1 Metodi risolutivi per PLI

Illustriamo due metodi classici per la risoluzione di problemi di PLI.

Metodi di taglio: restringono opportunamente la regione ammissibile del problema continuo.

Metodi di branch & bound: eliminano interi sottoinsiemi di soluzioni ammissibili, senza enumerare esplicitamente tutte le soluzioni intere in essi contenute.

1.1 Metodi di taglio

Sia dato il problema di PLI

$$(PLI) \quad \begin{cases} \min c^T x \\ Ax = b \\ x \geq 0 \\ x \text{ intero}, \end{cases}$$

con $X = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$.

I metodi di taglio si basano sull'idea di approssimare l'involucro convesso di X a partire dal rilassamento lineare. Si risolve quindi una successione di problemi di PL via via più vincolati, fino a quando non si trova una soluzione intera. Inizialmente di risolve il rilassamento lineare, se questo fornisce una soluzione intera, essa sarà soluzione del problema originale. Diversamente, si aggiungono dfei vincoli opportuni alla regione ammissibile di (RL) che modificano il poliedreo ma non scartano soluzioni ammissibili intere. Questi vincoli eliminano invece la porzione di poliedro che contiene l'ottimo non intero del rilassamento lineare.

Definizione 1. *Una diseguaglianza $a^T x \leq a_0$ si dice taglio se*

- *è soddisfatta da tutti gli elementi di X, cioè $a^T x \leq a_0$, $\forall x \in X$ (diseguaglianza valida per X);*
- *non è soddisfatta dall'ottimo del rilassamento lineare x^*, cioè $a^T x^* > a_0$.*

Il metodo dei tagli procede dunque così:

1. Si risolve il problema lineare rilassando i vincoli di interezza.
2. Se la soluzione è intera, STOP
3. Altrimenti, si aggiunge un nuovo vincolo (taglio)
4. Si risolve il nuovo problema di PL e si torna al passo 2.

I metodi di taglio costituiscono una famiglia di metodi, che differiscono dal modo in cui vengono generati i tagli.

1.2 Piani di taglio di Gomory

Il metodo dei piani di taglio determina la soluzione del problema (PLI) risolvendo una successione di problemi di PL generati in modo progressivo fino ad ottenerne uno la cui soluzione continua rispetti il vincolo di interezza. Si opera progressivamente una sequenza di tagli della regione ammissibile del problema continuo, che elimini la soluzione ottima non intera corrente, ma non soluzioni ammissibili intere, fino a che la soluzione ottima continua sia intera.

I piani di taglio di Gomory sfruttano le informazioni contenute nella tabella finale del rilassamento lineare. Se la soluzione del rilassamento lineare non è intera allora avrà qualche componente frazionaria. Sia β_r la componente frazionaria della variabile di base x_r .

La tabella finale del rilassamento lineare è:

x_1	...	x_r	...	x_m	x_{m+1}	...	x_j	...	x_n	β
1	...	0	...	0	$\alpha_{1,m+1}$	...	α_{1j}	...	α_{1n}	β_1
0	...	0	...	0	$\alpha_{2,m+1}$	...	α_{2j}	...	α_{2n}	β_2
.	.	.	.	.	.	.	.	.		
0	...	1	...	0	$\alpha_{r,m+1}$	...	α_{rj}	...	α_{rn}	$\beta_r < 0$
.	.	.	.	.	.	.	.	.		
0	...	0	...	1	$\alpha_{m,m+1}$	...	α_{mj}	...	α_{mn}	β_m
0	...	0	...	0	r_{m+1}	...	r_j	...	r_n	$-z_0$

Se ci fossero più variabili in base con valore frazionario, si può scegliere il vincolo il cui termine noto ha la parte frazionaria maggiore. Se $x_{j+1}, \ldots, x_n$ sono le variabili fuori base, la riga r-esima è

$$x_{\nu_r} + \alpha_{r,j+1}x_{j+1} + \ldots + \alpha_{rn}x_n = \beta_r. \tag{1}$$

Sostituendo ogni termine α_{rj} con la corrispondente parte intera $\lfloor \alpha_{rj} \rfloor$ si ha

$$x_{\nu_r} + \lfloor \alpha_{r,j+1} \rfloor x_{j+1} + \ldots + \lfloor \alpha_{rn} \rfloor x_n \leq \beta_r. \tag{2}$$

Poichè la soluzione ottima del problema deve essere intera e tutti i coefficienti della (2) sono interi, si può scrivere

$$x_{\nu_r} + \lfloor \alpha_{r,m+1} \rfloor x_{m+1} + \ldots + \lfloor \alpha_{rn} \rfloor x_n \leq \lfloor \beta_r \rfloor. \tag{3}$$

La (3) è una diseguaglianza valida. Inoltre, la soluzione ottima del rilassamento continuo non soddisfa la (3). È quindi un piano di taglio, ed è il nuovo vincolo da aggiungere a (RL). Questo taglio è detto **taglio intero**. Si rende standard il taglio inserendo una variabile scarto e si aggiuge anche la colonna associata alla variabile x_{n+1}. La nuova base è x_1, x_{n+1}. I costi ridotti da $m+1$ a n restano invariati, mentre il nuovo costo ridotto $\bar{c}_{n+1}$ vale zero, perchè x_{n+1} è una variabile di base. Si aggiunge il taglio standardizzato alla forma standard del rilassamento lineare. Si risolve poi il nuovo problema continuo. Spesso è utile scrivere il taglio in funzione delle variabili originali del problema per non perdere la forma canonica.

Osservazione 1. *Il metodo di Gomory converge lentamente. È possibile effettuare più tagli contemporaneamente per rendere il metodo più efficiente.*

È possibile generare un altro tipo di taglio detto **taglio frazionario** che si aggiuge direttamente alla tabella finale del rilassamento lineare.

Sia β_r la componente frazionaria della variabile di base x_r . La riga r-esima è

$$x_{\nu_r} + \alpha_{r,j+1}x_{j+1} + \ldots + \alpha_{rn}x_n = \beta_r. \tag{4}$$

Dato il taglio intero

$$x_{\nu_r} + \lfloor \alpha_{r,m+1} \rfloor x_{m+1} + \ldots + \lfloor \alpha_{rn} \rfloor x_n \leq \lfloor \beta_r \rfloor, \tag{5}$$

il taglio frazionario si ottiene sottraendo (5) a (4). Si ottiene

$$(\alpha_{r,j+1} - \lfloor \alpha_{r,j+1} \rfloor)x_{j+1} + \ldots + (\alpha_{rn} - \lfloor \alpha_{rn} \rfloor)x_n \geq \beta_r - \lfloor \beta_r \rfloor. \tag{6}$$

Il taglio va reso standard sottraendo una variabile surplus x_{n+1}, si ha

$$(\alpha_{r,j+1} - \lfloor \alpha_{r,j+1} \rfloor)x_{j+1} + \ldots + (\alpha_{rn} - \lfloor \alpha_{rn} \rfloor)x_n - x_{n+1} = \beta_r - \lfloor \beta_r \rfloor. \tag{7}$$

Infine, per avere a disposizione una base canonica, si moltiplica per -1 tutta l'espressione, ottenendo:

$$-(\alpha_{r,j+1} - \lfloor \alpha_{r,j+1} \rfloor)x_{j+1} - \ldots - (\alpha_{rn} - \lfloor \alpha_{rn} \rfloor)x_n + x_{n+1} = \lfloor \beta_r \rfloor - \beta_r. \tag{8}$$

Il taglio standardizzato si aggiunge alla tabella finale di (RL) e si aggiuge anche la colonna associata alla variabile x_{n+1}. I costi ridotti da $m+1$ a n restano invariati e il nuovo costo ridotto $\bar{c}_{n+1}$ vale zero, perchè x_{n+1} è una variabile di base. Dal momento che $\lfloor \beta_r \rfloor - \beta_r < 0$ si procede applicando il metodo del simplesso duale. Siamo infatti nel caso in cui c'è almeno un temine noto negativo e i costi ridotti sono tutti non negativi (ammissibilità della soluzione duale).

2 Branch and Bound

Il metodo del Branch and Bound è stato introdotto all'inizio degli anni '60 ed è ormai uno dei più noti e utilizzati metodi per risolvere problemi di programmazione lineare intera. Si ascrive alla classe dei metodi di enumerazione implicita ed è caratterizzato da uno schema ad albero che conserva la traccia dell'enumerazione e da una procedura di valutazione del nodo che tende ad individuare i rami che non possono contenere la soluzione ottima.

L'idea di base consiste nel partizionare l'insieme delle soluzioni ammissibili di un problema di programmazione lineare intera mediante una famiglia di insiemi, via via più ristretti. Per ciascuno di questi insiemi si determina una valutazione inferiore (lower bound) del valore ottimo della funzione obiettivo. Se il lower bound non è migliore del valore calcolato in una soluzione ammissibile e nota, allora l'insieme corrispondente è escluso da ulteriori indagini. L'algoritmo termina quando la migliore soluzione intera trovata ha un valore migliore del bound calcolato per ogni insieme rimanente. Il metodo fa uso di due procedure:

- bounding: si risolve un problema rilassato (in genere il rilassamento lineare) del problema intero che fornisce un lower bound per il valore ottimo del problema intero;
- braching: la soluzione del rilassamento lineare viene utilizzata per generare la partizione dell'insieme ammissibile e i sottoproblemi da risolvere.

Sia dato il problema

$$(P) \quad \begin{cases} \min c^T x \\ Ax = b \\ x \geq 0 \\ x \in \mathbb{Z}^n \end{cases}$$

e si consideri il rilassamento continuo

$$(\overline{P}) \quad \begin{cases} \min c^T x \\ Ax = b \\ x \geq 0 \\ x \in \mathbb{R}^n \end{cases}$$

Il valore ottimo di $(\overline{P})$ è un lower bound per il valore ottimo di (P). Se l'ottimo di $(\overline{P})$ è intero, esso è anche soluzione di (P) e non si procede ulteriormente. Se il valore ottimo x^* non è intero, si sceglie una variabile frazionaria x_h^* e si costruiscono i sottoproblemi

$$(P_1) \quad \begin{cases} \min c^T x \\ Ax = b \\ x \geq 0 \\ x \text{ intero} \\ x_h \leq \lfloor x_h^* \rfloor \end{cases} \qquad (P_2) \quad \begin{cases} \min c^T x \\ Ax = b \\ x \geq 0 \\ x \text{ intero} \\ x_h \geq \lceil x_h^* \rceil \end{cases}$$

dove $\lfloor x_h^* \rfloor$ e $\lceil x_h^* \rceil$ sono gli arrotondamenti per difetto e per eccesso di x_h^*. Il problema (P) è detto padre; i problemi (P_1), (P_2) sono detti figli; la variabile x_h è detta variabile di branch e la costruzione dei sottoptoblemi è detta operazione di branch. Ogni problema figlio è ottenuto dal problema padre aggiungendo ai vincoli iniziali il vincolo di branch. A questo punto, si risolvono i rilassamenti continui di (P_1) e (P_2) e se le soluzioni non sono intere si itera il procedimento. In questo modo si tende ad ottenere una successione gerarchica di sottoproblemi via via più vincolati e quindi più facili da risolvere. Il processo termina quando non ci sono più nodi su cui effettuare l'operazione di branch. Se il problema ha un numero finito di soluzioni, allora l'albero di ricerca è anch'esso finito. Ad esempio in un problema di PLI con n variabili 0/1, l'albero di ricerca avrà al più profondità n con 2^n nodi foglia.

2.1 Criteri di potatura

Inammissibilità della soluzione Se al nodo t il problema rilassato (PL_t) non è ammissibile, ad esempio il vincolo di branch non è compatibile con i vincoli iniziali, il nodo t si può chiudere.

Soluzione intera Se l'ottimo x^* del problema rilassato (PL_t) al nodo t è intero con costo $c^T x^*$, e se la migliore soluzione intera trovata x_{opt} ha costo $c^T x_{opt}$ con $c^T x^* < c^T x_{opt}$, allora x_{opt} si aggiorna con x^* e $c^T x_{opt}$ con $c^T x^*$ e il nodo si chiude.

Assenza di una soluzione migliorante Se l'ottimo x^* del problema rilassato non è intero con costo $c^T x^*$, risulta che $c^T x^*$ è un lower bound della migliore soluzione x_t^* che si ottiene a partire dal nodo t. Quindi se $c^T x^* \geq c^T x_{opt}$, dove x_{opt} è la migliore soluzione intera trovata fino a quel momento, allora il nodo si può chiudere.

I criteri di potatura permettono di evitare l'enumerazione esplicita di tutte le soluzioni. Migliorano inoltre l'efficienza del metodo:

- le regole di esplorazione dell'albero
- le regole di branch
- la procedura di calcolo del lower bound

2.2 Regole di esplorazione dell'albero. Depth first

Le regole di esplorazione dell'albero suggeriscono l'ordine di visita dei nodi dell'albero di ricerca. Le più comunemente usate sono la regola del depth first e quella del best first.

La regola del depth first esplora l'albero in profondità, tende a minimizzare il numero totale di sottoproblemi aperti ancora da esaminare e dunque lo spazio di memoria. Più in particolare, se un nodo non è chiuso, il nodo successivo è il primo dei suoi figli. Se è chiuso, si risale sull'albero verso la radice fino a trovare un nodo che abbia un figlio che non sia ancora stato esplorato.

2.3 Regole di esplorazione dell'albero. Best first

La regola del best first seleziona di volta in volta il nodo migliore, in genere quello che ha il minor lower bound. Aumenta così il numero dei sottoproblemi aperti ancora da esaminare, ma diminuisce il numero totale di nodi esaminati. Inoltre se un nodo t genera una soluzione continua con costo maggiore o uguale a quello relativo alla migliore soluzione intera trovata, allora quest'ultima è la soluzione ottima e il nodo t si chiude.

2.4 Regole di branch

Le regole di branch tendono a partizionare l'insieme delle soluzioni ammissibili S in sottoinsiemi S_i in modo che $S = \cup_i S_i$ e $S_i \cap S_j = \emptyset$ se $i \neq j$ (quest'ultima relazione non è strettamente necessaria). La soluzione ottima appartiene ad uno degli insiemi S_i e ogni insieme S_i è ottenuto aggiungendo ad S qualche vincolo. Esistono fondamentalmente due regole di branch: binaria ed ennaria. Con la regola binaria, da ogni nodo padre vengono generati due nodi figli. Con la regola ennaria, ogni nodo padre può generare più di due figli.

2.5 Calcolo del lower bound

Uno dei passi fondamentali del processo risolutivo di un problema di PLI è il calcolo di valutazioni inferiori del valore ottimo della funzione obiettivo. Ciò permette di stimare la qualità delle soluzioni ammissibili che si hanno a disposizione. Le procedure di calcolo del lower bound sono essenzialmente quattro:

- rilassamento per eliminazione dei vincoli
- funzione obiettivo modificata
- rilassamento Lagrangiano
- rilassamento surrogato (poco utilizzato)

2.5.1 Rilassamento per eliminazione dei vincoli

Sia dato il problema

$$P_0: \quad \min_{x \in X} f(x) = z_0,$$

dove X è il politopo delle soluzioni ammissibili. Si consideri $Y \supseteq X$ ottenuto eliminando uno o più vincoli del problema P_0 e si supponga che il problema rilassato

$$P_1: \quad \min_{x \in Y} f(x) = z_1,$$

sia più semplice da risolvere. Allora z_1 costituisce una valutazione inferiore del valore ottimo del problema iniziale. Il più noto rilassamento continuo si ottiene eliminando il vincolo di interezza.

2.5.2 Funzione obiettivo modificata

Sia dato il problema

$$P_0: \quad \min_{x \in X} f(x) = z_0$$

e sia $\phi(x)$ una funzione tale che

$$\phi(x) \leq f(x), \forall x \in X.$$

Allora il valore ottimo del problema

$$\min_{x \in X} \phi(x) = w$$

rappresenta un lower bound di z_0.

2.5.3 Rilassamento Lagrangiano

Combinando i due approcci precedenti si ottiene il rilassamento Lagrangiano. Sia dato il problema

$$P_0: \quad \min_{x \in X} f(x) = z_0,$$

con

$$X = \{x \in Y : g_i(x) \geq 0, i = 1, \ldots, r\},$$

dove Y è l'insieme dei vincoli facili e siano $g_i(x)$ i restanti vincoli che rendono il problema P_0 difficile da risolvere. Il *rilassamento Lagrangiano* consiste nel costruire la *funzione Lagrangiana*

$$L(x, \lambda) = f(x) - \lambda^T g(x),$$

dove $\lambda \in \mathbb{R}^r_+$ rappresenta il vettore dei moltiplicatori di Lagrange. Si minimizza quindi il problema

$$(LP) \quad \begin{cases} \min L(x, \lambda) = LP_0(\lambda) \\ x \in Y \\ \lambda \geq 0 \end{cases}$$

Poichè $\lambda \geq 0$ e $g_i(x) \geq 0$, si ha che $LP_0(\lambda) \leq z_0$. In altre parole, $LP_0(\lambda)$, $\lambda \geq 0$ rappresenta un lower bound per P_0. Per ottenere il lower bound migliore (più vicino a z_0), occorre risolvere il problema *Lagrangiano duale*

$$\max_{\lambda \geq 0} LP_0(\lambda) = LD_0,$$

dove x è un parametro e λ è il vettore delle variabili del problema.

Osservazione 2. *L'ottimo del rilassamento continuo è minore o uguale dell'ottimo del problema duale. In altre parole, l'ottimo duale è un lower bound più stringente per il problema intero.*

2.5.4 Calcolo dell'upper bound

Il valore della funzione obiettivo calcolata in un qualsiasi punto ammissibile costituisce un upper bound del valore ottimo. Se si ha a disposizione un'euristica, si valuta la funzione nel risultato di tale euristica. Diversamente, durante il calcolo dei vari lower bound si possono individuare elementi di S in cui valutare la funzione obiettivo. Ad esempio, se si calcola il lower bound tramite un rilassamento continuo, e la soluzione di quest'ultimo è intera, allora si può valutare la funzione in tale punto. In altri casi, si può arrotondare o approssimare per eccesso o per difetto il valore di una soluzione.

2.5.5 Problemi di massimo

- Per ottenere le valutazioni superiori (upper bound), si risolvono i problemi associati ai nodi dell'albero.
- Per avere il lower bound, si calcola la funzione in un valore ammissibile e noto, eventualmente utilizzando un'euristica.
- Per la regola del best first si sceglie il problema con l'upper bound maggiore.
- Un nodo viene chiuso se l'upper bound è minore o uguale del lower bound.

3 Metodo del Branch and Bound per il problema dello zaino

Consideriamo il problema

$$\begin{cases} \max \sum_{i=1}^{n} v_i x_i \\ \sum_{i=1}^{n} p_i x_i \leq b \\ x_i \in \{0,1\},\ \forall i \in \{1,\dots,n\}. \end{cases}$$

Per applicare il metodo del Branch and Bound occorre vedere come:

- calcolare un upper bound
- effettuare l'operazione di branch
- individuare soluzioni ammissibili

Assumiamo come regola di esplorazione dell'albero la strategia best first e come regola di branch quella binaria, attribuendo ad ogni variabile il valore 0 oppure 1.

3.1 Calcolo dell'upper bound

Risolviamo il rilassamento continuo

$$\begin{cases} \max \sum_{i=1}^{n} v_i x_i \\ \sum_{i=1}^{n} p_i x_i \leq b \\ 0 \leq x_i \leq 1,\ \forall i \in \{1,\dots,n\}. \end{cases}$$

La soluzione può essere ottenuta senza usare il metodo del simplesso. Si riordinino gli oggetti secondo i rapporti valore/peso decrescenti in modo da avere

$$\frac{v_1}{p_1} \geq \frac{v_2}{p_2} \geq \cdots \geq \frac{v_n}{p_n}.$$

Si determini l'indice h tale che

$$\sum_{i=1}^{h} p_i x_i \leq b, \quad \sum_{i=1}^{h+1} p_i x_i > b$$

con $h = 0$ se $p_1 > b$.

La soluzione del rilassamento continuo è allora

$$\begin{aligned} x_1 &= x_2 = \cdots = x_h = 1 \\ x_{h+1} &= \frac{b - \sum_{i=1}^{h} p_i x_i}{p_{h+1}} \\ x_{h+2} &= x_{h+3} = \cdots = x_n = 0. \end{aligned}$$

ed il valore ottimo è

$$\sum_{i=1}^{h} v_i + v_{h+1} \frac{b - \sum_{i=1}^{h} p_i x_i}{p_{h+1}}$$

La variabile x_{h+1} è detta variabile critica. La ricerca della variabile critica e dell'upper bound possono essere effettuati in tempo lineare. Un upper bound migliore è dato da:

$$\begin{aligned} UB &= \sum_{i=1}^{h} v_i + \max\left\{ \left\lfloor \left(b - \sum_{i=1}^{h} p_i\right) \frac{v_{h+2}}{p_{h+2}} \right\rfloor, \right. \\ &\quad \left. \left\lfloor v_{h+1} - \left(p_{h+1} - \left(b - \sum_{i=1}^{h} p_i\right)\right) \frac{v_h}{p_h} \right\rfloor \right\} \end{aligned}$$

3.2 Soluzione ammissibile

La soluzione

$$\begin{aligned} &x_1 = x_2 = \cdots = x_h = 1 \\ &x_{h+1} = x_{h+2} = \cdots = x_n = 0 \end{aligned}$$

ottenuta approssimando per difetto il valore dell'unica variabile (possibilmente) non intera della soluzione del rilassamento continuo, è ammissibile per S. Pertanto tale soluzione può essere utilizzata per il calcolo del lower bound.

4 Problema dell'assegnamento di costo minimo

Si considerino due insiemi disgiunti e le coppie (i, j), composte da un elemento del primo insieme ed uno del secondo. Si associ ad ogni coppia un costo c_{ij}. Si vuole assegnare ad ogni elemento del primo insieme uno ed un solo elemento del secondo insieme, facendo in modo che nessun elemento risulti non accoppiato e che il costo totale sia minimizzato.

$$\begin{cases} \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{j=1}^{n} x_{ij} = 1,\ i = 1, \ldots, n \\ \sum_{i=1}^{n} x_{ij} = 1,\ j = 1, \ldots, n \\ x_{ij} \in \{0, 1\},\ \forall i, j. \end{cases}$$

Dai vincoli si deduce che le variabili possono assumere solo i valori 0 e 1.

$$\begin{cases} \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{j=1}^{n} x_{ij} = 1,\ i = 1, \ldots, n \\ \sum_{i=1}^{n} x_{ij} = 1,\ j = 1, \ldots, n \\ x_{ij} \geq 0,\ \forall i, j. \end{cases}$$

Si potrebbe risolvere il problema con l'algoritmo del simplesso, ma i vertici del problema sono fortemente degeneri. Infatti dai vincoli del problema segue che solo n variabili possono assumere valore 1, inoltre una base ha $2n-1$ variabili. Di conseguenza una soluzione di base, dovendo avere $n-1$ variabili in base nulle è fortemente degenere. Risulta quindi più conveniente utilizzare un algoritmo specifico.

4.1 Risoluzione mediante l'algoritmo ungherese

I passi dell'algoritmo sono:

Passo 1 Si analizzi la matrice dei costi C e si calcoli per ogni riga il valore minimo. Si aggiornino gli elementi sottraendo l'elemento minimo ad ogni elemento della riga. Si ponga

$$c_{ij} = c_{ij} - \min_j c_{ij}, \quad i = 1, \ldots, n.$$

Passo 2 Si calcoli il valore minimo per ogni colonna e lo si sottragga agli elementi della colonna. Si ponga:

$$c_{ij} = c_{ij} - \min_i c_{ij}, \quad j = 1, \ldots, n.$$

Passo 3 Sia $G = (N_1, N_2, A)$ il grafo bipartito, in cui N_1 e N_2 corrispondono, rispettivamente, alle righe e alle colonne della matrice C e A sia formato dalle coppie (i, j) con $c_{ij} = 0$. Si cerchino n accoppiamenti di costo nullo, uno per ogni riga ed uno per ogni colonna. Se esistono allora STOP.

Passo 4 Si aggiorni la matrice C con la procedura P e si torni al Passo 3.

4.1.1 Procedura P

1. Considerare la matrice C con le righe e le colonne non barrate.
2. Etichettare le righe che non sono state accoppiate dall'algoritmo di ricerca del massimo matching.
3. Etichettare le colonne che hanno degli zeri in corrispondenza delle righe etichettate.
4. Etichettare le righe, che sono state accoppiate dall'algoritmo di ricerca del massimo matching, con le colonne etichettate.
5. Ripetere 3 e 4 finchè non ci sono più righe o colonne da etichettare.
6. Barrare le righe non etichettate e le colonne etichettate.
7. Trovare il minimo δ tra gli elementi non barrati.
8. Aggiungere δ agli elementi barrati sia per riga che per colonna e sottrarlo a quelli non barrati.
9. La matrice C è così aggiornata.

5 Metodo del Branch and Bound per TSP asimmetrico

Consideriamo il problema del commesso viaggiatore (TSP).

Un commesso viaggiatore deve determinare l'itinerario che gli permette di visitare n città una ed una sola volta, in modo da avere un percorso di lunghezza totale minima (di costo totale minimo). Si consideri il grafo $G = (N, A)$, dove N è l'insieme degli n nodi (città) ed A è l'insieme degli archi (strade). Si vuole determinare un circuito hamiltoniano (che tocchi ogni nodo una ed una sola volta) di costo minimo. Sia c_{ij} il costo associato all'arco (i, j) e sia x_{ij} la variabile che vale 1 se l'arco appartiene al circuito hamiltoniano e zero altrimenti.

$$\begin{cases} \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \text{(costo totale del circuito)} \\ \sum_{i:(i,v)\in A} x_{iv} = 1, \quad v \in N \\ \text{(in ogni nodo entra un solo arco)} \\ \sum_{j:(v,j)\in A} x_{vj} = 1, \quad v \in N \\ \text{(da ogni nodo esce uno ed un solo arco)} \\ \sum_{i,j\in S,(i,j)\in A} x_{ij} \leq 1, \quad \forall S : S \subset N,\ 2 \leq |S| \leq n-1 \\ \text{(assenza di sottocircuiti)} \\ x_{ij} \in \{0, 1\},\ \forall j \in \{1, \ldots, n\}, i \in \{1, \ldots, m\}. \end{cases}$$

Per applicare il metodo del Branch and Bound occorre vedere come:

- calcolare un lower bound
- effettuare l'operazione di branch
- individuare soluzioni ammissibili

Assumiamo come regola di esplorazione dell'albero la strategia best first.

5.1 Calcolo del lower bound

Si effettua un rilassamento rispetto al vincolo sui sottocircuiti:

$$\begin{cases} \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1, \quad j \in N \\ \sum_{\substack{j=1 \\ i \neq j}}^{n} x_{ij} = 1, \quad i \in N \\ x_{ij} \in \{0,1\}, \ \forall i,j \in \{1,\dots,n\}, i \neq j. \end{cases}$$

Tale problema si riconduce al problema dell'assegnamento, associando al grafo iniziale un grafo bipartito $G = (N_1 \cup N_2, A')$, dove N_1 ed N_2 si ottengono duplicando i nodi di N. Pertanto si ha $|N_1| = |N_2|$. Infine, per rendere il grafo bipartito completo, si aggiungono gli archi (i,i) di costo pari a $+\infty$.

Si risolve quindi il problema dell'assegnamento

$$\begin{cases} \min \sum_{i \in N_1}^{n} \sum_{j \in N_2} c_{ij} x_{ij} \\ \sum_{i \in N_1}^{n} x_{ij} = 1, \quad j \in N_2 \\ \sum_{j=1 \in N_2}^{n} x_{ij} = 1, \quad i \in N_1 \\ x_{ij} \in \{0,1\}, \ \forall i,j \in \{1,\dots,n\}. \end{cases}$$

5.2 Operazione di branching

È possibile utilizzare una regola di branch non binario che sfrutta i vincoli sui sottocircuiti. Data una soluzione del problema dell'assegnamento, si osserva che non essendo ottima per il problema del TSP, deve contenere dei sottocicli. Sia A_S il sottociclo di cardinalità minima. Per eliminiare il sottociclo, occorrerebbe che x_{ij}, $(i,j) \in A_S$ fosse nulla. Si generano allora nodi figli in numero pari alla cardinalità di A_S. In corrispondenza di ogni arco $(i,j) \in A_S$ si genera un problema figlio con il vincolo aggiuntivo $x_{ij} = 0$, cioè $c_{ij} = \infty$. Più esattamente, sia $A_S = \{(i_1,j_1),(i_2,j_2),\dots(i_h,j_h)\}$. Il primo nodo figlio viene generato imponendo che non sia presente l'arco (i_1,j_1) (cioè $c_{i_1,j_1} = +\infty$), il secondo nodo viene ottenuto imponendo che sia presente l'arco (i_1,j_1), ma non lo sia l'arco (i_2,j_2) (cioè $c_{i_2,j_2} = +\infty$) e così continuando fino all'ultimo nodo figlio, che si ottiene imponendo che siano presenti tutti gli archi tranne l'arco (i_h,j_h).

5.3 Soluzione ammissibile

L'algoritmo del nodo più vicino (Nearest Neighbour) costruisce un cammino contenente tutti i nodi, aggiungendo ad ogni iterazione il nodo che produce il minimo incremento di costo. Unendo poi gli estremi del cammino trovato, si ottiene un circuito hamiltoniano.

I passi dell'algoritmo sono:

Passo 1 Si fissi un nodo di partenza i e lo si inserisca in un insieme U inizialmente vuoto. Si ponga $r = i$.

Passo 2 Si consideri il nodo $s \in N \setminus U$ tale che

$$d_{rs} = \min_{j \in N \setminus U} d_{rj},$$

dove d_{ij} denota la distanza lungo l'arco (i,j). Si faccia seguire r da s nel circuito.

Passo 3 Si ponga $U = U \cup \{s\}$ e $r = s$. Se $U = N$ STOP si chiude il circuito andando da s al nodo iniziale i. Altrimenti si ritorni al Passo 2.