

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.

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

Ett linjärt problem plus **heltalskrav**.

Vi kan lösa LP-relaxationen med Simplexmetoden, man får då kanske inte heltal.

## 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.

$\min c_1x_1 + c_2x_2$  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:  $\sum_{j=1}^{17} x_j = 5$ .

$$\max \quad z = \sum_{j=1}^n c_j x_j$$

$$\begin{aligned} \text{då} \quad & \sum_{j=1}^n a_{ij} x_j \leq b_i \quad i = 1, \dots, m \\ & x_j \geq 0 \quad j = 1, \dots, n \\ & x_j \text{ heltal} \quad j = 1, \dots, n \end{aligned}$$

$$\text{ofta} \quad x_j \leq u_j \quad j = 1, \dots, n$$

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

Obs: Tillåtna mängden ej konvex.

## 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  $M \gg x$ )

$$2x_1 + 3x_2 \leq 5 + M(1 - y_1),$$

$$5x_1 + 2x_2 \leq 6 + M(1 - y_2),$$

$$y_1 + y_2 = 1,$$

$$y_1 \in \{0, 1\}, y_2 \in \{0, 1\}.$$

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

## Antingen-eller-villkor: Exempel

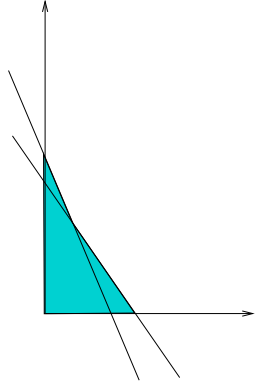
Antingen  $2x_1 + 3x_2 \leq 5$  eller  $5x_1 + 2x_2 \leq 6$ .

$M = 5$ :

$$2x_1 + 3x_2 \leq 5 + 5(1 - y_1),$$

$$5x_1 + 2x_2 \leq 6 + 5(1 - y_2),$$

$$y_1 + y_2 = 1, y_1 \in \{0, 1\}, y_2 \in \{0, 1\}.$$



## Speciell användning av heltalsvariabler

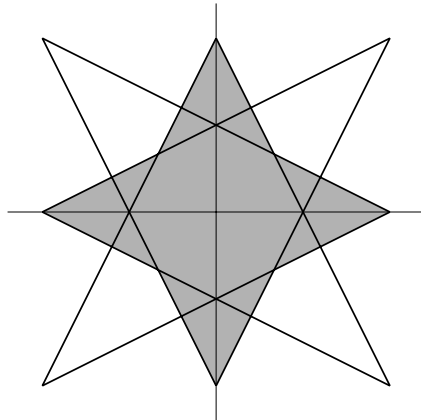
Icke-konvexa tillåtna områden, t ex villkorliga undre gränser:

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

Formulera som

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

## Speciell användning av heltalsvariabler



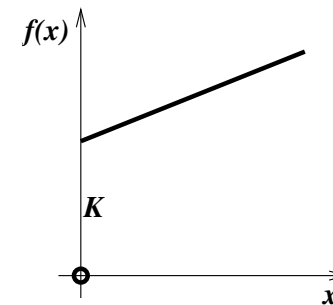
Ett icke-konvext område. (Unionen av två konvexa områden.)

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

## Speciell användning av heltalsvariabler

Fasta kostnader:

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



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

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

Förutsätter minimering.

## Kappsäcksproblemet:

$$\max z = \sum_{j=1}^n c_j x_j$$

$$\text{då} \quad \sum_{j=1}^n a_j x_j \leq b$$

$$0 \leq x_j \leq u_j, \text{ heltal } j = 1, \dots, n$$

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?

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

Åtgärd  $j$  kostar  $c_j$ . Minimera totalkostnaden.

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

$$\min z = \sum_{j=1}^n c_j x_j$$

$$\text{då} \quad \sum_{j=1}^n a_{ij} x_j \geq 1 \quad i = 1, \dots, m$$

$$x_j \in \{0, 1\} \quad j = 1, \dots, n$$

Uppdelningsproblemet (partitioneringsproblemet): 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

$$\min \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} + \sum_{i=1}^m f_i y_i$$

$$\text{då} \quad \sum_{j=1}^n x_{ij} \leq s_i y_i \quad \text{för alla } i \quad (1)$$

$$\sum_{i=1}^m x_{ij} = d_j \quad \text{för alla } j \quad (2)$$

$$x_{ij} - s_i y_i \leq 0 \quad \text{för alla } i, j \quad (3)$$

$$x_{ij} \geq 0 \quad \text{för alla } i, j \quad (4)$$

$$y_i \in \{0, 1\} \quad \text{för alla } i \quad (5)$$

Behövs bivillkor 3? (se lab 4)

# Nätverksdesignproblemet

Noder:  $\mathcal{N}$ . Möjliga länkar:  $\mathcal{A}$ .

Trafikbehov  $k$ : startnod  $o(k)$ , slutnod  $d(k)$ , behov  $r^k$ .

Kostnad för länk  $(i, j)$ : Fast kostnad:  $f_{ij}$ , rörlig kostnad:  $c_{ij}^k$ .

Maxkapacitet för länk  $(i, j)$ :  $u_{ij}$ .

Uppfyll trafikbehoven till minsta kostnad.

## Variabeldefinition:

$x_{ij}^k$  = flöde från  $o(k)$  till  $d(k)$  i länk  $(i, j)$ ,

$y_{ij} = \begin{cases} 1 & \text{om länk } (i, j) \text{ används} \\ 0 & \text{om inte.} \end{cases}$

# Nätverksdesignproblemet

$$\min \sum_{k \in \mathcal{C}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^k x_{ij}^k + \sum_{(i,j) \in \mathcal{A}} f_{ij} y_{ij}$$

$$\text{då } \sum_{j:(j,i) \in \mathcal{A}} x_{ji}^k - \sum_{j:(i,j) \in \mathcal{A}} x_{ij}^k = b_i^k \quad \text{för alla } i \in \mathcal{N}, k \in \mathcal{C} \quad (1)$$

$$\sum_{k \in \mathcal{C}} x_{ij}^k \leq u_{ij} y_{ij} \quad \text{för alla } (i, j) \in \mathcal{A} \quad (2)$$

$$x_{ij}^k \leq d_{ij}^k y_{ij} \quad \text{för alla } (i, j) \in \mathcal{A}, k \in \mathcal{C} \quad (3)$$

$$x_{ij}^k \geq 0 \quad \text{för alla } (i, j) \in \mathcal{A}, k \in \mathcal{C} \quad (4)$$

$$y_{ij} \in \{0, 1\} \quad \text{för alla } (i, j) \in \mathcal{A} \quad (5)$$

där  $d_{ij}^k = \min(r^k, u_{ij})$  och

$$b_i^k = \begin{cases} -r^k & \text{om } i = o(k) \\ r^k & \text{om } i = d(k) \\ 0 & \text{annars.} \end{cases}$$

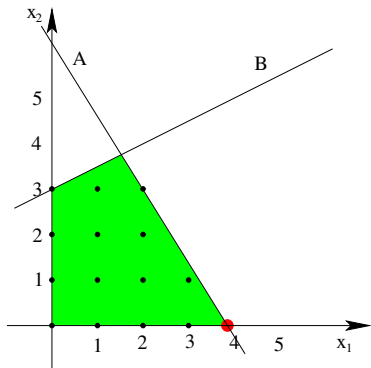
## Heltalsprogrammering: Exempel

$$\max z = 30x_1 + 18x_2$$

$$\text{då } \begin{matrix} 8x_1 & +5x_2 & \leq & 31 & (A) \\ -x_1 & +2x_2 & \leq & 6 & (B) \end{matrix}$$

$$x_1, x_2 \geq 0$$

$x_1, x_2$  heltal



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:  $z_{LP} \geq z^*$ .

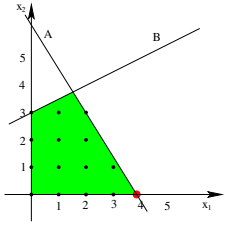
Alltså  $z^* \leq 116.25$

Alla koefficienter i målfunktionen är heltal. Alla variabler skall vara heltal.

$\Rightarrow z^*$  heltal. Avrunda  $z_{LP}$  neråt.

$\therefore z^* \leq 116$

## 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:  $z \leq z^*$ .

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

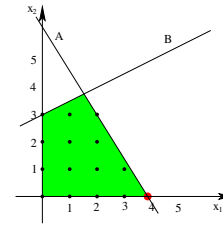
## Heltalsprogrammering: Vägval

Optimerande / heuristisk?

LP-baserad / kombinatorisk?

Trädsökning / plansnittning?

## 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.

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

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

## Heltalsprogrammering: Metodprinciper

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

*Fullständig uppräknning* av alla möjliga lösningar: **Nej!**

*Ofullständig uppräknning* kan vara bra. Kan dock aldrig garantera polynomisk lösningstid.

**Heuristik:** Snabbare. Inga garantier. Bevisar ej optimalitet.

Konstruktiva metoder.

Sökmeter.

Lagrangerelaxation.

### LP-baserad metod:

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

### Kombinatorisk metod:

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

## 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 trädets.

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

### Trädsökning:

- Relaxation.
- Förgrening: Dela upp problemet i flera (disjunkta) problem.
- Rekursivt.
- Kapa grenar.
- Avsöka hela trädets.
- 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: 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.**

Algoritm (max):

- Om inget oavsökt problem återstår: Stopp. Annars välj ett oavsökt problem.
- Lös LP-relaxationen (simplexmetoden). Om den saknar tillåten lösning: Kapa grenen. Gå till 1.
- Om  $z_{LP} \leq \underline{z}$ : Kapa grenen. Gå till 1.
- Om lösningen är heltalig: Spara lösningen och kapa grenen. Gå till 1.
- 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.

### Land-Doig-Dakins metod: Exempel

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

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.

Specificera:

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

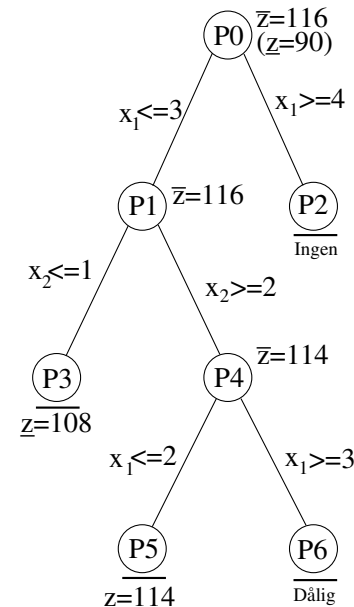
Exempel:

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



## Kranexemplet

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

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: 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.)

## 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  $x_j = 1$ .

## Bivillkorsfixering: Exempel

$$3x_1 + 4x_2 + 8x_3 - 5x_4 \leq 10 \quad (1)$$

$$5x_1 + 6x_2 + 5x_3 + 10x_4 \leq 14 \quad (2)$$

$$2x_1 - 4x_2 - 3x_3 + 3x_4 \leq -2 \quad (3)$$

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

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: Exempel: Sudoku

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

Bivillkor:

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

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

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

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

(siffran  $k$  en gång i mindre kvadrat  $(p, q)$ )

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

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

Ingen målfunktion.

## Bivillkorsfixering: Utvidgade tester för speciella strukturer

Utvidgade tester för Sudoku:

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

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

Eliminera siffran 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  $k$ .

Relaxation 1:

Billigaste uppspännande 1-träd.

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

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

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

## Kombinatoriskt baserad trädsökning för TSP

Förgrening 1:

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.

(Fungerar ej med tillordningsrelaxationen.)

Relaxation 2:

Tillordningsproblemet.

Motsvarar relaxation av subtursförbjudande/sammanhängande villkoren.

Alla noder får rätt valens. Lösningen ej säkert sammanhängande/kan innehålla mindre cykler.

Lös med ungerska algoritmen.

## Kombinatoriskt baserad trädsökning för TSP

Förgrening 2:

Syftar till att förbjuda för små cykler.

Förbjud en båge i cykeln.

Förgrening: För varje båge i cykeln: Skapa ett problem där bågen är förbjuden ( $x_{ij} = 0$ ).

Gren 1: Förbjud första bågen.

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

Gren 3: Förbjud tredje bågen och tvinga med första och andra bågarna.

Gren 4: ...

Många grenar.

(Kan ej användas för TSPr.)

## Kombinatoriskt baserad trädsökning för TSP

Förgrening 3:

Syftar till att göra lösningen sammanhängande.

Tvinga med en båge mellan separata delar.

Förgrening: För varje båge i snittet mellan två separata delar: Skapa ett problem där bågen är med ( $x_{ij} = 1$ ).

Minst en av dessa grenar måste innehålla optimum.

För att få optimum i exakt en, förbjud tidigare medtvingade bågar.

Gren 1: Tvinga med första bågen.

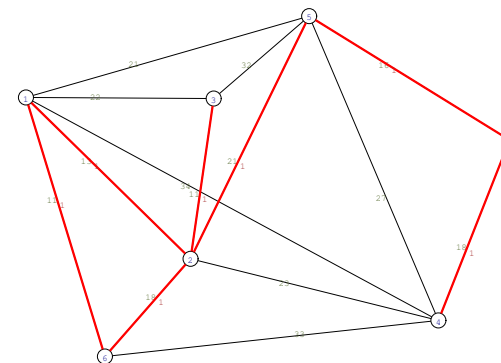
Gren 2: Tvinga med andra bågen och förbjud första bågen.

Gren 3: Tvinga med tredje bågen och förbjud första och andra bågarna.

Gren 4: ...

(Fungerar ej med 1-trädsrelaxationen.)

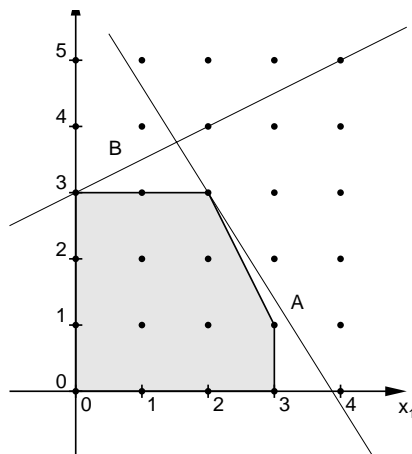
## Kombinatoriskt baserad trädsökning för TSP: Exempel



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.

Resultande 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  $i \in S$  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.)