

## Programmazione Lineare Intera(PLI)

La Programmazione Lineare Intera (PLI) è la branca della ricerca operativa che si occupa di risolvere problemi lineari in cui le variabili decisionali possono assumere solo valori interi o booleani.

Non esistono algoritmi che trovano direttamente l'ottimo di un problema di PLI non banale. Ogni problema di PLI inizia con la sua valutazione inferiore e superiore.

$$v_i \leq v_{PLI} \leq v_s : v_i, v_{PLI}, v_s \in \mathbb{Z}$$

Poi ci saranno degli algoritmi che ci permettono di ridurre l'intervallo  $[v_i, v_s]$ . Si arriva all'ottimo quando  $v_i = v_s$  oppure ci possiamo fermare dopo un certo numero di passi quando siamo soddisfatti della stima ottenuta.

### 1. Problema dello zaino

In inglese "knapsack problem", anche conosciuto come "problema di caricamento". Ho uno zaino di capienza massima 30, ho degli oggetti con un peso e un valore:

Peso	$p_1 = 6$	$p_2 = 8$	$p_3 = 9$	$p_4 = 11$	$p_5 = 15$	$p_6 = 18$
Valore	$v_1 = 2$	$v_2 = 4$	$v_3 = 7$	$v_4 = 9$	$v_5 = 13$	$v_6 = 16$

$$\begin{cases} \max 2x_1 + 4x_2 + 7x_3 + 9x_4 + 13x_5 + 16x_6 \\ 6x_1 + 8x_2 + 9x_3 + 11x_4 + 15x_5 + 18x_6 \leq 30 \\ x_i \in \{0, 1\} \end{cases}$$

Modello.

Problema dello zaino:

$$\text{Booleano: } \begin{cases} \max c^T x \\ p^T x \leq P \\ x \in \{0, 1\}^n \end{cases} \quad \text{Intero: } \begin{cases} \max c^T x \\ p^T x \leq P \\ x \in \mathbb{Z}_n^+ \end{cases}$$

### 1.1. Valutazione superiore dello zaino intero

Con riferimento all'esempio precedente, costruisco il problema  $(p)$  ottenuto eliminando il vincolo di interezza e aggiungendo il vincolo  $x_i \geq 0 \forall x_i$ .  $(p)$  si chiama **rilassato continuo** del problema dello zaino intero.

$$(p) = \begin{cases} \max 2x_1 + 4x_2 + 7x_3 + 9x_4 + 13x_5 + 16x_6 \\ 6x_1 + 8x_2 + 9x_3 + 11x_4 + 15x_5 + 18x_6 \leq 30 \\ x_i \geq 0 \forall x_i \end{cases}$$

Costruisco il suo duale:

$$(d) = \begin{cases} \min 30y_1 + 0y_2 + \dots + 0y_7 \\ 6y_1 - y_2 = 2 \quad \text{Slack} \\ 8y_1 - y_3 = 4 \\ 9y_1 - y_4 = 7 \\ 11y_1 - y_5 = 9 \\ 15y_1 - y_6 = 13 \\ 18y_1 - y_7 = 16 \\ y_i \geq 0 \forall y_i \end{cases} \equiv \begin{cases} \min 30y_1 \\ 6y_1 \geq 2 \\ 8y_1 \geq 4 \\ 9y_1 \geq 7 \\ 11y_1 \geq 9 \\ 15y_1 \geq 13 \\ 18y_1 \geq 16 \\ y_i \geq 0 \forall y_i \end{cases}$$

Il problema è diventato banale e la soluzione ottima è  $\bar{y} = \frac{8}{9}$ . Il valore ottimo è  $v(d) = 30 \cdot \frac{8}{9}$

Se trovo una  $\bar{x}$  tale che  $v(p) = v(d)$  sono arrivato all'ottimo.

Provo a saturare lo zaino con  $x_6$ :

$$(p) = \begin{cases} \max 2x_1 + 4x_2 + 7x_3 + 9x_4 + 13x_5 + 16x_6 \\ 18x_6 \leq 30 \\ x_i \geq 0 \end{cases} \implies \bar{x} = \left(0, 0, 0, 0, 0, \frac{30}{18}\right)$$

$$v(p) = \frac{30}{18} \cdot 16 = 30 \cdot \frac{8}{9} = v(d) \text{ quindi sono arrivato all'ottimo!}$$

Trovare la soluzione ottima del rilassato continuo del caricamento intero è triviale.

$v(p) = 26, \bar{6}$ , ho trovato un limite superiore al problema di caricamento.

$v_{PLI} \leq 26$ , ho troncato all'unità visto che il problema ammette solo soluzioni intere.

Per trovare una stima inferiore prendo  $\bar{x} = \left(0, 0, 0, 0, 0, \left\lfloor \frac{30}{18} \right\rfloor \right) = (0, 0, 0, 0, 0, 1)$ , che è sicuramente una soluzione ammissibile.

$$\therefore v_i = 16 \leq v_{PLI} \leq 26 = v_s$$

$$Err = \frac{v_s - v_i}{v_i} = \frac{10}{16} \implies \sim 60\% \text{ di errore (schifo).}$$

### Come trovare la valutazione superiore?

Considero  $r_i := \frac{v_i}{p_i}$  i rendimenti. La soluzione ottima del rilassato continuo  $\bar{x}_{RC}$  si trova saturando lo zaino con il bene di massimo rendimento.

La valutazione superiore è la parte intera del valore ottimo del rilassato continuo:  $v_s = \lfloor c^T \bar{x}_{RC} \rfloor$

### 1.2. Valutazione superiore dello zaino booleano

Il rilassato continuo dello zaino booleano è  $(p) = \begin{cases} \max v^T x \\ p^T x \leq P \end{cases}$  (elimino anche il vincolo  $x \geq 0$ )

Considero  $r_i := \frac{v_i}{p_i}$  i rendimenti. La soluzione ottima del rilassato continuo  $\bar{x}_{RC}$  si trova caricando

per primo il bene di massimo rendimento, per secondo il secondo bene di massimo rendimento, e così via. Al primo bene che non ci sta, si satura lo zaino con una frazione di quel bene.

### 1.3. Valutazioni inferiori e superiori dello zaino intero e booleano

Il passo base di ogni problema di PLI è la valutazione dei limiti inferiori e superiori. Ricapitoliamo i quattro algoritmi che ci permettono di risolvere il problema del caricamento:

Sia  $c = (10, 17, 22, 21)$ ,  $p = (4, 5, 6, 2)$ ,  $P = 7$ , i rendimenti  $r = (2.5, 3.4, 3.7, 10.5)$

Problema	Valutazione inferiore	Valutazione superiore
<b>Zaino booleano:</b> $\max_{38 \leq v_{PLI} \leq 39}$ $err = \frac{1}{38} = 2.6\%$	<b>Algoritmo greedy:</b> carico i beni di massimo rendimento finché non ci stanno più. $\bar{x}_A = (0, 1, 0, 1)$ $v_i = c^T \bar{x}_A = 38$	<b>Rilassato continuo:</b> carico i beni di massimo rendimento finché non ci stanno più; sature lo zaino con una frazione dell'elemento che non ci sta. $\bar{x}_{RC} = \left(0, 0, \frac{5}{6}, 1\right)$ $v_{RC} = c^T \bar{x}_{RC} = 39.\bar{3}$ , $v_s = \lfloor v_{RC} \rfloor = \lfloor 39.\bar{3} \rfloor = 39$
<b>Zaino intero:</b> $\max_{63 \leq v_{PLI} \leq 73}$ $err = \frac{10}{63} = 16\%$	<b>Algoritmo greedy:</b> aggiungo sempre il bene di massimo rendimento finché non ci sta più. $\bar{x}_A = (0, 0, 0, 3)$ $\bar{x}_A$ si arrotonda per difetto. $v_i = c^T \bar{x}_A = 63$	<b>Rilassato continuo:</b> sature lo zaino con il bene di massimo rendimento. $\bar{x}_{RC} = \left(0, 0, 0, \frac{7}{2}\right)$ $v_{RC} = c^T \bar{x}_{RC} = 73.5$ , $v_s = \lfloor v_{RC} \rfloor = \lfloor 73.5 \rfloor = 73$

Un **algoritmo greedy** è facile, veloce, buono (statisticamente vicino al valore ottimo).

$$\text{Produzione intera: } \begin{cases} \max c^T x & \max \\ Ax \leq b & : v_i \leq v_{PLI} < v_s, \\ x \in \mathbb{Z}_n^+ & \end{cases} \quad \begin{cases} \min c^T x & \min \\ Ax \geq b & : v_i \leq v_{PLI} < v_s \\ x \in \mathbb{Z}_n^+ & \end{cases}$$

**Attenzione:** l'arrotondamento non è sempre l'algoritmo corretto per trovare una soluzione ammissibile. Non funziona per i problemi di produzione intera o booleana con più di un vincolo. Il problema dello zaino è un problema di produzione, ma il contrario non è vero in generale. Il metodo per trovare una soluzione ammissibile cambia a seconda del problema.

## 1.4. Problemi particolari

### 1.4.1. Problema dei costruttori

Su Clash of Clans gli aggiornamenti degli edifici avvengono in un certo numero di giorni. Sia  $c = (6, 8, 9, 11, 13, 16, 18)$ , il numero di giorni che impiega ciascun edificio.  $\sum c_i = 81$

Ho  $k$  costruttori: come assegno i lavori per impiegare il minor tempo possibile? Questo può sembrare un problema di assegnamento, ma in realtà si tratta di uno zaino binario.

$$\begin{cases} \max 6x_1 + 8x_2 + 9x_3 + 11x_4 + 13x_5 + 16x_6 + 18x_7 \\ 6x_1 + 8x_2 + 9x_3 + 11x_4 + 13x_5 + 16x_6 + 18x_7 \leq \frac{81}{k} \text{ (per semplicità, } 81\%k = 0) \\ x \in \{0, 1\}^7 \end{cases}$$

### 1.4.2. Problema del multizaino

Devo tenere in considerazione, oltre al peso degli oggetti, anche il loro volume.

$$\begin{cases} \max c^T x \\ p^T x \leq P \\ v^T x \leq V \\ x \in \{0, 1\} \end{cases}$$

- La  $v_s$  si trova, come al solito, con il rilassato continuo.
- La  $v_i$  è molto difficile da trovare.

Questo problema è NP-completo e, addirittura, la variante booleana non ammette un algoritmo polinomiale che fornisce soluzioni approssimate.

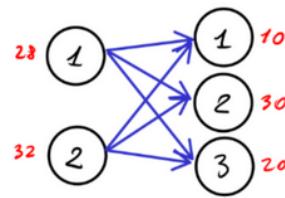
## 2. Problema di trasporto (e assegnamento intero)

Supponiamo che l'ottimo del rilassato continuo sia a componenti intere:  $\bar{x}_{RC} \in \mathbb{Z}_n^+$ .

Allora  $\bar{x}_{RC}$  è l'ottimo del problema di PLI poiché  $c^T \bar{x}_{RC} \leq v_{PLI} \leq c^T \bar{x}_{RC}$ .

Prendo un esempio di poliedro di un problema di trasporto:

$$\begin{cases} x_{11} + x_{12} + x_{13} = 28 \\ x_{21} + x_{22} + x_{23} = 32 \end{cases} \text{ (produzione)} \\ \begin{cases} x_{11} + x_{21} = 10 \\ x_{12} + x_{22} = 30 \\ x_{13} + x_{23} = 20 \end{cases} \text{ (destinazione)} \\ x_{ij} \geq 0$$



Sia  $T := \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$  la matrice di trasporto.

$\text{rank}(T) = 4$ , si può verificare vedendo che il determinante di tutti i minori  $5 \times 5$  (sono 6) fa zero, e trovando un minore  $4 \times 4$  che abbia determinante diverso da zero. Di conseguenza, un'equazione dei vincoli è eliminabile.

### TEOREMA.

**Teorema (unimodularità della matrice di trasporto):**

Sia  $T$  una matrice di trasporto. Tutti i minori di  $T$  di rango massimo invertibili hanno determinante  $+1$  o  $-1$ .

Nell'esempio, tutti i minori  $4 \times 4$  di  $T$  invertibili sono unimodulari.

**Osservazione:** le sottomatrici di rango massimo invertibili nella PL sono le matrici di base. Quindi tutte le matrici di base del trasporto sono unimodulari.

Allora la soluzione di base  $x = A_B^{-1}b_B$  è a componenti intere (si può verificare risolvendo con il metodo di Cramer, per il quale devo dividere per il determinante  $+1$  o  $-1$ ). Se tutte le soluzioni di base del trasporto sono a componenti intere, allora tutti i vertici sono a componenti intere.

### Proposizione.

L'ottimo del trasporto rilassato continuo è a componenti intere, quindi coincide con l'ottimo del trasporto intero.

**L'assegnamento è un caso particolare di trasporto**, quindi anch'esso gode della proprietà di interezza. Che l'assegnamento sia cooperativo o no, può essere risolto con linprog!

### 3. Problema del bin packing

Il problema del bin packing è un problema di ottimizzazione, in cui oggetti di diversi pesi devono essere inseriti in un numero finito di contenitori, ciascuno con una capacità, in modo da minimizzare il numero di contenitori utilizzati. In questo esempio ci sono solo contenitori con capacità 10.

Capacità: 10	$p_1 = 3$	$p_2 = 6$	$p_3 = 5$	$p_4 = 4$	$p_5 = 4$	$p_6 = 8$
--------------	-----------	-----------	-----------	-----------	-----------	-----------

$$x_{ij} := \begin{cases} 1, & \text{il contenitore } i\text{-esimo contiene l'oggetto } j\text{-esimo} \\ 0, & \text{altrimenti} \end{cases}$$

Si trova facilmente una soluzione ammissibile  $v_s = 4$  prendendo i primi 2 elementi e mettendoli nel contenitore; gli ultimi 2 non ci stanno e quindi vanno in 2 contenitori diversi. Per questo so con certezza che il numero di contenitori è (al massimo) 4.

$$\left\{ \begin{array}{l} \min y_1 + y_2 + y_3 + y_4 \\ \begin{array}{l} x_{11} + x_{21} + x_{31} + x_{41} = 1 \\ x_{12} + x_{22} + x_{32} + x_{42} = 1 \\ x_{13} + x_{23} + x_{33} + x_{43} = 1 \\ \dots \\ x_{16} + x_{26} + x_{36} + x_{46} = 1 \end{array} \\ \begin{array}{l} 3x_{11} + 6x_{12} + 5x_{13} + 4x_{14} + 4x_{15} + 8x_{16} \leq 10y_1 \\ 3x_{21} + 6x_{22} + 5x_{23} + 4x_{24} + 4x_{25} + 8x_{26} \leq 10y_2 \\ \dots \\ 3x_{31} + 6x_{32} + 5x_{33} + 4x_{34} + 4x_{35} + 8x_{36} \leq 10y_3 \\ 3x_{41} + 6x_{42} + 5x_{43} + 4x_{44} + 4x_{45} + 8x_{46} \leq 10y_4 \end{array} \\ x \in \{0, 1\}, y \in \{0, 1\} \end{array} \right. \begin{array}{l} \text{vincoli sugli oggetti} \\ \text{vincoli sui contenitori} \end{array}$$

$$\text{Ho introdotto } y_i := \begin{cases} 1, & \text{il contenitore } i\text{-esimo viene usato} \\ 0, & \text{altrimenti} \end{cases}$$

$10y_1, \dots, 10y_4$  serve per evitare che l'ottimizzatore prenda come soluzione ammissibile  $y = 0$ .

#### 3.1. Valutazione inferiore

La valutazione inferiore si trova con il rilassato continuo:  $v_i = \left\lceil \frac{\sum_{j=1}^n p_j}{P} \right\rceil = 3$

### 3.2. Valutazione superiore

La valutazione superiore  $v_s$  si trova ordinando gli oggetti per peso decrescente: 8 6 5 4 4 3

Procedo con uno di questi tre algoritmi:

- **Next-Fit Decreasing:** Inserisci gli oggetti in un contenitore finché c'è spazio. Quando un oggetto non ci sta, chiudi il contenitore e aprine uno nuovo.
- **First-Fit Decreasing:** Per ogni oggetto, cerca il primo contenitore disponibile con abbastanza spazio. Se non ci sta, apri un nuovo contenitore.
- **Best-Fit Decreasing:** Inserisci ogni oggetto nel contenitore che ha lo spazio minimo sufficiente, riducendo al massimo lo spazio libero rimasto.

Next-Fit Decreasing:

Cont	Ogg	Peso	Sfrido
1	1	8	2
2	2	6	4
3	3, 4	5, 4	1
4	5, 6	4, 3	3

First-Fit Decreasing:

Cont	Ogg	Peso	Sfrido
1	1	8	2
2	2, 4	6, 4	0
3	3, 5	5, 4	1
4	6	3	7

Best-Fit Decreasing:

Cont	Ogg	Peso	Sfrido
1	1	8	2
2	2, 4	6, 4	0
3	3, 5	5, 4	1
4	6	3	7

### 4. Algoritmo di riduzione del gap

#### 4.1. Risultati teorici

Assumiamo che  $A, b$  sono a componenti intere.

Sia  $(pli)$   $\begin{cases} \max c^T x \\ Ax \leq b \\ x \in \mathbb{Z}_n^+ \end{cases}$  un problema dello zaino. Sia  $(p)$   $\begin{cases} \max c^T x \\ Ax \leq b \end{cases}$  il suo rilassato continuo.

- Sia  $S := \{x \in \mathbb{Z}_n^+ : Ax \leq b\}$  la regione ammissibile di  $(pli)$ .  $S$  è un insieme finito di punti. Se lo zaino fosse binario, ci sarebbero al massimo  $2^n$  punti (vertici del cubo unitario).
- Sia  $P = \{Ax \leq b, x \geq 0\}$  il poliedro del rilassato continuo.

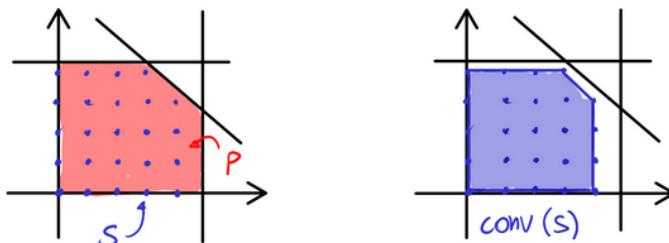


Figura 1: regione ammissibile di  $(pli)$  è contenuta in quella del suo rilassato continuo  $(p)$ .

So che il più piccolo insieme convesso che contiene  $S$  è  $\text{conv}(S)$  (assumendo che sia limitato). Per il teorema di Weil,  $\text{conv}(S)$  è un poliedro.

Mi accorgo di questa relazione:  $S \subset \text{conv}(S) \subset P$

### TEOREMA.

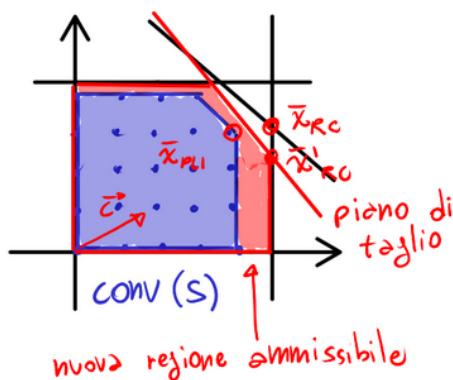
**Teorema (astratto di equivalenza tra la PL e la PLI):**

$$\text{Siano } A, b \text{ a componenti intere. Allora (pli)} \left\{ \begin{array}{l} \max c^T x \\ x \in S \end{array} \right. = (\text{pl}) \left\{ \begin{array}{l} \max c^T x \\ x \in \text{conv}(S) \end{array} \right.$$

La dimostrazione, che non vedremo, si basa sul fatto che  $\text{conv}(S)$  ha i vertici a componenti intere. Come mai si chiama teorema astratto? Risolvere un problema di PL è facile e quindi questo teorema sarebbe estremamente utile, se non fosse per il fatto che non esiste un algoritmo per costruire  $\text{conv}(S)$ . Però ci dà l'idea fondamentale per l'algoritmo di riduzione del gap.

### 4.2. Piani di taglio

**Idea:** separo l'ottimo del rilassato continuo da quello di  $\text{conv}(S)$  con dei **piani di taglio**.



Il piano di taglio è un vincolo che si aggiunge al rilassato continuo. La soluzione ottima del rilassato continuo trovata sulla nuova regione ammissibile è più vicino all'ottimo del problema di PLI.

#### Definizione.

**Definizione (disuguaglianza valida):**

Una disequazione  $\gamma x \leq \gamma_0$  è detta disuguaglianza valida (D.V.).

Si dimostra facilmente che se  $\gamma x \leq \gamma_0$  vale per  $S$ , allora vale anche per  $\text{conv}(S)$ .

#### Definizione.

**Definizione (piano di taglio):**

Sia  $\bar{x}_{RC}$  la soluzione ottima del rilassato continuo. Una D.V.  $\gamma x \leq \gamma_0$  tale che  $\gamma \bar{x}_{RC} > \gamma_0$  è detta piano di taglio (P.T.).

$$\begin{cases} \max c^T x \\ Ax \leq b \\ \gamma x \leq \gamma_0 \end{cases} \rightarrow v_{p \text{ new}} < v_s \implies v_i \leq v_{PLI} \leq \lfloor v_{p \text{ new}} \rfloor$$

Procedo iterativamente: se  $v_{p_{\text{new}}}$  è a componenti intere, vuol dire che sono arrivato all'ottimo.

### 4.3. Algoritmo riduzione del gap

*Erano gli anni 60 e Ralph E. Gomory, uno dei massimi dirigenti di IBM a New York, stava per lasciare l'azienda. Tuttavia, IBM gli offriva la libertà di lavorare sui progetti che preferiva e uno stipendio di 3 milioni di dollari, convincendolo a restare. In seguito, inventò l'algoritmo di riduzione del gap. Il giovane Pappalardo, un giorno entrò nell'ufficio di un suo allunno per un incontro. Lo trovò lì, con i piedi comodamente appoggiati sulla scrivania: una tipica abitudine americana.*

Per facilità, supponiamo che il problema di PLI sia nella forma

$$\begin{cases} \max c^T x \\ Ax = b \\ x \geq 0 \\ x \in \mathbb{Z}^n \end{cases}$$

Sia  $\bar{x}_{RC}$  la soluzione ottima del suo rilassato continuo. La soluzione ottima, essendo un vertice, è generata da una base.  $A = [A_B \ A_N]$ ,  $x = \begin{bmatrix} x_B \\ x_N \end{bmatrix}$

$$n \times m \quad m \times 1 \quad n \times 1$$

**Figura 2:** La soluzione di base rispetta i vincoli  $A\bar{x}_{RC} = b$

## Definizione.

**Definizione (parte frazionaria):**

Sia  $\alpha \in \mathbb{R}$ . La parte frazionaria di  $\alpha$  è  $\{\alpha\} := \alpha - \lfloor \alpha \rfloor$ .

Casi particolari:  $\{\pi\} = \pi - 3$ ,  $\{-\pi\} = -\pi + 4$

Sia  $\tilde{A} := A_B^{-1} A_N$  e sia  $r$  l'indice di una componente di  $\bar{x}_{RC}$  non intera. Nota che esiste necessariamente una componente non intera poiché, altrimenti, saremmo all'ottimo del problema di PLI. Inoltre,  $r \in B$  necessariamente.

### TEOREMA.

**Teorema (teorema di Gomory):**

Se  $(\bar{x}_{RC})_r \notin \mathbb{Z}$ , allora  $\sum_{j \in \mathbb{N}} \{\tilde{a}_{rj}\} x_j \geq \{(\bar{x}_{RC})_r\}$  è un P.T.  
dove  $\tilde{a}_{rj}$  è l'elemento di  $\tilde{A}$  situato alla  $r$ -esima riga e  $j$ -esima colonna.

Si osserva che  $\bar{x}_{RC}$  non verifica questa disequazione perché la parte a sinistra di  $\geq$  è sicuramente zero, mentre quella a destra è sicuramente maggiore di zero. Resta da dimostrare che il piano di taglio non taglia  $S$  e  $\text{conv}S$ , ma noi non lo faremo.

**Esempio 4.1** fare un passo dell'algoritmo di riduzione del gap al seguente problema:

$$(p) \begin{cases} \max 8x_1 + 6x_2 \\ 12x_1 + 6x_2 \leq 55 \\ 6x_1 + 8x_2 \leq 65 \\ x \geq 0, \quad x \in \mathbb{Z}^n \end{cases}$$

$$\bar{x}_{RC} = \left( \frac{5}{6}, \frac{15}{2} \right),$$

$$v_i = c^T \lfloor \bar{x}_{RC} \rfloor = c^T (0, 7) = 42,$$

$$c^T \bar{x}_{RC} = \frac{40}{6} + \frac{90}{2} = 51, 6 \implies v_s = 51.$$

Quindi  $42 \leq v_{PLI} \leq 51$ .

Non mi serve più la funzione obiettivo perché ho trovato il vertice  $\bar{x}_{RC} = \left( \frac{5}{6}, \frac{15}{2} \right)$ .

Porto il poliedro in formato duale standard:  $D := \begin{cases} 12x_1 + 6x_2 + x_3 = 55 \\ 6x_1 + 8x_2 + x_4 = 65 \\ x \geq 0 \end{cases}$

$$\bar{x}_{RC} = \left( \frac{5}{6}, \frac{15}{2}, 0, 0 \right), \quad B = \{1, 2\}, \quad A = \begin{bmatrix} 12 & 6 & 1 & 0 \\ 6 & 8 & 0 & 1 \end{bmatrix}, \quad \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array}$$

$$\tilde{A} := \begin{array}{cc} \frac{1}{2} & \frac{12}{2} \\ \frac{1}{2} & \frac{34}{2} \end{array} = A_B^{-1} I = A_B^{-1} = \frac{1}{60} \begin{bmatrix} 8 & -6 \\ -6 & 12 \end{bmatrix} = \begin{array}{cc} 3 & 4 \\ \frac{1}{2} & \frac{1}{10} \end{array} = \begin{bmatrix} 2/15 & -1/10 \\ -1/10 & 1/5 \end{bmatrix}$$

$r := 1$  (ma anche 2 va bene, non c'è nessuna regola anticiclo)

$$\text{Piano di taglio: } \left\{ \frac{2}{15} \right\} x_3 + \left\{ -\frac{1}{10} \right\} x_4 \leq \left\{ \frac{5}{6} \right\} \iff \frac{2}{15} x_3 + \frac{9}{10} x_4 \geq \frac{5}{6}$$

N.B. Per semplicità di notazione, mi riferisco alle righe di  $\tilde{A}$  usando i nomi degli indici di base (1,2), e alle colonne con gli indici non di base (3,4). Questo perché, anziché indicizzare  $A_N$  partendo da 1, uso gli indici reali corrispondenti alle variabili del problema.

**Esempio 4.2** fare un passo dell'algoritmo di riduzione del gap al seguente problema:

$$(p) \begin{cases} \max x_1 + 19x_2 \\ 12x_1 + 6x_2 \leq 53 \\ 6x_1 + 8x_2 \leq 65 \\ x \geq 0, x \in \mathbb{Z}^n \end{cases}$$

$$\bar{x}_{RC} = \left(0, \frac{65}{8}\right). \text{ Il poliedro in formato duale è } D = \begin{cases} 12x_1 + 6x_2 + x_3 = 55 \\ 6x_1 + 8x_2 + x_4 = 65 \\ x \geq 0 \end{cases}$$

$$\Rightarrow \bar{x}_{RC} = \left(0, \frac{65}{8}, \frac{25}{4}, 0\right)$$

$$A = \begin{smallmatrix} 1 & 2 & 3 & 4 \\ 2 & 6 & 1 & 0 \\ 6 & 8 & 0 & 1 \end{smallmatrix}, A_B = \begin{smallmatrix} 2 & 3 \\ 6 & 1 \\ 8 & 0 \end{smallmatrix}, A_N = \begin{smallmatrix} 1 & 4 \\ 12 & 0 \\ 6 & 1 \end{smallmatrix}, \tilde{A} := \begin{smallmatrix} 1 & 4 \\ 3/4 & 1/8 \\ 15/2 & -3/4 \end{smallmatrix}$$

Il piano di taglio per  $r := 2$  è  $\frac{3}{4}x_1 + \frac{1}{8}x_4 \geq \frac{1}{8}$

## ALGORITMO.

### Algoritmo di riduzione del gap

1. Calcolo l'ottimo del rilassato continuo  $\bar{x}_{RC}$
2. Porto il poliedro in formato duale standard e (ri)calcolo  $\bar{x}_{RC}$
3. Individuo la base che genera  $\bar{x}_{RC}$ , individuo  $A_B, A_N, r$
4. Calcolo  $\tilde{A}$
5. Scrivo il piano di taglio  $\sum_{j \in N} \{\tilde{a}_{rj}\} x_j \geq \{(\bar{x}_{RC})_r\}$

Aggiungo il piano di taglio ai vincoli del problema e torno al punto 1.

**Esempio 4.3** Fare un passo dell'algoritmo di riduzione del gap al seguente problema:

$$(p) \begin{cases} \max 30x_1 + 36x_2 + 27x_3 + 20x_4 + 24x_5 + 22x_6 \\ 13x_1 + 16x_2 + 14x_3 + 15x_4 + 17x_5 + 14x_6 \leq 57 \\ x \in \mathbb{Z}_n^+ \end{cases}$$

1.  $\bar{x}_{RC} \in \mathbb{R}^6$ ,  $\bar{x}_{RC} = \left( \frac{57}{3}, 0, 0, 0, 0, 0 \right) \Rightarrow v_s = c^T \bar{x}_{RC} = 131$ ,  $v_i = c^T [\bar{x}_{RC}] = 120$   
 $\therefore 120 \leq v_{PLI} \leq 131$
2. Il poliedro in formato duale,  $D := \begin{cases} 13x_1 + 16x_2 + 14x_3 + 15x_4 + 17x_5 + 14x_6 + x_7 \leq 57 \\ x \geq 0 \end{cases}$   
 Ricalcolo  $\bar{x}_{RC} \in \mathbb{R}^7$ ,  $\bar{x}_{RC} = \left( \frac{57}{3}, 0, 0, 0, 0, 0, 0 \right)$ ,
3.  $B = \{1\}$ ,  $N = \{2, 3, 4, 5, 6, 7\}$ ,  $A_B = [13]$ ,  $A_N = [16 \ 14 \ 15 \ 17 \ 14 \ 1]$ ,  $r := 1$
4.  $\tilde{A} = A_B^{-1} A_N = \left[ \begin{array}{cccccc} \frac{16}{13} & \frac{14}{13} & \frac{15}{13} & \frac{17}{13} & \frac{14}{13} & \frac{1}{13} \end{array} \right]$
5.  $\frac{3}{13}x_2 + \frac{1}{13}x_3 + \frac{2}{13}x_4 + \frac{4}{13}x_5 + \frac{1}{13}x_6 + \frac{1}{13}x_7 \leq \frac{5}{13} \Leftrightarrow$   
 $3x_2 + x_3 + 2x_4 + 4x_5 + x_6 + x_7 \leq 5$  è il piano di taglio.

**Esempio 4.4** Fare un passo dell'algoritmo di riduzione del gap al seguente problema:

$$(p) \begin{cases} \max 30x_1 + 36x_2 + 27x_3 + 20x_4 + 24x_5 + 22x_6 \\ 13x_1 + 16x_2 + 14x_3 + 15x_4 + 17x_5 + 15x_6 \leq 57 \\ x \in \{0, 1\}^n \end{cases}$$

1.  $\bar{x}_{RC} \in \mathbb{R}^6$ ,  $\bar{x}_{RC} = \left( 1, 1, 1, 0, 0, \frac{14}{15} \right)$
2. Attenzione:  $D \neq \begin{cases} 13x_1 + 16x_2 + 14x_3 + 15x_4 + 17x_5 + 15x_6 + x_7 = 57 \\ x \geq 0 \end{cases}$

Quando passo al duale, devo aggiungere il vincolo  $0 \leq x \leq 1$ .

$$D = \begin{cases} 13x_1 + 16x_2 + 14x_3 + 15x_4 + 17x_5 + 15x_6 + x_7 = 57 \\ x_1 + x_8 = 1 \\ x_2 + x_9 = 1 \\ x_3 + x_{10} = 1 \\ x_4 + x_{11} = 1 \\ x_5 + x_{12} = 1 \\ x_6 + x_{13} = 1 // \text{non devo aggiungere anche } x_7 \text{ poiché è la variabile di scarto.} \\ x \geq 0 \end{cases}$$

$$\therefore \bar{x}_{RC} = \left( 1, 1, 1, 0, 0, \frac{14}{15}, 0, 0, 0, 0, 1, 1, \frac{1}{15} \right)$$

$$3. B = \{1, 2, 3, 6, 11, 12, 13\}, \#B = 7; \quad N = \{4, 5, 7, 8, 9, 10\}, \#N = 6; \quad r := 6$$

$$A = \begin{array}{c|cccccccccccccc} & \textcolor{red}{1} & \textcolor{red}{2} & \textcolor{red}{3} & \textcolor{red}{4} & \textcolor{red}{5} & \textcolor{red}{6} & \textcolor{red}{7} & \textcolor{red}{8} & \textcolor{red}{9} & \textcolor{red}{10} & \textcolor{red}{11} & \textcolor{red}{12} & \textcolor{red}{13} \\ \hline \textcolor{red}{1} & 15 & 16 & 14 & 15 & 17 & 15 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \textcolor{red}{2} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \textcolor{red}{3} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \textcolor{red}{4} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \textcolor{red}{5} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \textcolor{red}{6} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \textcolor{red}{7} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

$$A_B = \begin{array}{c|cccccc} & \textcolor{red}{1} & \textcolor{red}{2} & \textcolor{red}{3} & \textcolor{red}{6} & \textcolor{red}{11} & \textcolor{red}{12} & \textcolor{red}{13} \\ \hline \textcolor{red}{1} & 15 & 16 & 14 & 15 & 0 & 0 & 0 \\ \textcolor{red}{2} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \textcolor{red}{3} & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \textcolor{red}{4} & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \textcolor{red}{5} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \textcolor{red}{6} & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \textcolor{red}{7} & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} \quad (7 \times 7), \quad A_N (7 \times 6)$$

$$4. \widetilde{A} (7 \times 6) = \begin{bmatrix} & \textcolor{red}{4} & \textcolor{red}{5} & \textcolor{red}{7} & \textcolor{red}{8} & \textcolor{red}{9} & \textcolor{red}{10} \\ \textcolor{red}{1} & A_B^{-1} & \left[ \begin{array}{c|cccccc} & \textcolor{red}{4} & \textcolor{red}{5} & \textcolor{red}{7} & \textcolor{red}{8} & \textcolor{red}{9} & \textcolor{red}{10} \\ \hline \textcolor{red}{2} & & & & & & \\ \textcolor{red}{3} & & & & & & \\ \textcolor{red}{6} & & x & x & x & x & x \\ \textcolor{red}{11} & & x & x & x & x & x \\ \textcolor{red}{12} & & x & x & x & x & x \\ \textcolor{red}{13} & & x & x & x & x & x \end{array} \right] & \text{non lo continuo...} \\ \textcolor{red}{(7 \times 7)} & (7 \times 6) & \end{bmatrix}$$

**Esempio 4.5** Fare un passo dell'algoritmo di riduzione del gap al seguente problema:

$$(p) \begin{cases} \min 5x_1 + 14x_2 \\ 16x_1 + 13x_2 \geq 62 \\ 6x_1 + 15x_2 \geq 52 \\ x \in \mathbb{Z}_n^+ \end{cases}$$

$$1. \bar{x}_{RC} = \left( \frac{26}{3}, 0 \right) \text{ risolto con linprog.}$$

$$2. D := \begin{cases} 16x_1 + 13x_2 - x_3 = 62 \\ 6x_1 + 15x_2 - x_4 = 52 * , \bar{x}_{RC} = \left( \frac{26}{3}, 0, x_3, 0 \right) \\ x \geq 0 \end{cases}$$

$$\text{Non mi interessa sapere } x_3 = \frac{26}{3} \cdot 16 - 62 \text{ perché tanto ho già l'indice } r := 1$$

3.  $B = \{1, 3\}$ ,  $A_B = \begin{bmatrix} 16 & -1 \\ 6 & 0 \end{bmatrix}$ ,  $A_N = \begin{bmatrix} 13 & 0 \\ 15 & -1 \end{bmatrix}$ ,  $r := 1$

4.  $\tilde{A} = A_B^{-1} A_N = \begin{bmatrix} 15/6 & -1/6 \\ \dots & \dots \end{bmatrix}$ , non mi interessa sapere la seconda riga in quanto  $r = 1$

5. P.T.  $\frac{3}{6}x_2 + \frac{5}{6}x_4 \geq \frac{2}{3} \Leftrightarrow 3x_2 + 5x_4 \geq 2$

Attenzione:  $3x_2 + 5x_4 \geq 2 \Leftrightarrow 5x_1 + 13x_2 \geq 44$  è lo stesso piano di taglio!

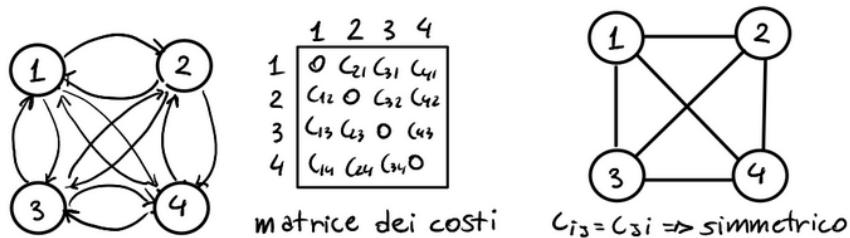
Si ottiene ricavando la variabile di scarto  $x_4$  dalle  $x_1, x_2$ :

\*  $x_4 = 6x_1 + 15x_2 - 52$ , quindi  $3x_2 + 5(6x_1 + 15x_2 - 52) \geq 4 \Rightarrow \dots \Rightarrow 5x_1 + 13x_2 \geq 44$

## 5. Problema del commesso viaggiatore

**TSP: Travelling Salesman Problem.** Dato un insieme di città, e note le distanze tra ciascuna coppia di esse, trovare il tragitto di minima percorrenza che un commesso viaggiatore deve seguire per visitare tutte le città una ed una sola volta e ritornare alla città di partenza.

Dato un grafo  $(N, A)$  completo, con  $c_{ij}$  costi sugli archi, trova un ciclo di costo minimo che passi su tutti i nodi una ed una sola volta (ciclo hamiltoniano).



### 5.1. TSP asimmetrico

Le soluzioni ammissibili sono tutte le permutazioni dei nodi, ovvero  $n!$

Introduco  $x_{ij} = \begin{cases} 1, & \text{il nodo } (i, j) \text{ viene utilizzato} \\ 0, & \text{altrimenti} \end{cases}$

La funzione da minimizzare è  $c^T x$ , dove  $x \in \mathbb{Z}^{n^2}$  (è didatticamente più semplice se si lasciano gli archi  $(i, j) : i = j$ ).

Ogni nodo deve avere un solo arco uscente:

$$\begin{cases} x_{11} + x_{12} + x_{13} + x_{14} = 1 \\ x_{21} + x_{22} + x_{23} + x_{24} = 1 \\ x_{31} + x_{32} + x_{33} + x_{34} = 1 \\ x_{41} + x_{42} + x_{43} + x_{44} = 1 \end{cases}$$

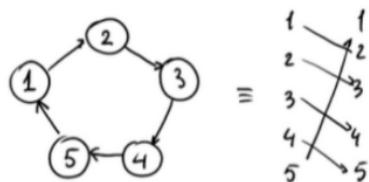
Ogni nodo deve avere un solo arco entrante:

$$\begin{cases} x_{11} + x_{21} + x_{31} + x_{41} = 1 \\ x_{12} + x_{22} + x_{32} + x_{42} = 1 \\ x_{13} + x_{23} + x_{33} + x_{43} = 1 \\ x_{14} + x_{24} + x_{34} + x_{44} = 1 \end{cases}$$

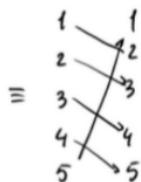
Con solo questi vincoli il problema del commesso viaggiatore pare un problema di assegnamento, che è banale da risolvere.

### Proposizione.

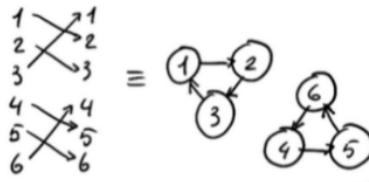
Tutti i cicli hamiltoniani sono assegnamenti, ma non tutti gli assegnamenti sono cicli hamiltoniani. Tutti gli assegnamenti sono unione di cicli hamiltoniani disgiunti.



Ciclo hamiltoniano.



Assegnamento.



Assegnamento.

Cicli hamiltoniani  
disgiunti.

Il vincolo che bisogna aggiungere è il vincolo di **connessione**. Il problema del commesso viaggiatore è **difficile**, NP completo.

Nell'esempio della Figura 3:  $x_{14} + x_{15} + x_{16} + x_{24} + x_{25} + x_{26} + x_{34} + x_{35} + x_{36} \geq 1$  dev'esserci almeno un collegamento tra i due cicli disgiunti.

In generale, questa condizione si esprime chiedendo che qualunque sottoinsieme dei nodi con qualunque complementare devono essere sempre connessi.

Sia  $N$  l'insieme dei nodi:  $\sum_{\substack{i \in S \\ j \notin S}} x_{ij} \geq 1 \quad \forall S \subseteq N, S \neq \emptyset, S \neq N, 1 < |S| < n - 1$

Posso togliere i casi particolari  $|S| = 0, |S| = 1, |S| = n - 1, |S| = |N|$ , ovvero  $2n + 2$  vincoli, ma la complessità è comunque esponenziale. Infatti, ci sono  $2^n$  sottoinsiemi  $S$  di  $N$ .

$\therefore$  In totale, si aggiungono  $2^n - 2n - 2$  vincoli.

### Modello.

TSP asimmetrico:

$$\begin{cases} \min c^T x \\ \sum_{i \in N \setminus \{j\}} x_{ij} = 1 \quad \forall j \in N \\ \sum_{j \in N \setminus \{i\}} x_{ij} = 1 \quad \forall i \in N \\ \sum_{i \in S, j \notin S} x_{ij} \geq 1 \quad \forall S \subseteq N, S \neq \emptyset, N \\ x_{ij} \in \{0, 1\} \end{cases}$$

Il problema è difficile perché non si riescono a trovare tutti i vincoli, ovvero trovare la soluzione ottima del rilassato continuo! Allora si tolgono i vincoli di connessione e si ottengono le stime inferiori e superiori in un altro modo.

#### 5.1.1. Valutazione inferiore (assegnamento di costo minimo)

$v_i$  si trova rilassando il problema di TSP in un problema di assegnamento di costo minimo. In generale, rilassare significa trovare un insieme sul quale è facile risolvere il problema che contiene la soluzione al problema di partenza.

#### 5.1.2. Valutazione superiore (algoritmo delle toppe)

$v_s$  si trova con l'algoritmo delle toppe. In alternativa, va bene un qualsiasi algoritmo greedy che fornisca una soluzione ammissibile.

### Algoritmo.

**Algoritmo delle toppe:**

1. Calcolo l'assegnamento di costo minimo
2. Seleziono un arco  $(i, j)$  del primo ciclo e un arco  $(k, l)$  del secondo.
3. Li elimino e li sostituisco con  $(i, l)$  e  $(k, j)$ , rispettivamente.

L'algoritmo delle toppe è un algoritmo greedy che ci permette di ottenere un ciclo hamiltoniano.

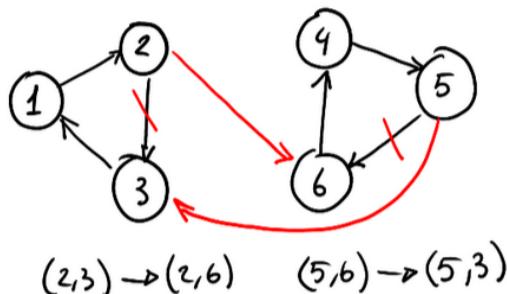


Figura 3: Algoritmo delle toppe,  $(i, j) \rightarrow (i, l); (k, l) \rightarrow (k, j)$

Nella Figura 4:  $v_{ass} - c_{23} - c_{56} + c_{26} + c_{53} = v_i \geq 0$  sicuramente, perché se fosse negativa il nuovo assegnamento sarebbe migliore, ma io ho già scelto l'assegnamento di costo minimo.

## 5.2. TSP simmetrico

Caso particolare di TSP.  $c_{ij} = c_{ji}$  quindi si dimezzano le variabili  $\frac{n^2 - n}{2} \in \mathbb{Z}$

$x_{ij} : i < j$ , nell'esempio di 5 nodi:  $x = (x_{12}, x_{13}, x_{14}, x_{15}, x_{23}, x_{24}, x_{25}, x_{34}, x_{35}, x_{45})$

### 5.2.1. Valutazione superiore (algoritmo del nodo più vicino)

$v_s$  si trova con l'algoritmo del nodo più vicino. Se parto dal nodo  $i$ , scelgo di collegarlo al nodo  $j$  tale che  $c_{ij}$  è il minimo.

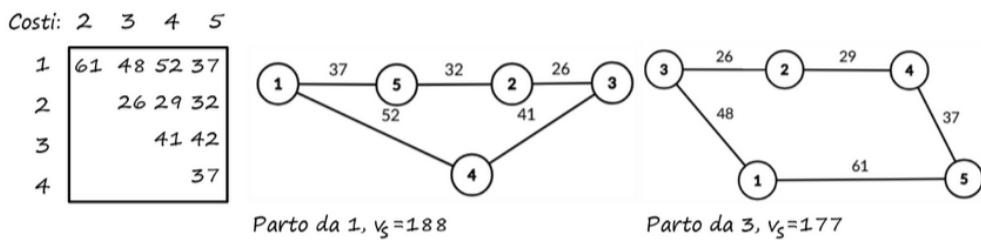


Figura 4: Partendo da nodi diverse non si ottiene la stessa soluzione, ma l'algoritmo mi garantisce che riesco sempre a trovare una valutazione superiore.

### 5.2.2. Valutazione inferiore ( $k$ -albero di costo minimo)

Nel caso simmetrico, non si può usare l'assegnamento di costo minimo.

**$k$ -albero:** isolo il nodo  $k$  e costruisco un albero di copertura con i nodi rimanenti. Collego il nodo  $k$  all'albero di copertura con due archi.

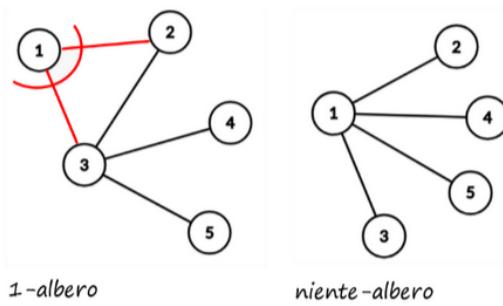


Figura 5: Un  $k$ -albero ha sempre  $n$  archi.

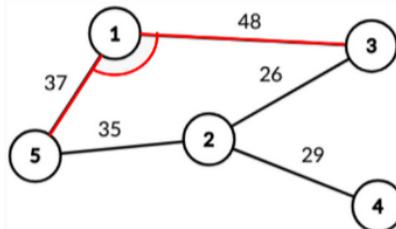
Per trovare il  $k$ -albero di costo minimo trovo il sottoalbero di copertura di costo minimo con l'algoritmo di Kruskal, poi aggiungo i due rami che collegano  $k$  meno costosi. Si può dimostrare che l'algoritmo di Kruskal è ottimo; siccome scelgo i rami di costo minimo, l'algoritmo appena descritto è ottimo.

### Algoritmo.

#### Algoritmo di Kruskal:

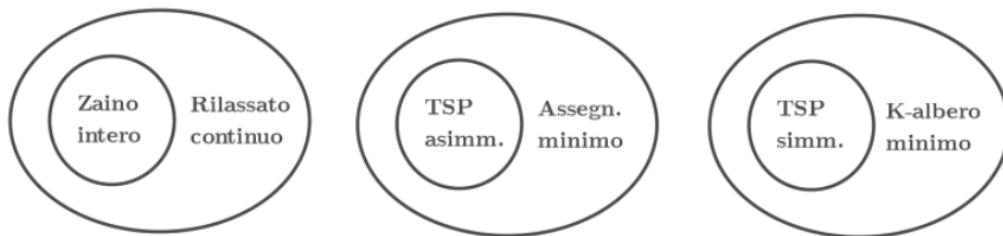
Per costruire l'albero di copertura di costo minimo (Minimum Spanning Tree)

1. Ordina tutti gli archi in ordine non decrescente in base al loro peso
2. Scegli l'arco più piccolo. Controlla se forma un ciclo con l'albero di copertura già formato. Se non si forma un ciclo, includi questo arco. Altrimenti, scartalo.
3. Ripeti il passo 2 finché non ci sono  $(n - 1)$  archi nell'albero di copertura.



**Ho trovato un rilassamento:** ogni ciclo hamiltoniano è un  $k$ -albero. Se un  $k$ -albero è un ciclo hamiltoniano, allora sono all'ottimo! In generale, se l'ottimo di un rilassato è ammissibile, vuol dire che sono arrivato all'ottimo del problema di PLI.

Soluzioni ottime:



### Modello.

#### TSP simmetrico:

$$\begin{cases} \min c^T \cdot x \\ \sum_{i < j} x_{ij} + \sum_{j < i} x_{ji} = 2 \quad \forall j & \leftarrow \text{vincoli di grado} \\ \sum_{\substack{i \in S, j \notin S, i < j}} x_{ij} + \sum_{\substack{i \notin S, j \in S, j < i}} \geq 1 \quad \forall j, S \subset N & \leftarrow \text{vincoli di connessione} \\ x_{ij} \in \{0, 1\} \end{cases}$$

Le variabili sono  $|x| = \frac{n!}{|S|!(n - |S|)!}$

La matrice  $A_{eq}$  ha dimensione  $n \times |x|$

La matrice  $A$  ha dimensione  $(2^n - 2n - 2) \times |x|$

## 6. Algoritmo del Branch and Bound

Il Branch and Bound è un approccio che può essere applicato a una vasta gamma di problemi di PLI, inclusi problemi NP-hard come il TSP. Il TSP asimmetrico non può essere risolto con l'algoritmo di riduzione del gap perché non riesco a scrivere i vincoli.

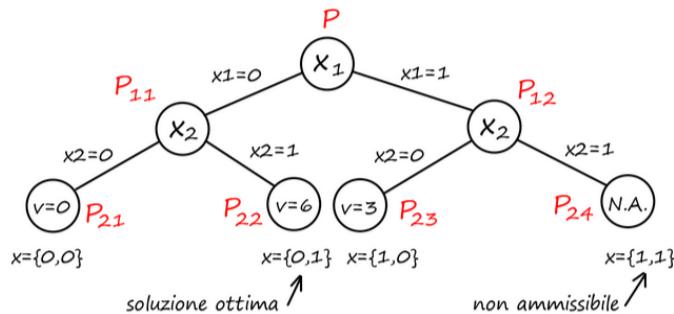
Per semplicità considero solo problemi di zaino booleano e di TSP simmetrico.

### 5.1. Albero di enumerazione totale

Considero (p)  $\begin{cases} \max x_1 + x_2 \\ 3x_1 + 6x_2 \leq 7 \\ x \in \{0, 1\} \end{cases}$  un problema di zaino intero.

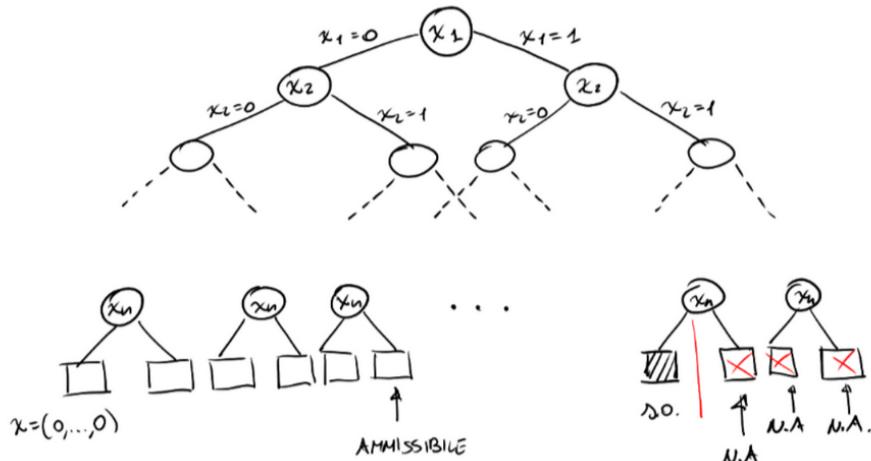
Il problema (p) si può partizionare in due sottoproblemi ( $p_{11}$ ) se  $x_1 = 0$ , ( $p_{12}$ ) se  $x_1 = 1$ . I sottoproblemi ( $p_{11}$ ), ( $p_{12}$ ) possono essere partizionati allo stesso modo per  $x_2 = 0$ ,  $x_2 = 1$ .

Posso disegnare un albero binario con quattro foglie, che rappresentano tutti i casi possibili. Sicuramente, la soluzione ottima si trova in una foglia.



**Figura 6:** Albero di enumerazione totale di un problema di zaino binario a due variabili

In generale, se ho un problema dello zaino intero (con anche più di due variabili), il suo albero di enumerazione totale è:



**Figura 7:** A partire da un certo punto, tutte le foglie sono inammissibili. Ci sono  $2^n$  foglie.

Un metodo per risolvere un problema dello zaino (con anche più di due variabili) è il **metodo dell'enumerazione esplicita**: disegno l'albero di enumerazione totale e controllo le foglie per trovare la soluzione ottima. Questo metodo è esponenziale e quindi inefficiente.

## 6.2. Branch and Bound

L'algoritmo del Branch and Bound si basa sull'albero di enumerazione totale. Nonostante anch'esso sia esponenziale, è molto più efficiente perché si "taglano" i sottoalberi generati dai nodi oltre i quali sono sicuro che non ci sarà la soluzione ottima. L'algoritmo termina quando ho visitato, esplicitamente o implicitamente, tutte le foglie.

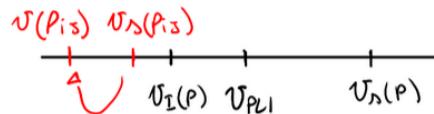
- **Branch:** partiziono la regione ammissibile per generare sottoproblemi (nel caso di zaino booleano è semplice, è un albero binario).
- **Bound:** stimo il valore dell'ottimo di ogni sottoproblema e utilizzo regole di taglio per depennare le sottoregioni che non contengono l'ottimo.

### Corollario.

#### Regole di taglio per zaino booleano (max):

Sia  $P_{ij}$  il sottoproblema situato nell'albero a livello  $i$  e a posizione orizzontale  $j$ . Valgono le seguenti:

1. Se  $P_{ij} = \emptyset$  allora devo chiudere il nodo  $P_{ij}$  perché tutto il sottoalbero è inammissibile.
2. Se  $v_s(P_{ij}) \leq v_i(P)$  allora devo chiudere il nodo  $P_{ij}$ . Ogni passo dell'algoritmo richiede il calcolo di  $v_s$  dello zaino booleano con il metodo che già conosciamo.



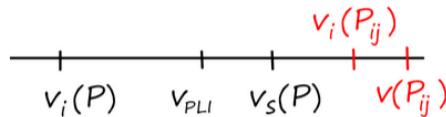
3. Se  $v_s(P_{ij}) > v_i(P)$ , con  $v_s(P_{ij})$  ammissibile i.e. a componenti intere, allora devo aggiornare  $v_i(P) = v_s(P_{ij})$  e chiudere il nodo  $P_{ij}$  (per la 2.)

### Corollario.

#### Regole di taglio per TSP simmetrico (min):

Sia  $P_{ij}$  il sottoproblema situato nell'albero a livello  $i$  e a posizione orizzontale  $j$ . Valgono le seguenti:

1. Se  $P_{ij} = \emptyset$  allora devo chiudere il nodo  $P_{ij}$  perché tutto il sottoalbero è inammissibile.
2. Se  $v_i(P_{ij}) \geq v_s(P)$  allora devo chiudere il nodo  $P_{ij}$ . Ogni passo dell'algoritmo richiede il calcolo di  $v_i$  del TSP con il metodo dei K-alberi di costo minimo.



3. Se  $v_i(P_{ij}) < v_s(P)$ , con  $v_i(P_{ij})$  ammissibile i.e. a componenti intere, allora devo aggiornare  $v_s(P) = v_i(P_{ij})$  e chiudere il nodo  $P_{ij}$  (per la 2.).

**Domanda:** se ho un TSP simmetrico, ad ogni passo calcolo un nuovo ciclo hamiltoniano? No. Mi basta calcolare un k-albero. Infatti, per verificare se un k-albero è un ciclo hamiltoniano, basta verificare i vincoli di grado (quelli di connessione sono già verificati).

### ALGORITMO.

#### Algoritmo del Branch and Bound

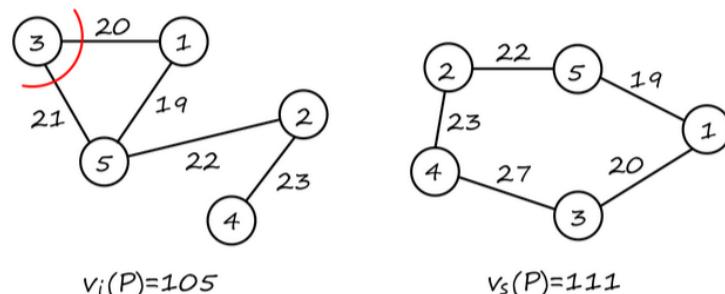
1. Calcola  $v_i(P)$  se max,  $v_s(P)$  se min.
2. Se tutti i nodi sono stati esplorati allora STOP
3. Seleziona il nodo (sottoproblema) da esplorare
4. Calcola  $v_s(P_{ij})$  se max,  $v_i(P_{ij})$  se min.
5. Controlla le regole di taglio.
6. Se il nodo non è chiuso, allora scendi
7. Vai al passo 2.

**Esempio 6.1** Sia data la matrice dei costi di un TSP simmetrico:

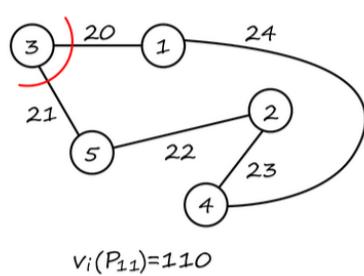
	2	3	4	5
1	26	20	24	19
2		34	23	22
3			27	21
4				32

➤ Fare un passo dell'algoritmo del Branch and Bound istanziando la variabile  $x_{15}$ .

Trovo innanzitutto le valutazioni inferiori e superiori. Faccio un  $k$ -albero di costo minimo e l'algoritmo del nodo più vicino.



- $P_{11}$ :  $x_{15} = 0$  i.e. non dev'esserci l'arco  $(1, 5)$ .

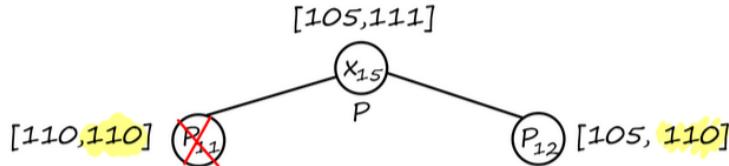


$110 = v_i(P_{11}) > v_s(P) = 111$ . Siccome è anche un ciclo hamiltoniano, ovvero è ammissibile, devo aggiornare  $v_s(P) = 110$  e chiudere il nodo  $P_{11}$

- $P_{12}$ :  $x_{15} = 1$  i.e. dev'esserci l'arco  $(1, 5)$ .

Siccome c'era già quando avevo calcolato  $v_i(P)$ , allora

$105 = v_i(P) = v_i(P_{12}) < v_s(P) = 111$ . Non è ammissibile (anche perché altrimenti prima sarei già arrivato all'ottimo), quindi non faccio niente.



➤ Fare un altro passo dell'algoritmo del Branch and Bound istanziando la variabile  $x_{35}$ .

- $P_{23}$ :  $x_{15} = 1, x_{35} = 0$  i.e. dev'esserci l'arco  $(1, 5)$ , non dev'esserci l'arco  $(3, 5)$ .

$v_i(P_{23}) = 111 \geq 110$ , quindi chiudo il nodo  $P_{23}$ .

- $P_{24}$ :  $x_{15} = 1, x_{35} = 0$  i.e. devono esserci gli archi  $(1, 5)$  e  $(3, 5)$ .

$v_i(P_{24}) = v_i(P_{12}) = 105$ , ed è lo stesso caso di prima, in quanto l'arco  $3, 5$  esiste già nella soluzione che genera  $v_i(P_{12})$ .

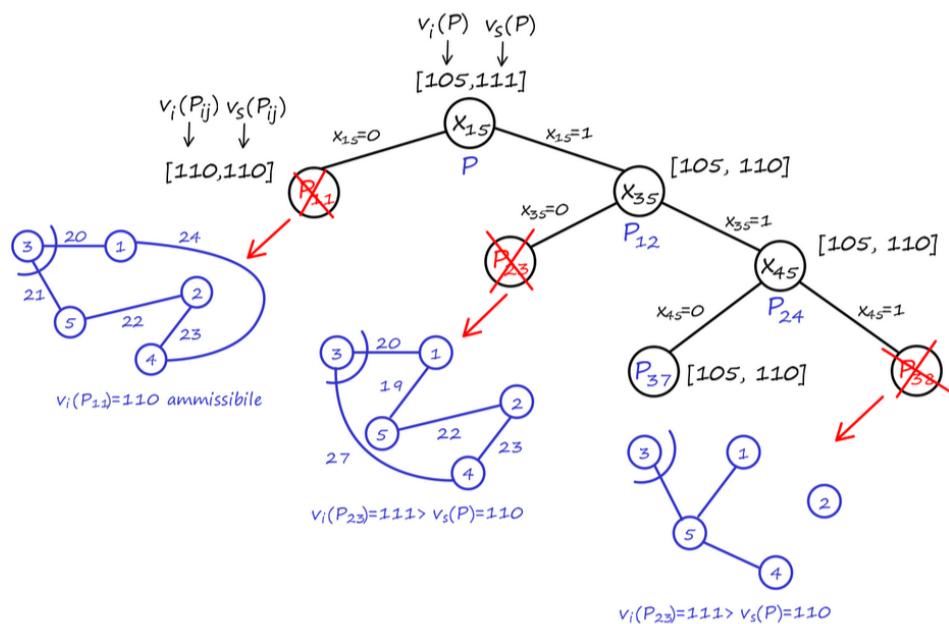
➤ Fare un altro passo dell'algoritmo del Branch and Bound istanziando la variabile  $x_{45}$ .

- $P_{37}$ :  $x_{15} = 1, x_{35} = 1, x_{45} = 0$  i.e. devono esserci  $(1, 5)$  e  $(3, 5)$ , ma non  $(4, 5)$ .

$v_i(P_{37}) = v_i(P_{24}) = 105$ , non cambia niente.

- $P_{38}$ :  $x_{15} = 1, x_{35} = 1, x_{45} = 1$  i.e. devono esserci  $(1, 5), (3, 5)$  e  $(4, 5)$ .

Mi accorgo che il problema  $P_{38} = \emptyset$  perché non esiste un  $k$ -albero con archi  $(1, 5), (3, 5), (4, 5)$ . Quindi chiudo il nodo  $P_{38}$ . **È un errore calcolare  $v_i(P_{38})$ .**



**Esempio 6.2** Sia dato un problema di zaino booleano con:

$$\begin{aligned} v &= [7 \quad 21 \quad 17 \quad 6 \quad 22 \quad 19 \quad 8] \\ p &= [479 \quad 272 \quad 173 \quad 473 \quad 54 \quad 230 \quad 25], \quad c = 531 \end{aligned}$$

Trovo innanzitutto la valutazione superiore e inferiore:

$$r := (0.015, 0.08, 0.098, 0.013, 0.41, 0.083, 0.32)$$

$$x_{RC} = \left(0, 1, 1, 0, 1, \frac{57}{230}, 1\right) \Rightarrow v_s = \lfloor c \cdot x_{RC} \rfloor = 72$$

$$x_a = (0, 1, 1, 0, 1, 0, 1) \Rightarrow v_i = 68 \quad \therefore 68 \leq v_{PLI} \leq 72$$

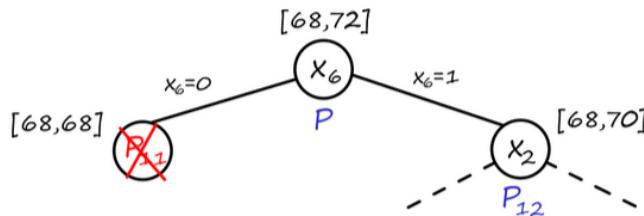
➤ Fare passo dell'algoritmo del Branch and Bound.

Quale variabile mi conviene istanziare? La **variabile frazionaria del rilassato continuo**,  $x_6$

$$P_{11}: \text{setto } x_6 = 0. \quad x_{RC} = \left(\frac{57}{479}, 1, 1, 0, 1, \cancel{0}, 1\right) \Rightarrow v_s(P_{11}) = \lfloor 68 + 7 \cdot (57 / 479) \rfloor = 68$$

$v_s(P_{11}) > v_i(P)$  ed è ammissibile, quindi setto  $v_i(P) = v_s(P_{11})$  e chiudo il nodo  $P_{11}$

$$P_{12}: \text{setto } x_6 = 1. \quad x_{RC} = \left(0, \frac{49}{222}, 1, 0, 1, \cancel{1}, 1\right) \Rightarrow v_s(P_{12}) = 70 > v_i(P).$$



**Figura 8:** Se voglio fare un altro passo, devo istanziare la variabile frazionaria, ovvero  $x_2$

## 7. Problema di copertura

### 7.1. Copertura di costo minimo

Dato un territorio suddiviso in 7 quartieri e 5 potenziali posizioni per un'ambulanza, l'obiettivo è posizionare il minor numero possibile di ambulanze, garantendo che tutti i quartieri siano coperti dal servizio di emergenza.

Quartieri	1	2	3	4	5	Depositi $\mathbf{X}_i$
1	1	1	0	0	1	
2	0	1	0	1	0	
3	1	0	0	0	0	
4	1	0	1	0	1	
5	1	0	0	1	1	
6	1	1	1	0	0	
7	1	0	1	1	0	
	$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	Costi (opzionale)

$$i = 1 \dots 5, \quad x_i = \begin{cases} 0, & \text{non metto l'ambulanza nel deposito } i \\ 1, & \text{metto l'ambulanza nel deposito } i \end{cases}$$

$$\begin{cases} \min x_1 + x_2 + x_3 + x_4 + x_5 \\ (\text{oppure}) \min c_1 x_1 + c_2 x_2 + c_3 x_3 + c_4 x_4 + c_5 x_5 \quad \text{se le scelte hanno un costo.} \end{cases}$$

In ogni quartiere dev'esserci un'ambulanza che la possa servire.

$$\begin{cases} x_1 + x_2 + x_5 \geq 1 & \text{devo servire il quartiere 1} \\ x_2 + x_4 \geq 1 \\ x_1 \geq 1 \\ x_1 + x_3 + x_5 \geq 1 \\ x_1 + x_4 + x_5 \geq 1 \\ x_1 + x_2 + x_3 \\ x_1 + x_3 + x_4 \geq 1 \end{cases}$$

### Modello.

Copertura di costo minimo:

$$(p) \begin{cases} \min c^T x \\ Ax \stackrel{\rightarrow}{\geq} 1 \\ x \in \{0, 1\}^n \end{cases}$$

#### 7.1.1. Valutazione inferiore $v_i$

Si trova con il rilassato continuo

$$\begin{cases} \min c^T x \\ \sum_{i=1}^n a_{ij} x_j \geq 1 \\ 0 \leq x_j \leq 1 \end{cases}$$

#### 7.1.2. Valutazione superiore $v_s$

Prima di tutto opero le regole di riduzione:

1. Una riga è tutta a zero  $\rightarrow$  problema vuoto
2. Una riga è tutta a 1  $\rightarrow$  la tolgo
3. Una riga ha un solo 1  $\rightarrow$  sono obbligato a prendere il sito corrispondente, cancello la colonna del sito e le righe dove la colonna ha  $a_{ij} = 1$

	1	2	3	4	5
1	1	1	0	0	1
2	0	1	0	1	0
3	1	0	0	0	0
4	1	0	1	0	1
5	1	0	0	1	1
6	1	1	1	0	0
7	1	0	1	1	0

#### 4. Regola della dominanza:

la riga 3 è "dominata" dalla riga 6.

Cancello la riga 5 perché devo lasciare in ballo le richieste di 4 perché sono più stringenti.

**Def:** Si dice che  $a \in \mathbb{R}^m$  è dominato da  $b \in \mathbb{R}^m$  se  $a_i \leq b_i \forall i$

	1	2	3	4	5
1	1	1	0	0	1
2	0	1	0	1	0
3	1	0	0	0	0
4	1	0	1	0	1
5	1	0	0	1	1
6	1	1	1	0	0
7	1	0	1	1	0

"3"  $\leq$  "6"  
La "3" è dominata dalla "6"

#### 5. Regola dei costi:

se la colonna  $i$  è dominata dalla colonna  $j$  e  $c_i > c_j$ , elimino  $i$ .

	1	2	3	4	5
1	1	1	0	0	1
2	0	1	0	1	0
3	1	0	0	0	0
4	1	0	1	0	1
5	1	0	0	1	1
6	1	1	1	0	0
7	1	0	1	1	0

$c=2$      $c=10$

"3"  $\leq$  "1"  
Elimino la colonna 3 se  $c_3 > c_1$

Una volta che ho ridotto il problema in uno più piccolo trovo una soluzione ammissibile.

Se tutte le scelte hanno costo 1, apro il servizio che soddisfa più clienti, lo pongo a 1 e cancello la sua colonna e tutte le righe da lui servite. Calcolo di nuovo il servizio che soddisfa più clienti e ripeto.

La soluzione ammissibile nel caso con i costi si trova con l'algoritmo di Chvatal:

- Costruisco il vettore dei costi unitari di copertura  $u = \left( \frac{c_1}{6}, \frac{c_2}{3}, \frac{c_3}{3}, \frac{c_4}{3}, \frac{c_5}{3} \right)$
- Posiziono il servizio nel sito tale che  $r$  è il minimo. Elimino la colonna e le righe servite.

Supponiamo  $\frac{c_2}{3} = \min(r)$ . Allora elimino la colonna 2 e le righe 1, 2, 6.

	1	2	3	4	5
1	1	0	0	1	
2	0	0	1	0	
3	1	0	0	0	0
4	1	0	1	0	1
5	1	0	0	1	1
6	1	1	0	0	
7	1	0	1	1	0

- Ricalcolo i costi unitari di copertura  $u = \left( \frac{c_1}{4}, \frac{c_3}{2}, \frac{c_4}{2}, \frac{c_5}{2} \right)$  e procedo iterativamente.

### Algoritmo.

#### Algoritmo di Chvatal:

Per trovare una soluzione ammissibile ad un problema di copertura.

1.  $I = \{1, \dots, m\}, J = \{1, \dots, n\}, x := 0$
2. Per ogni  $j \in J$  calcola  $u_j = \frac{c_j}{\sum_{i \in I} a_{ij}}$
3. Trova un indice  $k \mid u_k = \min u_j \forall j \in J$   
Poni  $x_k := 1$ , da  $J$  elimina  $k$ , da  $I$  elimina  $\{i \mid a_{ik} = 1\}$
4. Se  $I = \emptyset$  allora STOP ( $x$  è una copertura)
5. Se  $J = \emptyset$  allora STOP (non esistono coperture)
- Altrimenti torna al passo 2.

## 7.2. Massima copertura

Dato un territorio suddiviso in 8 quartieri e 4 potenziali posizioni dove aprire un supermercato, l'obiettivo è posizionare due supermercati, massimizzando il numero di persone (per ogni quartiere  $i$  ci sono  $h_i$  persone) coperte.

$$z_i = \begin{cases} 1, & \text{il quartiere } i \text{ viene servito} \\ 0, & \text{il quartiere } i \text{ non viene servito} \end{cases}$$

Luoghi di apertura  $X_i$ :

Quartieri	1	2	3	4	A	2000	1000	800	3000	2500	1800	1700	480	h
1	1	1	0	0										
2	0	0	1	1										
3	1	0	1	0										
4	1	0	0	1										
5	0	1	0	1										
6	1	0	1	1										
7	0	1	0	1										
8	1	0	0	1										

$$\left\{ \begin{array}{l} \max h_1 z_1 + \dots + h_8 z_8 \\ x_1 + x_2 + x_3 + x_4 = 2 \quad (\text{apro 2 supermercati}) \\ x_1 + x_2 \geq z_1 \\ x_3 + x_4 \geq z_2 \\ \dots \\ x_1 + x_4 \geq z_8 \\ x \in \{0,1\}^4 \\ z \in \{0,1\}^8 \end{array} \right.$$

### Modello.

#### Problema di copertura:

$$\text{Copertura di costo minimo: } \left\{ \begin{array}{l} \min c^T x \\ \vec{Ax} \geq \vec{1} \\ x \in \{0,1\}^n \end{array} \right. \quad \text{Massima copertura: } \left\{ \begin{array}{l} \max h^T z \\ Ax \geq z_i \\ \sum x_i > p \\ x \in \{0,1\}^n \\ z \in \{0,1\}^m \end{array} \right.$$