Heltalsprogrammering

Man ska hyra in ett antal kranar för att lyfta saker under byggnationen av det nya Studenthuset.

Det finns två olika sorters kranar, med olika kapacitet och kostnad.

x_j anger hur många kranar av sort j man hyr in.

Man vill maximera total lyftkapacitet under bivillkoret att kostnaden inte överstiger den budgeterade.

$$\begin{array}{lll} \max & z = & 10x_1 + 15x_2 \\ \text{då} & & 4x_1 + 7x_2 \leq 22 \\ & & x_1, x_2 \geq 0, \text{heltal} \end{array}$$

Ett linjärt problem plus heltalskrav.

Vi kan lösa LP-relaxationen med Simplexmetoden,

man får då kanske inte heltal.

Heltalsprogrammering

$$\begin{array}{llll}
\max & z = & \sum_{j=1}^{n} c_j x_j & \\
\text{då} & & \sum_{j=1}^{n} a_{ij} x_j \leq b_i & i = 1, \ldots, m \\
& & x_j \geq 0 & j = 1, \ldots, n \\
& & x_j \text{ heltal} & j = 1, \ldots, n \\
& & & \\
\text{ofta} & & x_j \leq u_j & j = 1, \ldots, n
\end{array}$$

Oftast c, A, b heltal. Ibland $u_j = 1$.

Obs: Tillåtna mängden ej konvex.

Speciell användning av heltalsvariabler

Logiska beslut:

$x_1 = 1$ om vi väljer alternativ 1, $x_2 = 1$ om vi väljer alternativ 2.

Alternativ 1 kostar c_1, alternativ 2 kostar c_2.

Vi måste göra ett av dem.

min $c_1 x_1 + c_2 x_2$ då $x_1 + x_2 = 1$, $x_1 \in \{0, 1\}$, $x_2 \in \{0, 1\}$ plus andra bivillkor.

Vi måste göra minst ett av dem: $x_1 + x_2 \geq 1$.

Vi får göra högst ett av dem: $x_1 + x_2 \leq 1$.

Vi måste göra 5 av 17 alternativ: $\displaystyle\sum_{j=1}^{17} x_j = 5$.

Speciell användning av heltalsvariabler

Antingen-eller-villkor:

Ex: I ord: Antingen $2x_1 + 3x_2 \leq 5$ eller $5x_1 + 2x_2 \leq 6$.

Formulera som följer (med $M >> x$)

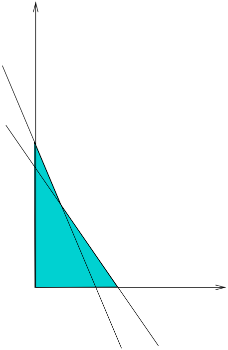
$$2x_1 + 3x_2 \leq 5 + M(1 - y_1),$$
$$5x_1 + 2x_2 \leq 6 + M(1 - y_2),$$
$$y_1 + y_2 = 1,$$
$$y_1 \in \{0, 1\},\ y_2 \in \{0, 1\}.$$

Utvidgning: minst 4 av 13 bivillkor skall gälla. $\sum_{j=1}^{13} y_j = 4.$

Antingen-eller-villkor: Exempel

Antingen $2x_1 + 3x_2 \leq 5$ eller $5x_1 + 2x_2 \leq 6$.

$M = 5$:
$2x_1 + 3x_2 \leq 5 + 5(1 - y_1)$,
$5x_1 + 2x_2 \leq 6 + 5(1 - y_2)$,
$y_1 + y_2 = 1$, $y_1 \in \{0, 1\}$, $y_2 \in \{0, 1\}$.



Speciell användning av heltalsvariabler

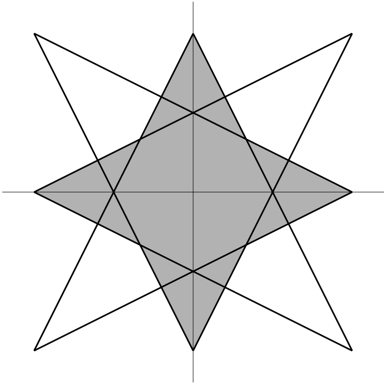
Icke-konvexa tillåtna områden, t ex villkorliga undre gränser:

I ord: $x = 0$ eller $x \geq L$.

Formulera som

$Ly \leq x \leq My$, $y \in \{0, 1\}$.

Speciell användning av heltalsvariabler



Ett icke-konvext område. (Unionen av två konvexa områden.)

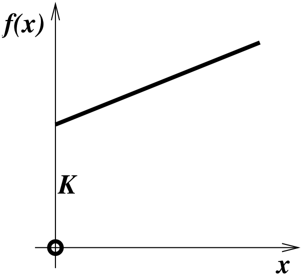
$y_1 + y_2 = 1$, $y_1 \in \{0, 1\}, y_2 \in \{0, 1\}$ ger $x \in X_1 \cup X_2$

Utmaning: Konstruera på detta sätt ett stiliserat M, C och E.

Speciell användning av heltalsvariabler

Fasta kostnader:

I ord: Kostnaden är 0 om $x = 0$, men $K + cx$ om $x > 0$.



I ord: Fast kostnad: $f(x) = \begin{cases} 0 \text{ om } x = 0 \\ K + cx \text{ om } x > 0 \end{cases}$

Modellera som: $f(x) = cx + Ky \quad \text{där} \quad x \leq My,\ y \in \{0, 1\}$

Förutsätter minimering.

Heltalsprogrammering: Speciella problem

Kappsäcksproblemet:

$$
\begin{array}{lll}
\max & z = & \displaystyle\sum_{j=1}^{n} c_j x_j \\
\text{då} & & \displaystyle\sum_{j=1}^{n} a_j x_j \leq b \\
 & & 0 \leq x_j \leq u_j, \text{ heltal} \quad j = 1, \ldots, n
\end{array}
$$

Ett bivillkor. (Ickenegativa koefficienter.)

LP-relaxationen kan lösas med en greedy-metod:

Sortera enligt $\max_j (c_j / a_j)$. Fyll på bäst först.
(Härled med LP-dualitet.)

Övertäckningsproblemet

Vilka åtgärder skall göras för att varje effekt skall erhållas?

Indata: $a_{ij} = \begin{cases} 1 \text{ om effekt } i \text{ ges av åtgärd } j \\ 0 \text{ om inte} \end{cases}$

Åtgärd j kostar c_j. Minimera totalkostnaden.

Variabeldefinition: $x_j = \begin{cases} 1 \text{ om åtgärd } j \text{ utförs} \\ 0 \text{ om inte} \end{cases}$

$$
\begin{array}{lll}
\min \quad z = & \sum_{j=1}^{n} c_j x_j & \\
\text{då} & \sum_{j=1}^{n} a_{ij} x_j \geq 1 & i = 1, \ldots, m \\
& x_j \in \{0, 1\} & j = 1, \ldots, n
\end{array}
$$

Uppdelningsproblemet (partioneringsproblemet): Likhet i bivillkoren.

Lokaliseringsproblemet

Anläggning i: Fast kostnad f_i, kapacitet s_i.

Kund j: Efterfrågan d_j.

Transportkostnad från anläggning i till kund j: c_{ij} per enhet.

Vilka anläggningar skall byggas?

Hur mycket ska varje anläggning skicka till varje kund?

Minimera totala byggkostnader och transportkostnader.

Variabeldefinition:

$$y_i = \begin{cases} 1 & \text{om anläggning } i \text{ byggs} \\ 0 & \text{om inte.} \end{cases}$$

x_{ij} = antal enheter som skickas från anläggning i till kund j

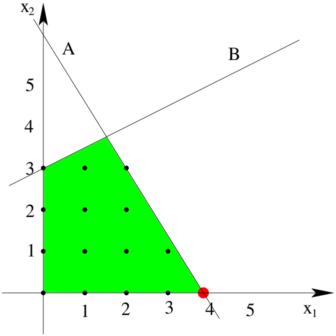
Lokaliseringsproblemet

$$
\begin{array}{llcll}
\min & \displaystyle\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} f_i y_i & & & \\
\text{då} & \displaystyle\sum_{j=1}^{n} x_{ij} & \leq & s_i y_i & \text{för alla } i \qquad (1) \\
 & \displaystyle\sum_{i=1}^{m} x_{ij} & = & d_j & \text{för alla } j \qquad (2) \\
 & x_{ij} - s_i y_i & \leq & 0 & \text{för alla } i,j \qquad (3) \\
 & x_{ij} & \geq & 0 & \text{för alla } i,j \qquad (4) \\
 & y_i & \in & \{0,1\} & \text{för alla } i \qquad (5)
\end{array}
$$

Behövs bivillkor 3? (se lab 4)

Heltalsprogrammering: Exempel

$$
\begin{array}{llrrll}
\max & z = & 30x_1 & +18x_2 & & \\
\text{då} & & 8x_1 & +5x_2 & \leq 31 & (A) \\
 & & -x_1 & +2x_2 & \leq 6 & (B) \\
 & & x_1, x_2 \geq 0 & & & \\
 & & x_1, x_2 \text{ heltal} & & &
\end{array}
$$



LP-relaxationen: $x_1 = 3.875$, $x_2 = 0$ och $z_{LP} = 116.25$.

Heltalsprogrammering: Exempel

LP-relaxationen: $x_1 = 3.875$, $x_2 = 0$ och $z_{LP} = 116.25$.

Sats

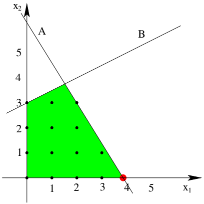
LP-relaxationen ger en optimistisk uppskattning av z^*.
För max-problem: $z_{LP} \geq z^*$.

Alltså $z^* \leq 116.25$

Alla koefficienter i målfunktionen är heltal. Alla variabler skall vara heltal.
$\Rightarrow z^*$ heltal. Avrunda z_{LP} neråt. (Avrunda inte x.)

$\because z^* \leq 116$

Heltalsprogrammering: Exempel



LP-relaxationen: $x_1 = 3.875$, $x_2 = 0$ och $z_{LP} = 116.25$.

Avrundning: $x_1 = 4$, $x_2 = 0$. Ej tillåten.

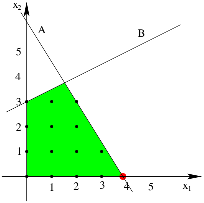
Avrundning till närmaste tillåtna punkt: $x_1 = 3$, $x_2 = 0$. $z = 90$.

Sats

Varje tillåten lösning ger en pessimistisk uppskattning av z^*. För max-problem: $z \leq z^*$.

$\because 90 \leq z^* \leq 116$

Heltalsprogrammering: Exempel



Förbättra gränserna:

Enkel lokal sökning: Ändra en variabel i taget.

Tillåtet att öka x_2 till 1. $\Rightarrow z = 108$.

Ingen enkel ändring ger förbättring: Lokalt optimum.

$\because 108 \leq z^* \leq 116$

(Optimum: $x_1 = 2$, $x_2 = 3$ och $z^* = 114$.)

Heltalsprogrammering: Vägval

Optimerande / heuristisk?

LP-baserad / kombinatorisk?

Trädsökning / plansnittning?

Heltalsprogrammering: Metodprinciper

Optimerande metod: Finner garanterat optimum (och bevisar det).

Fullständig uppräkning av alla möjliga lösningar: Nej!

Heuristik: Snabbare. Inga garantier. Bevisar ej optimalitet.

Konstruktiva metoder.

Sökmetoder.

Lagrangerelaxation.

Heltalsprogrammering: Metodprinciper

LP-baserad metod:

- Lös LP-relaxationen.
- Modifiera LP-problemet.
- Lös om.

Kombinatorisk metod:

- Utgå från problemets kombinatoriska struktur. (0/1, graf)

Heltalsprogrammering: Metodprinciper

Trädsökning:

- Relaxation.
- Förgrening: Dela upp problemet i flera (disjunkta) problem.
- Rekursivt.
- Kapa grenar.
- Avsöka hela trädet.
- Övre och undre gränser (för z^*).
- "Branch-and-bound"

Plansnittning:

- Lägg till bivillkor som skär bort LP-optimum, men inte någon tillåten heltalslösning.
- Ett LP-problem.
- Ökande storlek.

Heltalsprogrammering: LP-baserad trädsökning

Land-Doig-Dakins metod:

Relaxation: LP-relaxationen.

Förgrening: Skapa två nya problem:
Ett där $x_j \leq \lfloor \bar{x}_j \rfloor$ och ett där $x_j \geq \lceil \bar{x}_j \rceil = \lfloor \bar{x}_j \rfloor + 1$.

Kapa grenar som:

- Inte kan ge någon tillåten lösning.
- Inte kan ge någon bättre lösning.
- Ger heltalslösning.

Varje ny förgrening innebär ytterligare begränsningar. Optimistiska uppskattningen z_{LP} kan ej bli bättre, när vi går djupare i trädet.

Varje tillåten lösning ger en pessimistisk uppskattning, $\underline{z}$, som gäller i hela trädet.

Heltalsprogrammering: Land-Doig-Dakins metod

Undersökning av LP-problem:

- Om tillåten lösning saknas: Kapa grenen.
- Om z_{LP} är *sämre* än känd lösning $\underline{z}$: Kapa grenen.
- Om lösningen är *heltalig* och z_{LP} är *bättre* än känd lösning $\underline{z}$: Spara lösningen. Kapa grenen.
- Om lösningen inte är heltalig och z_{LP} är bättre än känd lösning $\underline{z}$: Förgrena.

Heltalsprogrammering: Land-Doig-Dakins metod

Algoritm (max):

1. Om inget oavsökt problem återstår: Stopp.

Annars välj ett oavsökt problem.

2. Lös LP-relaxationen (simplexmetoden).

Om den saknar tillåten lösning: Kapa grenen. Gå till 1.

3. Om $z_{LP} \leq \underline{z}$: Kapa grenen. Gå till 1.

4. Om lösningen är heltalig: Spara lösningen och kapa grenen. Gå till 1.

5. Förgrena: Välj en variabel, x_j, som ej är heltal, $\bar{x}_j$. Skapa två nya problem:

Ett där $x_j \leq \lfloor \bar{x}_j \rfloor$ och ett där $x_j \geq \lceil \bar{x}_j \rceil = \lfloor \bar{x}_j \rfloor + 1$.
Gå till 1.

Heltalsprogrammering: Land-Doig-Dakins metod

Specificera:

- Förgreningsstrategi (vilken variabel först).
- Nodavsökningsstrategi (vilken nod/problem först).

Exempel:
Djup-först.

Förgrena över variabeln med störst fraktionell del först.
Gå ner i ($\geq$)-grenen först.

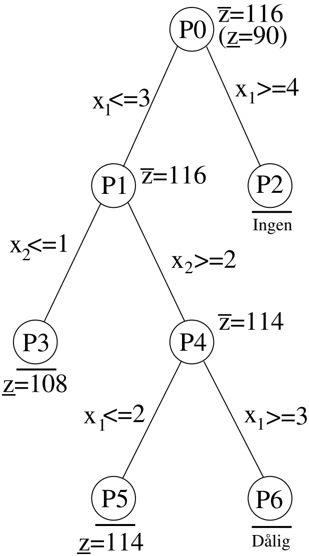
Förgrena över variabeln med minst fraktionell del först.
Gå ner i ($\leq$)-grenen först.

Land-Doig-Dakins metod: Exempel

$$
\begin{array}{llrcll}
\max & z = & 30x_1 + 18x_2 & & & \\
\text{då} & & 8x_1 + \ 5x_2 & \leq & 31 & (A) \\
& & -x_1 + \ 2x_2 & \leq & 6 & (B) \\
& & x_1, x_2 \geq 0 \text{ heltal} & & &
\end{array}
$$

LP-relaxationen P0: $x_1 = 3.875$, $x_2 = 0$ och $z_{LP} = 116.25$. $\bar{z} = 116$.
Valfritt: Avrunda neråt: $x_1 = 3$, $x_2 = 0$ och $z = 90$. $\underline{z} = 90$.
Förgrena: Skapa P1: P0 + $(x_1 \leq 3)$ och P2: P0 + $(x_1 \geq 4)$.
Lös P1: $x_1 = 3$, $x_2 = 1.4$ och $z = 116$. $\bar{z} = 116$.
Förgrena: Skapa P3: P1 + $(x_2 \leq 1)$ och P4: P1 + $(x_2 \geq 2)$.
Lös P3: $x_1 = 3$, $x_2 = 1$ och $z = 108$. Heltal. $\underline{z} = 108$. Kapa.
Lös P4: $x_1 = 2.625$, $x_2 = 2$ och $z = 114.75$. $\bar{z} = 114$.
Förgrena: Skapa P5: P4 + $(x_1 \leq 2)$ och P6: P4 + $(x_1 \geq 3)$.
Lös P5: $x_1 = 2$, $x_2 = 3$ och $z = 114$. Heltal. $\underline{z} = 114$. Kapa.
P6 har $\bar{z} = 114$. Kapa. (Saknar tillåten lösning.)
Lös P2: Saknar tillåten lösning. Kapa.

Land-Doig-Dakins metod: Exempel



Kranexemplet

$$\begin{array}{lll} \max & z = & 10x_1 + 15x_2 \\ \text{då} & & 4x_1 + 7x_2 \leq 22 \\ & & x_1, x_2 \geq 0, \text{heltal} \end{array}$$

Dividera målfunktionen med 5: $\max z = 2x_1 + 3x_2$

Lös LP-relaxationen: $x_1 = 11/2 = 5.5$, $x_2 = 0$, $z = 11$, vilket ger $\bar{z} = 11$.

Förgrena över x_1: P1 = P0+($x_1 \leq 5$), P2 = P0+($x_1 \geq 6$).

P1: $x_1 = 5$, $x_2 = 2/7$, $z = 10 + 6/7$, vilket ger $\bar{z} = 10$.

Förgrena över x_2: P3 = P1+($x_2 \leq 0$), P4 = P1+($x_2 \geq 1$).

P3: $x_1 = 5$, $x_2 = 0$, $z = 10$, tillåten lösning, vilket ger $\underline{z} = 10$.

P4: Kapa, ty $\bar{z} = \underline{z} = 10$.

P2: Saknar tillåten lösning.

Trädet avsökt, optimum (från P3): $x_1 = 5$, $x_2 = 0$, $z = 10$.

I ord: Hyr in 5 kranar av sort 1.

Kombinatoriskt baserad trädsökning

Balas metod för 0/1-problem: (Constraint Programming)

Undersök ett bivillkor i taget, cykliskt.

Observation: Det som är förbjudet i ett bivillkor är inte tillåtet.
Ta bort förbjudna möjligheter (variabelvärden), genom att fixera variabler.
Är $x_7 = 1$ omöjligt, så gäller $x_7 = 0$. Är $x_7 = 0$ omöjligt, så gäller $x_7 = 1$.

Kapa grenar som inte kan ge någon tillåten lösning.

Gör om målfunktionen till ett bivillkor:
Kräv bättre lösning, för max-problem: $c^T x \geq \underline{z} + 1$.

Förgrening: Skapa två problem: Ett med $x_j = 0$ och ett med $x_j = 1$.

Kombinatoriskt baserad trädsökning: Bivillkorsfixering

Undersökning av delproblem (bivillkor):

- Om tillåten lösning saknas: Kapa grenen.
- Om tillåten lösning saknas då $x_j = 1$: Fixera x_j till 0.
- Om tillåten lösning saknas då $x_j = 0$: Fixera x_j till 1.
- Om inget av ovanstående för något bivillkor: Förgrena.

Fixering av variabel innebär att vi slipper en av två grenar.

Constraint Programming: Samma princip, krångligare bivillkor.
(Bivillkor = subrutin.)

Bivillkorsfixering: Exempel

$$\begin{array}{lcrl} 3x_1 + 4x_2 + 8x_3 - 5x_4 & \leq & 10 & (1) \\ 5x_1 + 6x_2 + 5x_3 + 10x_4 & \leq & 14 & (2) \\ 2x_1 - 4x_2 - 3x_3 + 3x_4 & \leq & -2 & (3) \end{array}$$

Inga fixeringar. Förgrena över x_1. P1: $x_1 = 1$:

(1): $3 + 4x_2 + 8x_3 - 5x_4 \leq 10$: $4x_2 + 8x_3 - 5x_4 \leq 7$. Ger inget.

(2): $5 + 6x_2 + 5x_3 + 10x_4 \leq 14$: $6x_2 + 5x_3 + 10x_4 \leq 9$. Fixera $x_4 = 0$.

(3): $2 - 4x_2 - 3x_3 \leq -2$: $-4x_2 - 3x_3 \leq -4$. Fixera $x_2 = 1$.

(1): $3 + 4 + 8x_3 \leq 10$: $8x_3 \leq 3$. Fixera $x_3 = 0$.

Allt fixerat. Kolla alla bivillkor en gång till.

(2): $5 + 6 \leq 14$: OK.

(3): $2 - 4 \leq -2$: OK.

(1): $3 + 4 \leq 10$: OK.

Tillåten lösning funnen: $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 0$.

(Nu har vi grenen med $x_1 = 0$ kvar.)

Bivillkorsfixering: Exempel

		5	7		2	4		
7	9		1		8	6		
6		2		4				3
1	7		5		3			8
							2	1
	8	3	4		9		7	6
9	2	1	3			8		
4		7	8					5
		8		7	6		9	

Fler än man skulle tro använder bivillkorsfixering...

Bivillkorsfixering: Exempel: Sudoku

Variabeldefinition: $x_{ijk} = 1$ om siffran k ska stå i position (i, j).

Bivillkor:

$$\sum_j x_{ijk} = 1 \quad \text{för alla } i,\ k \quad (\text{siffra } k \text{ en gång i rad } i)$$

$$\sum_i x_{ijk} = 1 \quad \text{för alla } j,\ k \quad (\text{siffra } k \text{ en gång i kolumn } j)$$

$$\sum_k x_{ijk} = 1 \quad \text{för alla } i,\ j \quad (\text{en siffra i position } (i, j))$$

$$\sum_{3p-2 \leq i \leq 3p,} \ \sum_{3q-2 \leq j \leq 3q} x_{ijk} = 1 \quad \text{för } p = 1, \ldots, 3,\ q = 1, \ldots, 3, \text{ alla } k$$

$$(\text{siffra } k \text{ en gång i mindre kvadrat } (p, q))$$

$x_{ijk} \in \{0, 1\}$ för alla i, j, k.

$x_{ijk} = \bar{x}_{ijk}$ för vissa givna i, j, k.

Ingen målfunktion.

Bivillkorsfixering: Exempel: Sudoku

Givna siffror:
Om $x_{i'j'k'} = 1$ så fixeras alla andra variabler i samma bivillkor till noll.

Ingen mer siffra k' i rad i'.

Ingen mer siffra k' i kolumn j'.

Ingen annan siffra i position (i', j').

Ingen mer siffra k' i mindre kvadrat (p, q).

När siffra k bara har en möjlighet kvar i en rad/kolumn/mindre kvadrat:
    Lås den och fixera på ovanstående sätt.

När position (i, j) bara har en möjlighet kvar:
    Lås den och fixera på ovanstående sätt.

För lätt sudoku ger detta alltid en unik lösning.

Metod: Låt X_{ij} vara alla siffror som kan placeras i position (i, j). Först $X_{ij} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Ta sedan bort de som inte kan vara där.

Bivillkorsfixering: Utvidgade tester för speciella strukturer

Utvidgade tester för Sudoku:

Två siffror ska vara i två positioner, men vi vet inte vilken.

T.ex. $x_{111} + x_{132} = 2$ (siffra 1 och 2 ska vara i pos (1,1) och (1,3)).

Eliminera siffra 1 och 2 från alla andra positioner i rad 1, och lilla kvadrat (1,1), samt eliminera alla andra siffror från pos (1,1) och (1,3).

Bygger på strukturen hos problemet.

Övertäckningsproblemet: min $c^T x$ då $Ax \geq 1$, x binär. Reduktionstester:

- En rad endast nollor: Ingen tillåten lösning.
- Kolumn j endast nollor: Sätt $x_j = 0$ och stryk kolumn j.
- Rad i endast en etta, i kolumn j: Sätt $x_j = 1$, stryk kolumn j. Stryk alla rader med ettor i kolumn j.
- Rad i dominerar rad k (dvs. $a_{ij} \geq a_{kj}$ för alla j): Stryk rad i.
- Kolumn j dominerar kolumn k (dvs. $a_{ij} \geq a_{ik}$ för alla i och $c_j \leq c_k$): Sätt $x_k = 0$ och stryk kolumn k.

Kombinatoriskt baserad trädsökning för TSP

Relaxation: Billigaste uppspännande 1-träd.

Finn billigaste uppspännande träd för noderna 2 - n. Addera de två billigaste bågarna till nod 1.

Ger en cykel genom nod 1 som kan vara för liten. Nod 1 får rätt valens, alla andra noder kan få fel.

Motsvarar relaxation av valensvillkoren för alla noder utom en, samt av vissa subtursförbjudande villkor.

Ger undre gräns.

Övre gräns fås från en handelresandetur (tillåten lösning).

Kombinatoriskt baserad trädsökning för TSP

Förgrening:

Syftar till att ge alla noder rätt valens.

Förbjud en av de bågar som går in till en nod med för hög valens.

Förgrening: För varje båge som ansluter till noden: Skapa ett problem där bågen är förbjuden ($x_{ij} = 0$).

Minst en av grenarna måste innehålla optimum.
För att få optimum i exakt en, tvinga med tidigare förbjudna bågar.

Valensen skall vara 2, vilket ger följande tre möjligheter.

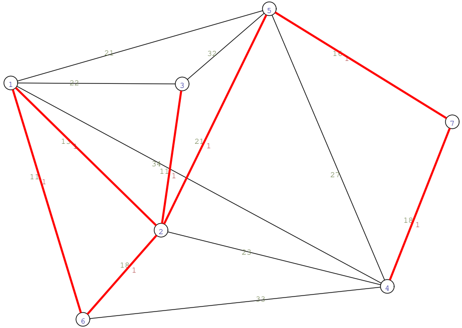
Gren 1: Förbjud första bågen.

Gren 2: Förbjud andra bågen och tvinga med första bågen.

Gren 3: Tvinga med första och andra bågarna samt förbjud alla andra bågar till noden.

Kombinatoriskt baserad trädsökning för TSP: Exempel

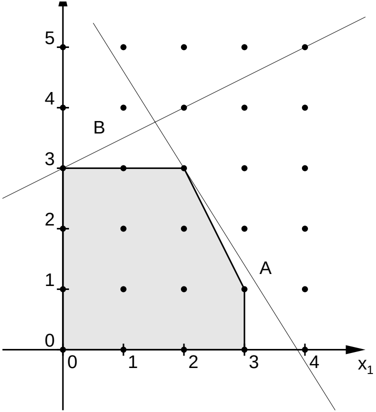
NEOPT



1-träd, kostnad: 108.

Plansnittning

Övergång från icke-konvext problem till konvext: Skapa konvexa höljet.



Plansnittning

Algoritm:

1. Lös LP-relaxationen.

2. Om lösningen är heltal: Stopp. Optimum.

3. Konstruera ett linjärt bivillkor som skär bort LP-optimum, men inga tillåtna heltalspunkter. Gå till 1.

Hur konstruera bivillkor? Helst fasetter.

- Gomorys metod (simplextablå).
- Problemspecifikt:

  Kappsäcksproblem: Minimala övertäckningar.

  TSP: (forskning)

Gomorysnitt

Plocka en rad ur optimala simplextablån: $\sum_j a_{ij} x_j = b_i$

Avrunda alla koefficienter i vänsterledet neråt.

Vänsterledet blir mindre och heltal.

Avrunda högerledet neråt.

Resulterande bivillkor, Gomorysnitt: $\sum_j \lfloor a_{ij} \rfloor x_j \leq \lfloor b_i \rfloor$

Exempel: $0.3x_1 + 4.5x_2 - 3.2x_3 + x_4 = 6.7$

Gomorysnitt: $4x_2 - 4x_3 + x_4 \leq 6$

Baslösning: $x_1 = 0$, $x_2 = 0$, $x_3 = 0$,$x_4 = 6.7$. Den lösningen skärs bort.

Minimala övertäckningar

Minsta mängden som är för stor.

Bivillkor: $\sum_j a_j x_j \leq b, \; x_j \in \{0, 1\}$ för alla j.

En övertäckning, S, är för stor, $\sum_{j \in S} a_j > b$.

Giltigt bivillkor: $\sum_{j \in S} x_j \leq |S| - 1$. (Alla i S får inte plats.)

Välj helst den minsta mängd som inte får plats.

Exempel: $7x_1 + 5x_2 + 6x_3 + 3x_4 \leq 12$.

Minimala övertäckningar:

$7x_1 + 5x_2 + 6x_3 + 3x_4 \leq 12$, $\{1, 3\} :\Rightarrow x_1 + x_3 \leq 1$.
$7x_1 + 5x_2 + 6x_3 + 3x_4 \leq 12$, $\{2, 3, 4\} :\Rightarrow x_2 + x_3 + x_4 \leq 2$.
$7x_1 + 5x_2 + 6x_3 + 3x_4 \leq 12$, $\{1, 2, 4\} :\Rightarrow x_1 + x_2 + x_4 \leq 2$.

Heltalsprogrammering

Kombinerade metoder: Trädsökning + plansnittning + förbehandling.

För att lösa riktigt stora problem, måste man utnyttja flera verktyg.

Förbehandling: Försök reducera problemstorleken eller omformulera problemet på ett bättre sätt.

Avlägsna redundanta bivillkor: Jämför möjliga värden på vänsterledet med högerledet. (Mindre LP.)

Fixera variabler: Balas metod, constraint programming. (Mindre LP, mindre träd.)

Modifiera koefficienter: Minska övre gränser, anpassa bivillkorskoefficienter. (Mindre träd.)

Addera logiska bivillkor: T ex minimala övertäckningar. (Mindre träd.)

Lab 4

Lös lokaliseringsproblemet med GLPK:

1. Gör en modellfil i GMPL-format.
2. Lös ett antal instanser. Notera hur jobbigt.
3. Lös LP-relaxationerna av samma instanser. Notera målfunktionsgap. Lösning?
4. Modifiera modellen.
5. Lös LP-relaxationerna av samma instanser. Notera målfunktionsgap. Lösning?
6. Lös samma instanser. Notera hur jobbigt.
7. Lös en instans för olika proportioner mellan fasta och rörliga kostnader.

Lab 5:
Gör en egen heuristik för samma problem. Implementera i Python eller Matlab.