} x_j \leq b_i \quad \text{for } i = 1, \dots, m \\ & && x_j \geq 0 \quad \text{for } j = 1, \dots, n \end{aligned} \tag{2}$$

hvor alle variable x_j er reelle tal. Se gerne kapitel 29 i [1] for yderligere beskrivelse af lineær programmering.

Opgave 3 Vis at LP på formen (2) giver en øvre grænseværdi for IP givet ved (1). ■

Branch-and-bound algoritme for IP

Vi vil nu konstruere en branch-and-bound algoritme til løsning af IP. Hvis vi anvender dybde-først søgning kan algoritmen skrives som en rekursiv procedure der i hvert skridt opdeler løsningsrummet i to dele svarende til begrænsningerne

$$\sum_{j=1}^n a'_j x_j \leq \lfloor b' \rfloor \tag{3}$$

$$\sum_{j=1}^n a'_j x_j \geq \lceil b' \rceil \tag{4}$$

Algoritmen kan skitseres som:

```

BRANCHANDBOUND( $A, b, c$ )
  løs problemet (2) for nuværende værdi af  $A, b, c$ , lad løsningsværdien være  $z$ 
  hvis (2) ikke har nogen løsning then return
  hvis  $z \leq z^*$  then return /* upper bound test */
  hvis alle  $x_j$  i løsningen til (2) er heltallige then  $z^* \leftarrow z; x^* \leftarrow x$ ; return
  opdel løsningsrummet i to dele svarende til de to begrænsninger (3) og (4)
  kald BRANCHANDBOUND rekursivt med begrænsning (3) tilføjet.
  kald BRANCHANDBOUND rekursivt med begrænsning (4) tilføjet.

```

Opgave 4 Beskriv hvordan du ville vælge koefficienterne a'_j og b' i ulighederne (3) og (4). Bevis at opdelingen af løsningsrummet er en lovlig opdeling af løsningsrummet. ■

Opgave 5 Godtgør at opdelingsprocessen (3) og (4) med dit valg af koefficienter vil konvergere mod en heltallig løsning. ■

Opgave 6 Implementer en fuldstændig version af BRANCHANDBOUND. ■

Testkørsler

Opgave 7 Brug den udviklede algoritme til at løse følgende \mathcal{NP} -hårde problemer optimeringsproblemer formuleret som IP problemer. Alle instanser findes på DAT2A hjemmesiden.

navn	problem type	begrænsninger	variable
mini.lp	2D eksempel	3	2
knapsack10.lp	knapsack problem	11	10
knapsack20.lp	knapsack problem	21	20
knapsack30.lp	knapsack problem	31	30
knapsack40.lp	knapsack problem	41	40
subsetsum10.lp	subset sum problem	11	10
subsetsum20.lp	subset sum problem	21	20
subsetsum30.lp	subset sum problem	31	30
subsetsum40.lp	subset sum problem	41	40
clique6.lp	klike problem 6 knuder	53	21
clique7.lp	klike problem 7 knuder	72	28
clique8.lp	klike problem 8 knuder	94	36
clique9.lp	klike problem 9 knuder	119	45
tsp4.lp	traveling salesman 4 byer	28	16
tsp5.lp	traveling salesman 5 byer	50	25
tsp6.lp	traveling salesman 6 byer	89	36
tsp7.lp	traveling salesman 7 byer	161	49

Rapporter optimale løsning, antal besøgte knuder i branch-and-bound træet, samt løsnings-
ningstid. Kommenter resultaterne. ■

Ministryger løst med heltalsprogrammering

Ministryger (*Minesweeper*) er et simpelt spil, hvor det er deltagerens opgave at bestemme en række miners position ud fra oplysninger om antal omkringliggende miner for en række allerede afdækkede felter. For en detaljeret beskrivelse af ministryger spillet se f.eks.

www.cse.unsw.edu.au/~rizwans/Minesweeper/instructions.html

Samme sted er der mulighed for at afprøve spillet. Kaye [4] viste at MINESWEEPER-DECISION er \mathcal{NP} -hårdt. MINESWEEPER-DECISION problemet skal afgøre om det er muligt at placere miner på spillepladen således at alle begrænsninger overholdes.

Opgave 8 Betragt følgende minetryger problem. Det oplyses at der er 10 miner gemt på spillepladen. Formuler problemet som et heltalsprogrammeringsproblem på formen (1). Læs evt. kapitel 29.1 i [1] for teknikker til opskrivning af standard form.

			1	
	4	2	2	1
			4	3
2				

■

Opgave 9 Hvorledes skal algoritmen til løsning af heltalsprogrammeringsproblemer modificeres så den finder samtlige semi-kritiske løsninger til et problem? Løs ovenstående minetryger problem ved brug af den modificerede algoritme. Rapportér de fundne løsninger.

Såfremt tiden tillader det, løs samme problem hvis samtlige løsninger til problemet skal findes. ■

Opgave 10 Vi betragter optimeringsversionen af minetryger MAX-MINESWEEPER, dvs. givet en spilleplade find det maksimale antal miner der kan være på pladen så alle givne begrænsninger overholdes. Formuler nedenstående problem som et heltalsprogrammeringsproblem i maksimeringsform og løs det med din algoritme. Flagene markerer positionen af kendte miner. Det er ikke nødvendigt at finde samtlige løsninger til problemet.

				1
1			3	
	6			
			4	3
			2	1

■

Konkurrenceopgaver

Konkurrencen består i at løse størst mulige instanser af MAX-MINESWEEPER problemet. Se instrukserne på hjemmesiden såfremt du ønsker at deltage i konkurrencen.



Kommentarer

Opgaven har primært til hensigt at give indsigt i løsning af \mathcal{NP} -hårde problemer, mens ren programmering kun er et sekundært mål. På hjemmesiden for DAT2A

<http://www.diku.dk/teaching/2002f/dat2a/>

ligger derfor en færdig ramme for programmet som skal anvendes i besvarelsen. Her findes de nødvendige rutiner til tidsmåling og indlæsning af data, ligesom ovenstående programskitser er implementeret. Tabeller adresseres så $a[1][1]$ svarer til a_{11} for at opnå et læseligt program. Rammen er afprøvet på DIKU's linux computere, og det anbefales at bruge disse ved udarbejdelse af besvarelsen. Selv om rammeprogrammet kan virke stort er det kun nødvendigt at skrive algoritmen BRANCHANDBOUND.

Noter

Edmonds nævnes ofte som en af fortalere for IP på lineær form. Edmonds indså tidligt at takket være LP har vi en generel og effektivt beregnelig grænseværdi funktion for IP-problemer, hvilket muliggør en generelt løser.

Igennem de sidste 10-20 år er der sket store landvindinger indenfor løsning af IP-problemer. Specielt har computerens stadigt stigende hastighed gjort det muligt at løse store problemer, men også på det algoritmiske område er vores teknikker blevet væsentligt bedre. Crowder, Johnson, Padberg [2] var de første til at løse store problemer med adskillige tusinde binære variable. Van Roy, Wolsey [3] viste hvorledes principperne kunne generaliseres til problemer hvor både heltallige og lineære variable indgår.

Wolsey [6] nævner følgende algoritmiske forbedringer: Tilføjelse af ekstra uligheder til et problem resulterer i strammere grænseværdier fra LP-relaxering; Teknikker fra constraint programming gør det muligt løbende at reducere udfaldsrummet af de implicerede variable; Hybride varianter af branching-strategien er udviklet så man hurtigt finder frem til en god løsning med dybde-først søgning og derefter bruger bedste-først søgning; LP-løsere er blevet stadig hurtigere således at grænseværdiberegningen tager mindre tid.

Nogle af de mest kendte kommercielle programmer til heltalsprogrammering er CPLEX [8], Xpress MP [7], Mosek [9]. Sidstnævnte er dansk, og en gratis studenterversion kan hentes via Mosek's hjemmeside.

Hvis man vil udvikle en hurtigere IP-løser end skitseret ovenfor følger her en række hints:

- Hvis alle koefficienter i c er heltal, må den optimale løsningsværdi også være heltallig. Dette kan bruges til at stramme grænseværditesten i algoritmen BRANCHANDBOUND.
- Ofte er a -matricen tynd (dvs. de fleste indgange har værdi 0). Dette kan udnyttes til at implementere en mere effektiv version af simplex algoritmen.

- Det er en stor besparelse at løse det dual problem af (2), jf. afsnit 29.4 i [1]. En tidligere fundet LP-løsning vil fortsat være lovlig for det duale problem når der tilføjes yderligere uligheder af formen (3) og (4) i branching-processen. Dette betyder at man ikke skal køre simplex-algoritmen fra starten, men kan tage udgangspunkt i den forrige løsning og blot foretage en mindre reoptimering.
- Det kan betale sig at finde en god start-løsning z . Grådige heuristikker, metaheuristikker m.m. kan benyttes.
- Man kan forsøge at eliminere variable fra et problem a-priori. For en binær variabel x_i udledes en øvre grænseværdi u for problemet (2) med den ekstra begrænsning $x_i = 0$. Såfremt $u \leq z$ kan vi konkludere at $x_i = 1$ i enhver bedre løsning end den hidtil fundne, og vi har dermed elimineret variabelen. På symmetrisk vis kan man foretage en sådan test med den ekstra begrænsning $x_i = 1$ tilføjet. For ikke-binære variable generaliseres princippet nemt.
- Det kan være hensigtsmæssigt at tilføje ekstra uligheder til problemet (1). F.eks. er cover-uligheder [6] nemme at beregne, samtidig med at de ofte kan stramme grænseværdierne kraftigt.

Heltalsprogrammeringsløseren kan også forbedres således at den accepterer begrænsninger med $=$ og \geq . Sådanne begrænsninger konverteres nemt til formen (1).

Litteratur

- [1] T.H.Cormen, C.E.Leiserson, R.L.Rivest, C.Stein "Introduction to Algorithms", MIT-press (2001).
- [2] H. Crowder, E.L.Johnson, M.Padberg "Solving large-scale zero-one linear programming problems", *Operations Research*, **31**, 803–834 (1983).
- [3] T.J. Van Roy, L.A. Wolsey "Solving mixed 0-1 programs by automatic reformulation", *Operations Research*, **35**, 45–57 (1987).
- [4] R. Kaye (2000) "Minesweeper is NP-complete!" *Mathematical Intelligencer*, **22**, 9–15 (2000).
- [5] R. Kaye, "Richard Kaye's Minesweeper Pages"
<http://www.mat.bham.ac.uk/R.W.Kaye/minesw/minesw.htm>.
- [6] Wolsey, "Integer Programming", Wiley (1998).
- [7] Xpress MP <http://www.dash.co.uk/products.html>
- [8] CPLEX <http://www.ilog.com/products/cplex/>
- [9] MOSEK optimization software <http://www.mosek.com/>