Linköpings universitet
Matematiska institutionen
Optimeringslära

Optimeringslära grundkurs för Y
Laboration 2
14 februari 2021

Laboration 2 - Heltalsoptimering

1 Problemställning — Synande av cellprover

När ett biologiskt cellprov ska avsynas med hjälp av ett mikroskop ryms inte hela cellprovet inom mikroskopets synfält, utan detta täcker bara en liten del av hela cellprovet. Dessutom innehåller cellprovet stora delar av ointressant yta som inte behöver avsynas av ett mänskligt öga. Man kan dock genom att använda bildbehandling identifiera den del av provet som innehåller intressanta delar som vi skulle vilja avsyna visuellt. Det vi vill åstakomma är en tidseffektiv metod för att avsyna alla intressanta delar av cellprovet. Genom att dela upp tidseffektiviseringsaspekten i två delar kan vi angripa problemet. Den första delen är att vi vill täcka in hela det intressanta området med så få mikroskopbilder som möjligt (övertäckningsproblemet, se sektion 2). Den andra delen är att den tid (sträcka) som vi flyttar mikroskopets synfält mellan olika mikroskopbilder ska vara så kort som möjligt (handelsresandeproblemet, se sektion 3).

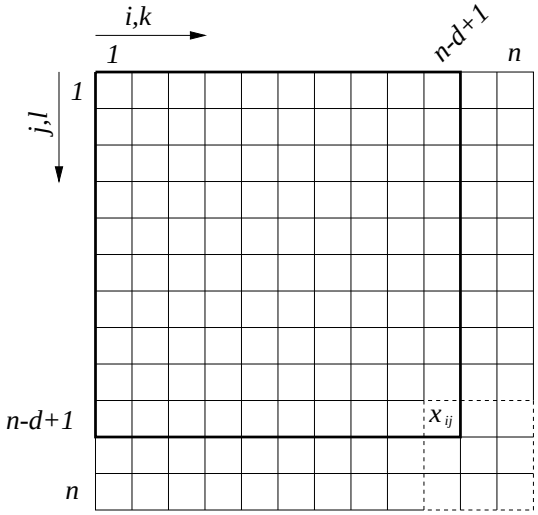
2 Övertäckningsproblemet

2.1 Modell

Cellprovets yta består av $n \times n$ pixlar. Koordinaterna som beskriver en pixels position ges av (k, l) där $k, l \in \mathcal{P} = \{1, \ldots, n\}$. Värdet b_{kl} anger om pixeln (k, l) ska avsynas ($b_{kl} = 1$) eller inte ($b_{kl} = 0$). I en mikroskopbild kan man se $d \times d$ pixlar. Låt koordinaten (i, j) definiera positionen på mikroskopbildens övre vänstra hörn; vi kan då använda den binära variabeln x_{ij} för att ange om mikroskopbilden används ($x_{ij} = 1$) eller ej ($x_{ij} = 0$). De koordinater för mikroskopbildpositionen som kan användas är $i, j \in \mathcal{B} = \{1, \ldots, n - d + 1\}$ (se exempel i figur 1). Om en bild med position (i, j) täcker en pixel med position (k, l) är $a_{ij}^{kl} = 1$. Alltså gäller:

$$a_{ij}^{kl} = \begin{cases} 1 & k - d < i \leq k,\ l - d < j \leq l \\ 0 & \text{annars} \end{cases}$$

då $k, l \in \mathcal{P}$ och $i, j \in \mathcal{B}$.



Figur 1: Hela bilden innehåller $n \times n$ pixlar ($n = 12$). En mikroskopbild ($x_{ij} = 1$), täcker $d \times d$ pixlar ($d = 3$), se det streckade området. Det är bara intressant att lägga ut mikroskopbilders övre vänstra hörn inom det området som är markerat med tjock linje.

Övertäckningsproblemet kan formuleras som:

$$\begin{aligned} \min z = \quad & \sum_{i\in\mathcal{B}}\sum_{j\in\mathcal{B}} x_{ij} \\ \text{då} \quad & \sum_{i\in\mathcal{B}}\sum_{j\in\mathcal{B}} a_{ij}^{kl} x_{ij} \quad \geq b_{kl}, \quad k,l \in \mathcal{P} \quad (1) \\ & x_{ij} \in \{0,1\}, \qquad\qquad\quad i,j \in \mathcal{B} \quad (2) \end{aligned}$$

Målfunktionen säger att vi vill minimera antalet mikroskopbilder som används. Bivillkorsmängden (1) är bara uppfyllt då vi har sett varje intressant pixel ($b_{kl} = 1$) minst en gång. Genom att använda definitionen för a_{ij}^{kl} och att om $b_{kl} = 0$ så är villkoret redundant, kan vi skriva om (1), vilket ger följande formulering:

$$\begin{aligned} \min\ z = \quad & \sum_{i\in\mathcal{B}}\sum_{j\in\mathcal{B}} x_{ij} \\ \text{då} \quad & \sum_{i\in\mathcal{B}:\ k-d<i\leq k}\ \sum_{j\in\mathcal{B}:\ l-d<j\leq l} x_{ij} \quad \geq 1, \quad k,l \in \mathcal{P} : b_{kl} = 1 \\ & x_{ij} \in \{0,1\}, \qquad\qquad\qquad\qquad\qquad i,j \in \mathcal{B} \end{aligned}$$

Förberedelseuppgift: Teckna villkoret för $k = 3$ och $l = 2$ *explicit* (utan att använda summatecken).

. . . . . .

2.2 Optimering

Ni ska nu lösa ett litet övertäckningsproblem av ovanstående typ med AMPL.

1. **Öppna ett terminalfönster och ge kommandot** `module add prog/ampl-demo/`
2. **Kopiera filer till Er hemkatalog med** `cp /courses/TAOP07/Lab2/* .` (Observera punkten på slutet.)
3. **Komplettera modellfilen** `setcover.mod`.
4. **Lös optimeringsproblemet**.

Till Er hjälp finns en datafil (`setcover1.dat`) som innehåller värdena på skalärerna n och d samt matrisen b. Det finns även en kommandofil (`setcover.run`) som löser Er modell (`setcover.mod`) och genererar en datafil (`stsp.dat`) till nästa delproblem (handelsresandeproblemet).

AMPL-tips: För att definiera bivillkor som finns bara för index k och l sådana att $b_{kl} = 1$ används en konstruktion av typen

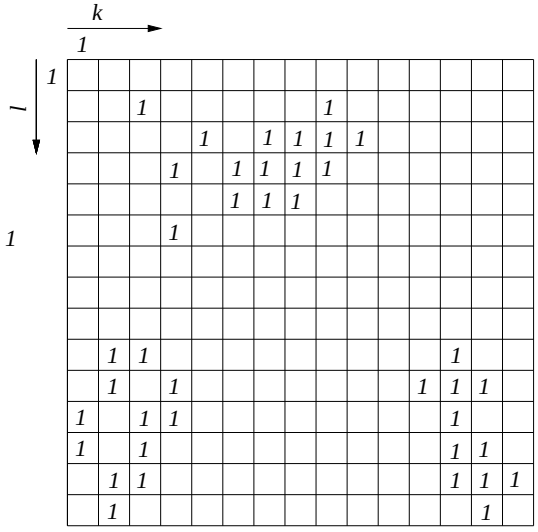
`subject to con {.....: b[k,l] == 1}: ...`

På liknande sätt kan man skriva en summa över index som ska uppfylla *två* krav med en konstruktion av typen

`sum {.....: ... && ...} ...`

Optimal heltalslösning: .....

Rita in den funna optimala övertäckningen i figur 2.



Figur 2: Pixlar som skall avsynas och en optimal övertäckning (ritas in).

[Anmärkning: Övertäckningsproblemet, som det är formulerat ovan, uppvisar ofta ett litet dual-gap, dvs skillnad i optimalt målfunktionsvärde mellan heltalsproblemet och dess LP-relaxation. Ibland kommer LP-optimum till och med att bli heltaligt och därför även vara ett heltalsoptimum (i vilket fall dual-gapet är noll).]

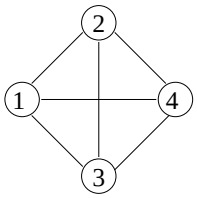
För att göra om övertäckningsproblemets lösning till ett handelsresandeproblem antar vi att det inte spelar någon roll vid vilka positioner mikroskopet startar respektive slutar. Detta kan vi åstakomma genom att utöka handelseresandeproblemet med en extra nod som har avståndet noll till alla andra noder.

3 Handelsresandeproblemet

3.1 Modell

I ett handelsresandeproblem vill man besöka ett antal *noder* (platser) $i \in \mathcal{N}$, i det här fallet är det ett antal mikroskoppositioner. Mellan varje par av noder, $i, j \in \mathcal{N} : i < j$, finns det en *båge* $e = (i, j)$. Vi definierar $\mathcal{E}$ som mängden av *oriktade* bågar. Avståndet c_e, $e \in \mathcal{E}$ mellan varje par av noder, är det euklidiska avståndet mellan mikroskopbilderna. Kostnaden c_e uppstår om vi väljer att ta med båge e i handelsresandeturen, dvs om $x_e = 1$.

För att förenkla hanteringen av bågarna i modellen behöver vi ett enkelt sätt att beskriva om en båge ansluter till en nod i. Vi definierar därför mängden $\delta(i)$ som de bågar vars ena ände ansluter till nod i (se exempel i figur 3).



Figur 3: Graf $G = (\mathcal{N}, \mathcal{E})$ där $\mathcal{N} = \{1, 2, 3, 4\}$, $\mathcal{E} = \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)\}$ och $\delta(2) = \{(1, 2), (2, 3), (2, 4)\}$.

Det är nu möjligt att formulera en första bivillkorsgrupp. Om en uppsättning bågar skall definiera en handelsresandetur så måste varje nod ha exakt två bågar anslutna till sig, dvs

$$\sum_{e \in \delta(i)} x_e = 2, \quad i \in \mathcal{N},$$

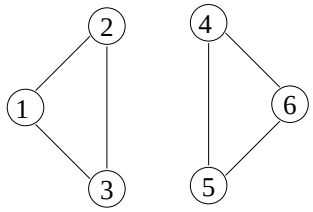
eftersom en båge behövs för att ta sig till noden och en för att ta sig därifrån. Detta är dock inget tillräckligt krav för en handelsresandetur.

Inom en delmängd, $\mathcal{S}$, av noderna ($\mathcal{S} \subseteq \mathcal{N}$) kan en subtur uppstå, dvs det kan finnas en tur som bara besöker noderna $i \in \mathcal{S}$. Tillhörande mängden $\mathcal{S}$ kan vi nu definiera den delmängd av bågar $\mathcal{E}(\mathcal{S})$ som består av de bågar $e = (i, j) \in \mathcal{E}$ vars anslutande

noder båda ligger i $\mathcal{S}$ ($i, j \in \mathcal{S}$). Vi kan observera att om det existerar en subtur så finns det åtminstone ytterligare en (se figur 4). Detta medför att det i så fall finns en subtur som inte innefattar nod 1. Det räcker därmed att förbjuda den subtur som inte innehåller nod 1, dvs $\mathcal{S} \subseteq \mathcal{N} \setminus \{1\}$. En subtur existerar om det inom en nodmängd finns lika många bågar, $\sum_{e \in \mathcal{E}(\mathcal{S})} x_e$, som det finns noder, $|\mathcal{S}|$. En subtur måste innehålla minst tre noder, $|\mathcal{S}| \geq 3$. Vi kan nu definiera de subtursförbjudande bivillkoren som:

$$\sum_{e \in \mathcal{E}(\mathcal{S})} x_e \leq |\mathcal{S}| - 1, \quad \mathcal{S} \subseteq \mathcal{N}', \quad |\mathcal{S}| \geq 3,$$

då $\mathcal{N}' = \mathcal{N} \setminus \{1\}$. Observera att antalet bivillkor av denna typ typiskt är mycket stort, eftersom antalet sätt att välja ut $\mathcal{S}$ växer snabbt med avseende på $|\mathcal{N}|$.



Figur 4: Den subtur som definieras av mängden $\mathcal{S} = \{4, 5, 6\}$.

Det är nu möjligt att göra en kompakt formulering av det symmetriska handelsresandeproblemet (symmetriskt därför att det kostar lika mycket att ta sig från nod i till nod j som från j till i).

$$(P) \begin{cases} \min z = & \sum_{e \in \mathcal{E}} c_e x_e & & \\ \quad \text{då} & \sum_{e \in \delta(i)} x_e = 2, & i \in \mathcal{N} & (1) \\ & \sum_{e \in \mathcal{E}(\mathcal{S})} x_e \leq |\mathcal{S}| - 1, & \mathcal{S} \subseteq \mathcal{N}', |\mathcal{S}| \geq 3 & (2) \\ & x_e \in \{0, 1\}, & e \in \mathcal{E} & (3) \end{cases}$$

Återstoden av laborationen handlar om hur detta problem kan lösas.

Förberedelseuppgift: Beräkna approximativt antalet bivillkor i (2) **för problem** (P) **då** $|\mathcal{N}| = 10$**.**

. . . . . .

Många heltalsprogrammeringsmetoder (tex trädsökning) bygger på att det finns en effektiv lösare för det LP-relaxerade problemet. Vi koncentrerar oss därför på att utveckla en effektiv lösare till problemet där heltalskravet (3) är relaxerat.

$$(P_L)\begin{cases} \min z_L = & \sum_{e\in\mathcal{E}} c_e x_e & & \\ \quad \text{då} & \sum_{e\in\delta(i)} x_e & = 2, & i \in \mathcal{N} \quad (1) \\ & \sum_{e\in\mathcal{E}(\mathcal{S})} x_e & \leq |\mathcal{S}| - 1, & \mathcal{S} \subseteq \mathcal{N}', |\mathcal{S}| \geq 3 \quad (2) \\ & x_e \in [0,1], & & e \in \mathcal{E} \quad (3) \end{cases}$$

Förberedelseuppgift: Vilken är relationen mellan z_L^* och z^*? .....

Varför? .....

3.2 Bivillkorsgenerering

Antalet bivillkor i (2) växer mycket snabbt (exponentiellt) med avseende på $|\mathcal{N}|$, och det är därför opraktiskt att direkt utnyttja (P_L). Genom att stryka (2) ur (P_L) erhålls en relaxering.

$$(P_R)\begin{cases} \min z_R = & \sum_{e\in\mathcal{E}} c_e x_e & & \\ \quad \text{då} & \sum_{e\in\delta(i)} x_e & = 2, & i \in \mathcal{N} \quad (1) \\ & x_e \in [0,1], & & e \in \mathcal{E} \quad (3) \end{cases}$$

När detta problem löses kommer optimallösningen troligen att bryta mot något av bivillkoren i (2). Genom att addera bivillkor från (2) som inte är uppfyllda till (P_R) erhålls ett nytt och mer begränsat optimeringsproblem. Genereringen av nya bivillkor kan sedan upprepas tills en tillåten lösning till (P_L) erhållits.

Antag att l bivillkor har genererats i något steg av denna process och låt dessa bivillkor definieras av mängderna $\mathcal{S}_i$, $i = 1, \ldots, l$. [Vilka alltså alla uppfyller att $\mathcal{S}_i \subseteq \mathcal{N}'$ och $|\mathcal{S}_i| \geq 3$.] Optimeringsproblemet i steg l kan då formuleras som följer.

$$(P_R^l)\begin{cases} \min z_R^l = & \sum_{e\in\mathcal{E}} c_e x_e & & \\ \quad \text{då} & \sum_{e\in\delta(i)} x_e & = 2, & i \in \mathcal{N} \quad (1) \\ & \sum_{e\in\mathcal{E}(\mathcal{S}_i)} x_e & \leq |\mathcal{S}_i| - 1, & i = 1, \ldots, l \quad (2') \\ & x_e \in [0,1], & & e \in \mathcal{E} \quad (3) \end{cases}$$

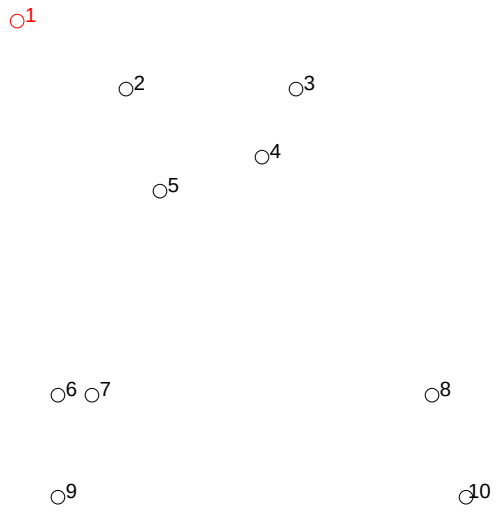
Förberedelseuppgift: Vilken är relationen mellan z_R^*, z_R^{l*}, z^* och z_L^*?

.....

Problemet (P_R^l) [eller (P_L)] har i allmänhet *inte* ett heltaligt LP-optimum, varför vi i processen ovan måste kunna generera bivillkor av typen (2) som inte är uppfyllda i en icke-heltalig punkt. För att så enkelt som möjligt illustrera principen för bivillkorsgenerering ska vi dock först studera ett exempel (`stsp.dat`) där optimallösningarna till (P_R^l) *råkar* bli heltaliga.

Gör nu följande steg för att lösa problemet (P_L) med bivillkorsgenerering.

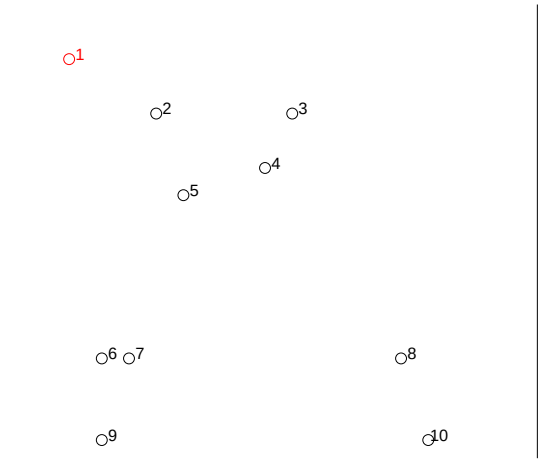
1. **I filen `stsp.mod` finns AMPL-modellen för (P_R^l), förutom målfunktionen. Komplettera filen med densamma.** [Från början finns inga bivillkor av typen $(2')$, dvs $l = 0$.]
2. **Lös problemet (`stsp.run`).**
3. **Notera målfunktionsvärdet och rita optimallösningen nedan.**

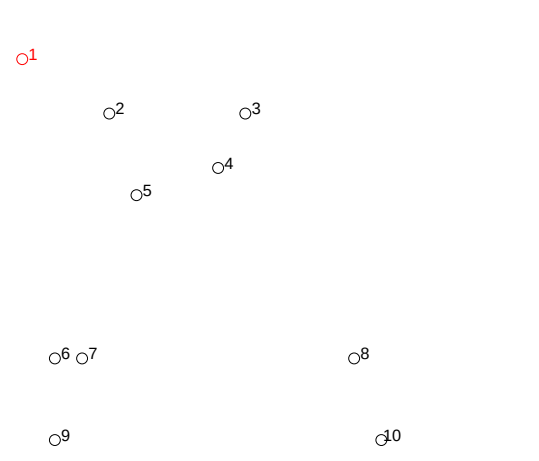


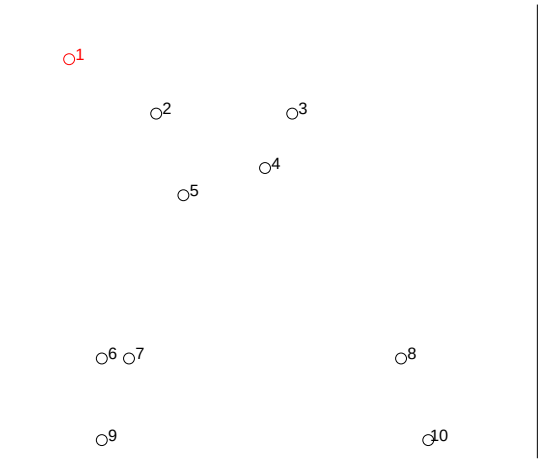
4. **Finn ett villkor av typen (2) som förbjuder en subtur som finns i optimallösningen. Inkludera villkoret i (P_R^l).**
5. **Lös om problemet. Notera målfunktionsvärdet och rita optimallösningen (i nästa figur).**
6. **Upprepa punkterna 4. och 5. tills dess att det inte finns en subtur.**

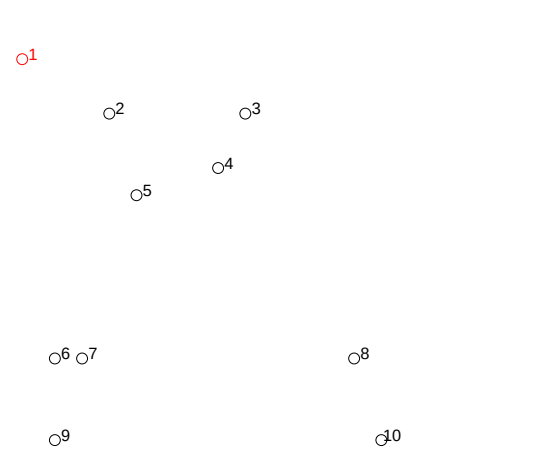
AMPL-tips: För att definiera ett villkor ur $(2')$, tex för $i = 1$ (dvs för S_1), så skapar man först i datafilen `stsp.dat` en mängd $S1$, tex som `S1 := 4 5 6;`. Sedan definieras $S1$ i `stsp.mod` (set $S1$;), varefter följande bivillkor kan bildas:

```
sum {i in S1, j in S1: (i,j) in EDGES} x[i,j] <= card(S1)-1;
```









Hur har målfunktionsvärdet förändrats mellan lösningarna? Förklara!

. . . . . .

. . . . . .

Att för hand undersöka vilka subtursförbjudande bivillkor som inte är uppfyllda i en optimallösning till (P_R^l) är inte rimligt för större problem, i synnerhet inte då optimallösningen inte är heltalig (vilket är normalfallet). I nästa avsnitt beskrivs en automatisering av bivillkorsgenereringsprocessen.

3.3 Separation

Givet en optimallösning till (P_R^l), säg $\bar{x}_e$, $e \in \mathcal{E}$, vill vi nu automatiskt undersöka om det finns något bivillkor av typen (2) som inte är uppfyllt. Detta är fallet om

$$\sum_{e \in \mathcal{E}(\mathcal{S})} \bar{x}_e > |\mathcal{S}| - 1 \Leftrightarrow \sum_{e \in \mathcal{E}(\mathcal{S})} \bar{x}_e - |\mathcal{S}| + 1 > 0,$$

för något $\mathcal{S} \subseteq \mathcal{N}', |\mathcal{S}| \geq 3$. Om inte alla bivillkor i (2) är uppfyllda, så kan lösningen till (P_R^l) skäras bort genom att något av de icke uppfyllda bivillkoren genereras. Vi kommer att välja bivillkor genom att lösa ett så kallat separationsproblem, vilket ger ett bivillkor som är mest överskridet. (Notera att eftersom varje villkor i (2) är förknippat med en mängd $\mathcal{S}$, så svarar detta mot att finna en speciell mängd $\mathcal{S}$.)

Separationsproblemet är ett optimeringsproblem i vilket beslutsvariablerna beskriver vilka noder som ska ingå i mängden $\mathcal{S}$; vi låter $z_i = 1$ om nod i ska tas med i $\mathcal{S}$ och $z_i = 0$ annars. Antalet noder i mängden, $|\mathcal{S}|$, är då $\sum_{i \in \mathcal{N}'} z_i$. Enligt definitionen av $\mathcal{E}(\mathcal{S})$ ska mängden bara innehålla de bågar, $e = (i, j)$, som har båda ändnoderna i $\mathcal{S}$, dvs om $z_i z_j = 1$. Eftersom vi vet att $z_1 = 0$ ska vi bara söka bland bågarna i mängden $\mathcal{E}' = \mathcal{E} \setminus \{\delta(1)\}$.

Hur ska kravet $|\mathcal{S}| \geq 3$ uppfyllas? Vi vill finna ett villkor (en subtur) som inte inkluderar nod 1, alltså ska det istället innehåller någon nod $k \in \mathcal{N}'$. Men vi vet inte vilken nod det är. Resultatet blir att vi måste lösa ett subproblem för varje nod k där vi fixerar $z_k = 1$. Vi kan se att $\sum_{e \in \mathcal{E}(\mathcal{S})} \bar{x}_e > |\mathcal{S}| - 1$ alltid är uppfyllt då $|\mathcal{S}| = 1$ och då $|\mathcal{S}| = 2$, därför kommer vi att kunna identifiera det bivillkor som inte är uppfyllt genom att lösa problemet då $|\mathcal{S}| \geq 1$ istället för då $|\mathcal{S}| \geq 3$. Att $|\mathcal{S}| \geq 1$ gäller vet vi då vi valt $z_k = 1$. Optimeringsproblemet för att finna det bivillkor som är mest otillåtet kan formuleras enligt följande:

$$\begin{aligned} \max z_s^k = \quad & 1 + \sum_{e=(i,j) \in \mathcal{E}' : i<j} \bar{x}_e z_i z_j - \sum_{i \in \mathcal{N}'} z_i \\ \text{då} \quad & z_i \in \{0,1\}, \qquad i \in \mathcal{N}' \\ & z_k = 1. \end{aligned}$$

Målfunktionen är som synes icke-linjär, men genom en omskrivning av produkten $z_i z_j$ kan denna komplikation kringgås. Vi inför nya variabler $w_e = z_i z_j \in [0,1]$ och nya bivillkor $w_e \leq z_i$, $w_e \leq z_j$ och $w_e \geq z_i + z_j - 1$ för $e = (i,j) : i < j$. När $z_i = 0$ eller $z_j = 0$ kommer $w_e = 0$, däremot kommer w_e bli 1 när $z_i = z_j = 1$, eftersom det tredje bivillkoret då kommer att tvinga w_e till 1. Vi kan alltså omformulera separationsproblemet (för fixt värde på k) till

$$(P_s^k) \begin{cases} \max z_s^k = & 1 + \sum_{e \in \mathcal{E}'} \bar{x}_e w_e - \sum_{i \in \mathcal{N}'} z_i & \\ \text{då} \quad & w_e \leq z_i, & e = (i,j) \in \mathcal{E}' \\ & w_e \leq z_j, & e = (i,j) \in \mathcal{E}' \\ & w_e \geq z_i + z_j - 1, & e = (i,j) \in \mathcal{E}' \\ & w_e \in [0,1], & e \in \mathcal{E}' \\ & z_i \in \{0,1\}, & i \in \mathcal{N}' \\ & z_k = 1. & \end{cases}$$

Man kan visa att detta är ett nätverksproblem med en fullständigt unimodulär bivillkorsmatris (och heltaliga högerled), varför det har heltalsegenskap. Det räcker alltså att lösa det LP-relaxerade problemet, eftersom lösningen ändå blir heltalig. Med speciell nätverkskod kan detta optimeringsproblem lösas mycket effektivt; i laborationen kommer vi dock bara lösa det som ett vanligt LP-problem.

Förberedelseuppgift: Skriv ned AMPL-formuleringen av (P_s^k). Fortsätt på den redan befintliga delen av modellfilen.

```
problem stspvalid;

set NODESP := {i in NODES: i != 1}; # Noder utom 1:an
set EDGESP within (NODESP cross NODESP) :=
  { (i,j) in EDGES: i != 1 && j != 1}; # Bågar utan de som ansluter till nod 1

param cx {EDGES}; # Målfunktionskoefficienter för w

param k; # Fixerad nod

var w {(i,j) in EDGESP} >= 0, <=1; # Beslutsvariabel
var z {i in NODESP} >= 0, <=1; # Beslutsvariabel


# Låt målfunktionsvärdet i separationsproblemet heta Slack
```

stspvalid.mod

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

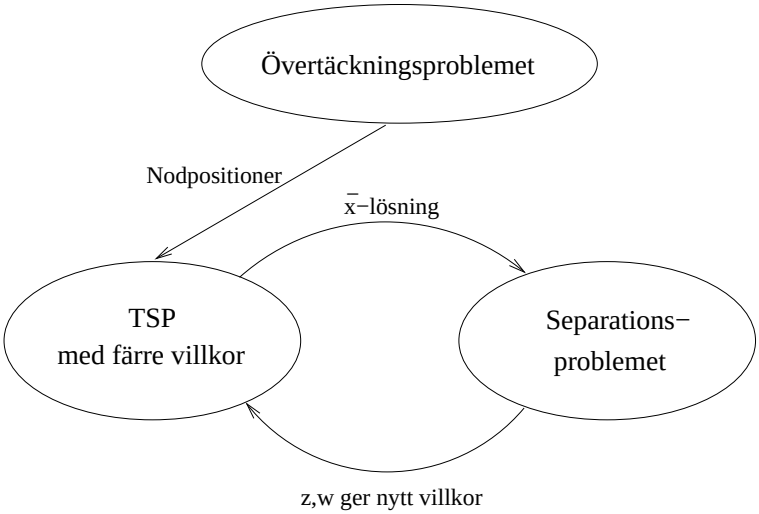
. . . . . .

. . . . . .

Vi ska tillämpa den automatiska bivillkorsgenereringen på ett handelsresandeproblem för vilket de relaxerade problemen (P_R^l) inte får heltaliga optima (dvs på det fall som är typiskt). Det **nya handelsresandeproblemet** skapas genom att **lösa ett nytt övertäckningsproblem**, denna gång med datafilen `setcover2.dat`. (Ändras i `setcover.run`.) Detta ger en ny datafil till handelsresandeproblemet (`stsp.dat`). [Om Ni vill spara den gamla datafilen `stsp.dat`: kopiera den till en annan fil!] **Komplettera filen `stspvalid.mod` med modellen för separationsproblemet.**

Gör följande steg för att lösa problemet (P_L) med hjälp av bivillkorsgenerering baserad på separationsproblemet. Använd här kommandofilen `stsp2.run`. [Med denna fil löses automatiskt *både* det relaxerade problemet (P_R^l) *och* separationsproblemen (P_s^k) för $k = 2, 3, 4, \ldots$.]

1. **Lös problemet** (P_R^l) **med** $l = 0$ **[inga bivillkor i** $(2')$**].** Optimallösningen $\bar{x}_e$, $e \in \mathcal{E}'$, blir målfunktionskoefficienter i separationsproblemen (P_s^k).
2. **Lös separationsproblemen** (P_s^k). Detta görs för $k = 2, 3, 4, \ldots$, tills ett subtursförbjudande villkor som inte är uppfyllt har hittats. Värdena på variablerna z_i, $i \in \mathcal{N}'$, beskriver då mängden $\mathcal{S}$ för detta villkor.
3. **Lägg bivillkoret som genererats till** (P_R^l)**.**
4. **Lös om problemet** (P_R^l).
5. **Upprepa punkterna 2. till 4. tills alla bivillkor i** (2) **är uppfyllda**.



Figur 5: De olika problemen och hur informationen mellan den skickas.

Målfunktionsvärden $(z_R^{l\,*})$ **och genererade bivillkor (mängder** $\mathcal{S}$**):**

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Optimalt målfunktionsvärde (z_L^*)**:** . . . . . .

Vi har nu löst LP-relaxationen (P_L), och dess optimallösning är som synes *inte* heltalig. För att lösa handelsresandeproblemet (P) kan man då fortsätta genom att tillämpa trädsökning. Denna tillgår som vanligt, *förutom* att det kan vara nödvändigt att generera fler subtursförbjudande bivillkor vid lösandet av LP-relaxationer i trädets grenar. [Anledningen är att de hitintills genererade villkoren behövdes för att definiera optimum till LP-relaxationen av handelsresandeproblemet, men efter att artificiella förgreningsvillkor införts så löser vi LP-relaxationer av handelsresandeproblem som modifierats genom att vissa variabler fixeras till noll eller ett.]

Förgrena över en fraktionell variabel (båge), förslagsvis över x_{68}. (Förgreningen görs enklast genom att i modellfilen lägga till villkoret $x_{68} = 1$ respektive $x_{68} = 0$.) Lös LP-relaxationerna i de två grenarna. Rita trädet nedan och markera vid noderna vilka uppskattningar (övre/undre) av z^* som fås.

Vilken blir slutsatsen angående z^*? Motivera!

.....

.....

Generera i de två grenarna, om så är möjligt, fler subtursförbjudande villkor. Vilken blir nu slutsatsen angående z^*? Motivera!

.....

.....

4 Appendix

Filerna nedan och två datafiler kan kopieras från kursbiblioteket med hjälp av:

```
cp /courses/TAOP07/Lab2/* .
```

Modellfiler

setcover.mod

```
problem setcover;

param n; # Storleken på cellprovet
param d; # Storleken på mikroskopbilden

set B := 1..n-d+1;
set P := 1..n;

param b{k in P,l in P}; # De pixlar som ska övertäckas

var x{i in B, j in B} binary; # Anger om bildposition används

# målfunktionsvärdet ska döpas till antal_mikroskopbilder
```

setcover.mod

stsp.mod

```
problem stsp;

#set S1; # Mängden som beskriver en subtur, definieras i stsp.dat
set NODES; # Antalet noder i handelsresandeproblemet (tsp)
set EDGES within (NODES cross NODES); # Mängden möjliga bågar

param c {EDGES}; # Bågkostnad (avstånd)
param Node_Pos{NODES, 1..2}; # Positioner för noder

var x {(i,j) in EDGES} >= 0, <=1; # Beslutsvariabler

minimize totallength: # Lägg till målfunktion här.

subject to valensvillkor {k in NODES}:
sum {(i,j) in EDGES: i==k || j==k} x[i,j] = 2;

# Modell för hur nya subtursförbjudande bivillkor kan adderas
#subject to subtur1:
#sum { i in S1, j in S1 : (i,j) in EDGES } x[i,j] <=  card(S1) - 1;
```

stsp.mod

stspvalid.mod

```
problem stspvalid;

set NODESP := {i in NODES: i != 1}; # Noder utom 1:an
set EDGESP within (NODESP cross NODESP) :=
  { (i,j) in EDGES: i != 1 && j != 1}; # Bågar utan de som ansluter till nod 1

param cx {EDGES}; # Målfunktionskoefficienter för w

param k; # Fixerad nod

var w {(i,j) in EDGESP} >= 0, <=1; # Beslutsvariabel
var z {i in NODESP} >= 0, <=1; # Beslutsvariabel

# Låt målfunktionsvärdet i separationsproblemet heta Slack
```

stspvalid.mod

Kommandofiler

setcover.run

```
reset;
option solver cplex;

model setcover.mod;
data setcover1.dat;

solve setcover;

# OBSERVERA!!!!!!!
# Antal_mikroskopbilder måste ges målfunktionsvärdet och
# variablerna måste döpas till x för att datafilen till
# handelsresandeproblemet ska kunna genereras.

display antal_mikroskopbilder;
display x;

# Ni behöver inte göra några förändringar nedan i programmet.
# Beräkna parametrar för handelsresandeproblemet
# och skriver ned datafilen stsp.dat.

param i_node;
param N_Nodes;
let N_Nodes := antal_mikroskopbilder + 1;
param Node_Pos {1..N_Nodes, 1..2};
set NODES := 1..N_Nodes;
set EDGES within (NODES cross NODES);
param c {EDGES} >= 0;

let Node_Pos[1,1] := -1;
let Node_Pos[1,2] := -1;
```

```
let i_node := 2;
for {i in 1..n-d+1, j in 1..n-d+1} {
  if (x[i,j] = 1) then {
    let Node_Pos[i_node,1] := i;
    let Node_Pos[i_node,2] := j;
    let i_node := i_node+1;
  }
}
#display Node_Pos;

printf "set NODES :=" > stsp.dat;
for { i in NODES } printf " %d", i > stsp.dat;
printf ";\n" > stsp.dat
printf "param: EDGES: c :=\n" > stsp.dat;

param dx;
param dy;

for { i in NODES } {
  if (i = 1) then {
    for {j in NODES}
     if (j != 1) then printf "       1 %d   0\n", j > stsp.dat ;
  }
  else {
    for {j in NODES} {
      if (j > i) then {
        let dx := Node_Pos[i,1] - Node_Pos[j,1];
        let dy := Node_Pos[i,2] - Node_Pos[j,2];
        printf "       %d %d   %f\n", i, j, sqrt(dx*dx + dy*dy) > stsp.dat ;
      }
    }
  }
}
printf ";\n" > stsp.dat;
printf "param " > stsp.dat;
display Node_Pos > stsp.dat;
close stsp.dat;
```

setcover.run

stsp.run

```
reset;

option solver cplex;

model stsp.mod;
data  stsp.dat;

solve stsp;

display x;
```

stsp.run

stsp2.run

```
reset;
option solver cplex;

model stsp.mod;
data  stsp.dat;
model stspvalid.mod;

solve stsp;

display x;

# Döp variabeln till x, målfunktionskoefficienterna
# till separationsproblemet blir då cx

let {(i,j) in EDGES} cx[i,j] := x[i,j];
for {n in NODESP} {
  let k := n;

  # Låt målfunktionsvärdet i separationsproblemet vara Slack
  solve stspvalid;

  if (Slack > 0) then {
    display z;
    break;  # Bara nödvändigt att finna första bivillkoret
  }
}
include drawtour; #program som ritar ut turen på skärmen
```

stsp2.run