Metodi e Modelli per l'Ottimizzazione Combinatoria

Metodi esatti per il problema del commesso viaggiatore

L. De Giovanni M. Di Summa

Il problema del commesso viaggiatore (Travelling Salesman Problem - TSP) è un problema di ottimizzazione definito su un grafo, che può essere orientato o non orientato. Nel caso di un grafo orientato $D = (N, A)$, si considerano definiti dei costi c_{ij} associati ad ogni arco $(i, j) \in A$. Nel caso di un grafo non orientato $G = (V, E)$, si considerano dei costi c_e associati a ciascuno spigolo $e \in E$. Il TSP consiste nel determinare, se esiste, un *ciclo hamiltoniano* di costo minimo sul grafo. Ricordiamo che un ciclo hamiltoniano è un ciclo che visita tutti i nodi del grafo una ed una sola volta. Si fa notare che possiamo supporre, senza perdita di generalità, che il grafo sia completo, ossia che contenga archi / spigoli per ogni coppia ordinata / non ordinata di nodi: basta porre a $+\infty$ il costo sugli archi / spigoli non presenti.[1] I modelli e le tecniche risolutive possono essere specializzate per i due casi di grafo orientato (TSP asimmetrico) o non orientato (TSP simmetrico). Trattiamo pertanto i due casi separatamente.[2] In ogni caso, ricordiamo che il TSP è un problema $\mathcal{NP}$-hard e pertanto la sua soluzione esatta richiede degli algoritmi a complessità "esponenziale (a meno che $\mathcal{P} = \mathcal{NP}$).

1 TSP asimmetrico

Il TSP asimmetrico (*Asymmetric TSP - ATSP)* è definito su un grafo orientato $D = (N, A)$ con costi c_{ij} associati a ciascun arco $(i, j) \in A$. Un modello del problema considera delle variabili decisionali associate ad ogni arco:

$$\forall\, (i,j) \in A,\ x_{ij} = \begin{cases} 1, & \text{se } (i,j) \text{ è nel ciclo;} \\ 0, & \text{altrimenti.} \end{cases}$$

[1] Si può anche imporre un costo pari a quello del cammino minino tra i due nodi.

[2] In realtà, i metodi che presenteremo possono essere adattati ed utilizzati per entrambi i casi: in ogni sezione si presentano metodi che, dal punto di vista dell'efficienza computazionale, sono più adatti al caso asimmetrico o simmetrico.

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} & & & & \\
s.t. \quad & \sum_{i\in N:(i,v)\in A} x_{iv} & = & 1 & \forall & \; v \in N \\
& \sum_{j\in N:(v,j)\in A} x_{vj} & = & 1 & \forall & \; v \in N \\
& \sum_{i\in S, j\in S:(i,j)\in A} x_{ij} & \leq & |S|-1 & \forall & \; S \subset N : 2 \leq |S| \leq |N|-1 \\
& x_{ij} & \in & \{0,1\} & \forall & \; (i,j) \in A
\end{aligned}
$$

La funzione obiettivo minimizza la somma dei costi degli archi selezionati. Gli archi selezionati devono garantire di entrare una e una sola volta in ogni nodo (primo vincolo) e di uscire una e una sola volta da ogni nodo (secondo vincolo). In questo modo garantiamo di passare una e una sola volta da ogni nodo, ma sarebbero ammissibili anche soluzioni che contengono sotto-cicli. Il quarto gruppo di vincoli serve proprio per eliminare tutte le soluzioni che contengono sotto-cicli, visto che, per ogni sottoinsieme proprio dei nodi ($S : 2 \leq |S| \leq |N|-1$), viene impedito di selezionare un numero di archi *interni* a S stesso pari (o superiore) al numero di nodi in S. Tali vincoli sono detti *vincoli di eliminazione dei sotto-cicli* (*sub-tour elimination constraints*).

L'eliminazione dei sotto-cicli può essere ottenuta in modo equivalente sostituendo il quarto gruppo di vincoli con i seguenti (più espliciti, ma più numerosi):

$$\sum_{(i,j)\in C} x_{ij} \leq |C| - 1 \; \forall \, C : \; C \text{ è un sotto-ciclo in } G$$

Si fa notare che il numero di vincoli del modello è pari a $2|N|$ più il numero dei possibili sottoinsiemi o sotto-cicli, che è comunque *esponenziale*. Il modello, pertanto, non può essere utilizzato direttamente per la soluzione del problema, a meno di casi in cui il numero di nodi è piccolo e sia possibile esplicitare tutti i vincoli di eliminazione dei sotto-cicli. È necessario pertanto elaborare degli approcci risolutivi diversi rispetto a quelli visti finora, in grado di gestire un numero esponenziale di vincoli. L'idea di base è quella di non considerare tutti i vincoli di eliminazione dei sotto-cicli, ma di aggiungere al modello solo un sottoinsieme limitato di tali vincoli. I vincoli da aggiungere sono ricavati dinamicamente sulla base di un approccio iterativo. I due metodi che presenteremo di seguito si basano sulla seguente osservazione.

Eliminando dal modello i sub-tour elimination constraints, otteniamo esattamente il modello di un problema di assegnamento: si può immaginare un grafo bipartito con i nodi di N a sinistra e una loro replica a destra; il costo di assegnamento del nodo i a sinistra al nodo j a destra è il costo c_{ij} per andare da i a j; nella soluzione, $x_{ij} = 1$ indica che dopo il nodo i visito il nodo j. Quindi, rimuovendo *tutti* i vincoli di eliminazione dei sotto-cicli, il problema è risolvibile in modo efficiente (ad esempio, con il metodo del simplesso), viste le proprietà del problema dell'assegnamento. Chiaramente, la soluzione ottenuta non è in

generale ammissibile, visto che potrebbe contenere dei sotto-cicli. Potrebbe però essere anche ammissibile, visto che, come caso particolare, le x_{ij} relative a cicli hamiltoniani soddisfano sicuramente i vincoli di assegnamento. In ogni caso, il valore della soluzione rappresenterebbe un *lower bound* al valore della soluzione ottima ammissibile, visto che è stata ottenuta da un problema meno vincolato: la soluzione ottima del problema meno vincolato fornisce l'ottimo in un insieme di soluzioni che contiene, come caso particolare, le soluzioni con un solo ciclo hamiltoniano e, quindi, è chiaramente migliore o uguale della soluzione ottima tra i soli cicli hamiltoniani. Di conseguenza, se la soluzione ottenuta eliminando i vincoli non contenesse dei sotto-cicli, allora sarebbe la soluzione ottima. Lo stesso ragionamento vale se, invece di eliminare tutti i sub-tour elimination constraints, ne eliminassimo solo un loro sotto-insieme. Possiamo quindi affermare che:

Osservazione 1 *Se dal modello per il TSP eliminiamo tutti o parte dei sub-tour elimination constraints, allora la soluzione* ottima *del problema risultante potrebbe:*

- *contenere dei sotto-cicli: il valore della soluzione ottima è un lower-bound per il TSP;*
- *non contenere dei sotto-cicli: la soluzione è ammissibile e ottima per il TSP.*

Ora supponiamo che la soluzione ottima del problema di assegnamento non sia ammissibile per il TSP per la presenza di sotto-cicli: sia $\hat{C}$ uno dei sotto-cicli che comprende i nodi del sottoinsieme $\hat{S}$. Potremmo cercare di ottenere la soluzione ottima che non contenga $\hat{C}$, nella speranza che la nuova soluzione ottima sia anche ammissibile. Si tratta di aggiungere, almeno in linea di principio, il vincolo

$$\sum_{i\in\hat{S},j\in\hat{S}:(i,j)\in A} x_{ij} \leq |\hat{S}| - 1$$

oppure, equivalentemente

$$\sum_{(i,j)\in\hat{C}} x_{ij} \leq |\hat{C}| - 1$$

Ad esempio, se si individua il sotto-ciclo $1 \to 2 \to 3 \to 4 \to 1$, bisognerebbe introdurre il vincolo:

$$x_{12} + x_{21} + x_{13} + x_{31} + x_{14} + x_{41} + x_{23} + x_{32} + x_{24} + x_{42} + x_{34} + x_{43} \leq 3$$

oppure

$$x_{12} + x_{23} + x_{34} + x_{41} \leq 3$$

L'aggiunta del nuovo vincolo fa perdere, in generale, le proprietà strutturali (totale unimodularità) della matrice dei vincoli che non è più relativa ad un problema di assegnamento "puro". Pertanto il sotto-problema non è più risolvibile in modo efficiente (a meno di perderebbe l'interezza delle soluzioni e quindi la possibilità di ottenere una soluzione ammissibile per il TSP).

1.1 Generazione di vincoli per ATSP

In base alle osservazioni precedenti, si può ricavare un metodo risolutivo per il TSP corrispondente al metodo di *generazione di vincoli*:

1. Si eliminino tutti i sub-tour elimination constraints e i vincoli di interezza della variabili e si risolva il corrispondente problema di assegnamento.
2. *Si reintroducano i vincoli di interezza delle variabili.*
3. Se la soluzione corrente non contiene sotto-cicli allora STOP: soluzione ottima.
4. Si individuino uno (o più) sotto-cicli nella soluzione corrente.
5. Si aggiungano al modello i vincoli di eliminazione di almeno uno dei sotto-cicli individuati.
6. Si risolva il problema corrente e si torni al passo 3

La convergenza dell'algoritmo è garantita dal fatto che, al limite, si aggiungeranno *tutti* i sub-tour elimination constraints, ottenendo pertanto una soluzione ammissibile (e ottima) per il TSP. In realtà, la *ratio* che sottende l'algoritmo di generazione di vincoli è l'osservazione che non tutti i vincoli sono necessari per eliminare tutti i sotto-cicli. La realtà sperimentale mostra che si è in grado di ottenere un ciclo hamiltoniano con l'aggiunta di un sottoinsieme limitato dei sub-tour elimination constraints. Ovviamente, dal punto di vista della complessità computazionale, il numero di iterazioni nel caso peggiore resta pari al numero di possibili sotto-cicli.
Oltre all'elevato numero di iterazioni, l'algoritmo di generazione di vincoli per il TSP soffre di un altro problema computazionale: ad ogni iterazione bisogna risolvere un problema di programmazione lineare intera e quindi ogni singola iterazione ha complessità computazionale "esponenziale".
La situazione può essere migliorata perché è possibile dimostrare che, ai fini dell'interezza della soluzione del problema con l'aggiunta dei vincoli di sotto-ciclo, non è necessario che tutte le variabili siano vincolate ad essere intere: basta vincolare l'interezza delle variabili coinvolte nei vincoli via via aggiunti per garantire che anche le altre variabili assumano valori 0 o 1. L'algoritmo di generazione di vincoli per TSP sarebbe quindi il seguente:

Generazione di vincoli per TSP

1. Si eliminino tutti i sub-tour elimination constraints e i vincoli di interezza della variabili e si risolva il corrispondente problema di assegnamento.
2. Se la soluzione corrente non contiene sotto-cicli allora STOP: soluzione ottima.
3. Si individuino uno (o più) sotto-cicli nella soluzione corrente.
4. Si aggiungano al modello i vincoli di eliminazione di almeno uno dei sotto-cicli individuati e i vincoli di interezza per le variabili coinvolte nei vincoli.
5. Si risolva il problema corrente e si torni al passo 2

Ad ogni iterazione, quindi, si riesce a limitare il numero di variabili intere da gestire. Si avrebbe comunque un problema di programmazione lineare intera mista, la cui complessità computazionale è "esponenziale".

Esempio 1 *Si risolva il problema del commesso viaggiatore relativo alla seguente matrice delle distanze tra i nodi:*

	1	**2**	**3**	**4**	**5**	**6**
1	–	33.6	14.0	40.9	14.5	11.5
2	34.7	–	21.7	13.0	20.2	23.4
3	14.8	21.5	–	29.3	2.0	3.9
4	41.7	13.1	29.4	–	27.6	30.3
5	15.0	20.2	2.0	27.5	–	3.9
6	12.0	22.8	2.0	30.1	4.0	–

Risolviamo il problema con l'algoritmo per generazione di vincoli (il lettore può verificare i risultati con AMPL o altro software adeguato).

Inizializzazione
Eliminiamo i vincoli di sotto-ciclo e di interezza e, risolvendo il problema di assegnamento, otteniamo:

$$x_{16} = x_{61} = x_{24} = x_{42} = x_{35} = x_{53} = 1; \text{ costo } 53.6$$

Iterazione 1
Si verifica l'esistenza dei sotto-cicli

$$1 - 6 - 1 \qquad 2 - 4 - 2 \qquad 3 - 5 - 3$$

Si aggiungono i vincoli di eliminazione dei tre sotto-cicli:

$$x_{16} + x_{61} \leq 1$$
$$x_{24} + x_{42} \leq 1$$
$$x_{35} + x_{53} \leq 1$$

e di interezza

$$x_{16}, x_{61}, x_{24}, x_{42}, x_{35}, x_{53} \in \{0, 1\}$$

e si ottiene la soluzione:

$$x_{12} = x_{63} = x_{31} = x_{24} = x_{45} = x_{52} = 1; \text{ costo } 89.1$$

Iterazione 2
La nuova soluzione contiene i sotto-cicli

$$1 - 6 - 3 - 1 \qquad 2 - 4 - 5 - 2$$

Si aggiungono i vincoli di eliminazione dei due sotto-cicli:

$$x_{16} + x_{63} + x_{31} \leq 2$$
$$x_{24} + x_{45} + x_{52} \leq 2$$

e di interezza (i vincoli $x_{16}, x_{24} \in \{0, 1\}$ sono già presenti dall'iterazione precedente):

$$x_{63}, x_{31}, x_{45}, x_{52} \in \{0, 1\}$$

ottenendo la soluzione:

$$x_{16} = x_{63} = x_{35} = x_{52} = x_{24} = x_{41} = 1; \text{ costo } 90.4$$

Iterazione 3
La nuova soluzione non contiene sotto-cicli e corrisponde alla soluzione ottima del TSP:

$$1 - 6 - 3 - 5 - 2 - 4 - 1$$

1.2 Branch-and-bound per ATSP

Sulla base dell'Osservazione 1, possiamo sfruttare il lower-bound a disposizione per implementare un approccio risolutivo di tipo Branch-and-Bound. Al nodo radice si considera il problema completo e si eliminano tutti i vincoli di sotto-ciclo. Si risolve il problema risultante come un problema di assegnamento e si ottiene un lower-bound. Se la soluzione non contiene sotto-cicli allora la soluzione è ottima e non si sviluppano ulteriori nodi. Altrimenti, si fa branching generando dei sotto-problemi (nodi figli) le cui soluzioni ammissibili non contengano i sotto-cicli individuati. La procedura potrebbe essere ripetuta ricorsivamente per i nuovi nodi, ottenendo un Branch-and-Bound in grado di

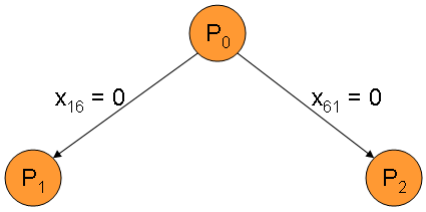
risolvere il TSP. Anche in questo caso, però, l'aggiunta dei vincoli di sotto-ciclo potrebbe farci perdere la totale unimodularità e la valutazione di ogni nodo del Branch-and-Bound diventerebbe quindi esponenziale. Ricordando che anche il numero dei nodi da valutare è, nel caso peggiore, esponenziale, si avrebbe di nuovo un metodo risolutivo molto poco efficiente.

Per migliorare la situazione, osserviamo che:

- per arrivare all'ammissibilità, non è necessario eliminare *tutti* i sotto-cicli individuati nella soluzione del problema associato ad un nodo: basta eliminare un sotto-ciclo per volta e (al costo di un maggior numero di nodi o iterazioni) si perverrà comunque ad eliminare tutti i sotto-cicli necessari;
- per eliminare la possibilità di ottenere un sotto-ciclo contenente l'arco (i, j) è sufficiente imporre il vincolo $x_{ij} = 0$ (non potendo utilizzare l'arco, non si può utilizzare neanche un ciclo che lo contiene).

Lo schema di branching può quindi essere precisato come segue: sia $\hat{C}$ *uno* dei sotto-cicli nella soluzione del problema di assegnamento (al nodo radice). Per ogni arco $(i, j) \in \hat{C}$ si generi un sotto-problema aggiungendo il vincolo $x_{ij} = 0$.

Se consideriamo l'Esempio 1, dopo aver risolto il problema di assegnamento relativo al nodo radice, otteniamo i sotto-cicli $1-6-1$, $2-4-2$ e $3-5-3$. Scegliamo il sotto-ciclo $1-6-1$ e facciamo branching secondo lo schema:



Facciamo notare che lo schema di branching:

- esclude sia da P_1 che da P_2 tutte le soluzioni contenenti il sotto-ciclo $1-6-1$;
- non perde soluzioni ammissibili: i cicli *hamiltoniani* che includono l'arco $(1, 6)$, esclusi da P_1, sono presenti in P_2; i cicli *hamiltoniani* che includono l'arco $(6, 1)$, esclusi da P_2, sono presenti in P_1;

I vantaggi dello schema di branching descritto derivano dalla semplicità dei vincoli che lo realizzano, che rendono possibile un'implementazione senza aggiunta di vincoli. Infatti

per imporre $x_{ij} = 0$ *basta porre* $c_{ij} = +\infty$.

In questo modo non è necessario aggiungere dei vincoli espliciti e la struttura dei vincoli rimane la stessa dell'assegnamento: sono semplicemente cambiati i costi della funzione obiettivo. Pertanto, ad ogni nodo, viene *preservata la totale unimodularità della matrice dei vincoli* del problema associato al nodo stesso e la valutazione del bound può essere fatta in modo efficiente (basta ad esempio, risolvere con il metodo del simplesso per preservare l'interezza delle variabili decisionali).

Nell'esempio, il lower bound del nodo P_1 è calcolato risolvendo il problema di assegnamento con i seguenti costi:

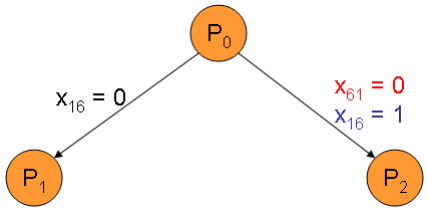
	1	**2**	**3**	**4**	**5**	**6**
1	–	33.6	14.0	40.9	14.5	$+\infty$
2	34.7	–	21.7	13.0	20.2	23.4
3	14.8	21.5	–	29.3	2.0	3.9
4	41.7	13.1	29.4	–	27.6	30.3
5	15.0	20.2	2.0	27.5	–	3.9
6	12.0	22.8	2.0	30.1	4.0	–

e, per il problema P_2, abbiamo

	1	**2**	**3**	**4**	**5**	**6**
1	–	33.6	14.0	40.9	14.5	11.5
2	34.7	–	21.7	13.0	20.2	23.4
3	14.8	21.5	–	29.3	2.0	3.9
4	41.7	13.1	29.4	–	27.6	30.3
5	15.0	20.2	2.0	27.5	–	3.9
6	$+\infty$	22.8	2.0	30.1	4.0	–

Si fa notare che i sottoinsiemi di soluzioni presenti in P_1 e P_2 *non sono disgiunti*. Nell'esempio, il ciclo hamiltoniano $1-2-3-4-6-5-1$ appartiene sia a P_1 che a P_2. Da un punto di vista concettuale, questo non rappresenta un problema per il Branch-and-Bound che deve semplicemente garantire di non perdere soluzioni ammissibili in fase di branching. L'efficienza computazionale, però, potrebbe risentirne: una stessa soluzione potrebbe trovarsi in più nodi e il numero di nodi da esplorare tende ad aumentare.

Per limitare la sovrapposizione dei nodi generati, possiamo imporre, nel nodo P_2 dell'esempio, $x_{16} = 1$: le soluzioni con $x_{16} = 0$ sono già considerate in P_1 ed è inutile riconsiderarle in P_2. Si ottiene quindi:



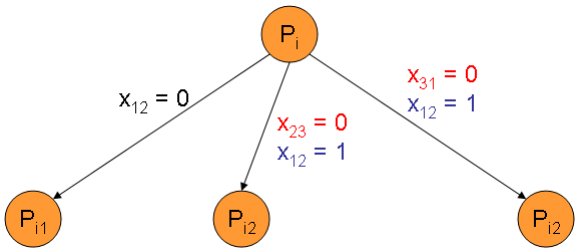
Anche l'imposizione del valore di una variabile a 1 può essere ottenuta senza intervenire sulla matrice dei vincoli, ma fissando i costi degli archi in modo da forzare l'uso dell'arco voluto:

per imporre $x_{ij} = 1$ *basta porre* $c_{ij} = +\infty$, $\forall\, j \neq i$.

Nell'esempio, il lower bound al nodo P_2 si ottiene risolvendo il problema di assegnamento con costi:

	1	**2**	**3**	**4**	**5**	**6**
1	–	$+\infty$	$+\infty$	$+\infty$	$+\infty$	11.5
2	34.7	–	21.7	13.0	20.2	23.4
3	14.8	21.5	–	29.3	2.0	3.9
4	41.7	13.1	29.4	–	27.6	30.3
5	15.0	20.2	2.0	27.5	–	3.9
6	$+\infty$	22.8	2.0	30.1	4.0	–

Ribadiamo che il nuovo schema di branching *limita* le sovrapposizioni ma non le esclude del tutto. Ad esempio, se ad un nodo P_i si ottenesse il sotto-ciclo $1-2-3-1$, si avrebbe:



e le soluzioni ammissibili con $x_{23} = x_{31} = 0$ sono presenti sia in P_{i_2} sia in P_{i_3}. Per eliminare ogni possibile sovrapposizione, in questo esempio è possibile generare i nodi $P_{i_1}, P_{i_2}, P_{i_3}$ come segue: in P_{i_1} imponiamo $x_{12} = 0$; in P_{i_2} imponiamo $x_{12} = 1$ e $x_{23} = 0$; in P_{i_3} imponiamo $x_{12} = x_{23} = 1$ e $x_{31} = 0$. In questo modo le regioni ammissibili dei tre sottoproblemi formano una partizione della regione ammissibile del problema originale. Si tenga però presente che sebbene il fatto di evitare sovrapposizioni possa essere un vantaggio, d'altra parte con questa tecnica si tende a costruire un albero più sbilanciato (ci sono più soluzioni nel primo sottoproblema che nell'ultimo), il che potrebbe dare problemi dal punto di vista computazionale.

Esempio 2 *Si applichi il metodo del Branch-and-Bound come sopra descritto per risolvere il TSP dell'Esempio 1.*

Con riferimento alla Figura 1:
$\underline{P_0}$: soluzione $1-6-1$, $2-4-2$, $3-5-3$ di costo 53.4.

Scelgo il sotto-ciclo $1-6-1$ per il branching generando:

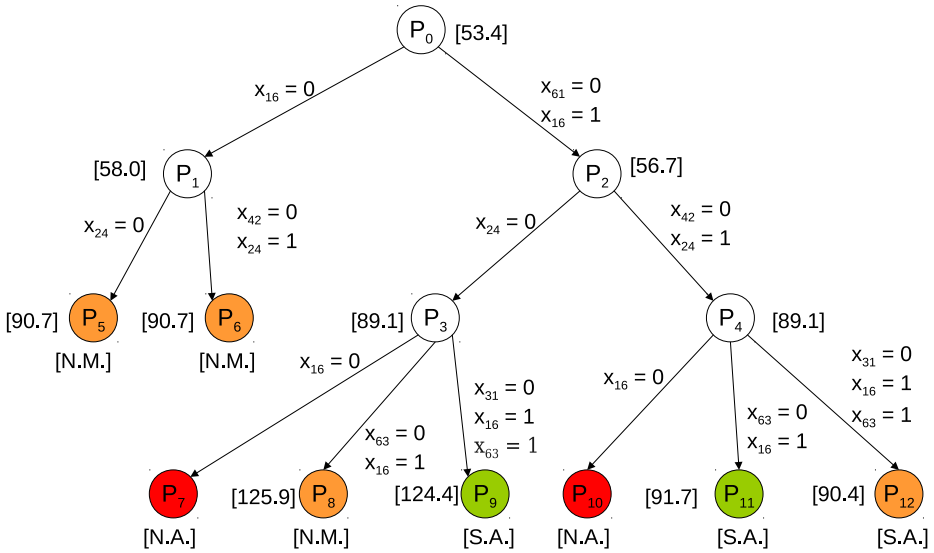


Figura 1: Albero del Branch-and-Bound per l'Esempio 1.

$\underline{P_1}$: soluzione $1-3-5-6-1$, $2-4-2$ di costo 58.0;

$\underline{P_2}$: soluzione $1-6-3-5-1$, $2-4-2$ di costo 56.6.

Applico la strategia di esplorazione *Best node* e faccio branching su P_2, scegliendo il sotto-ciclo $2-4-2$ (scelgo il più piccolo per aprire meno nodi), ottenendo:

$\underline{P_3}$: soluzione $1-6-3-1$, $2-5-4-2$ di costo 89.1;

$\underline{P_4}$: soluzione $1-6-3-1$, $2-4-5-2$ di costo 89.1.

Adesso il best node è P_2 e, scegliendo il sotto-ciclo $2-4-2$, genero i figli:

$\underline{P_5}$: soluzione $1-3-6-1$, $2-5-4-2$ di costo 90.7;

$\underline{P_6}$: soluzione $1-3-6-1$, $2-4-5-2$ di costo 90.7.

Sviluppo P_3 considerando il sotto-ciclo $1-6-3-1$:

$\underline{P_7}$: non ammissibile (i vincoli $x_{16}=1$ e $x_{16}=0$ sono in conflitto);

$\underline{P_8}$: soluzione $1-6-2-3-1$, $4-5-4$ di costo 125.9;

$\underline{P_9}$: soluzione $1-6-3-2-5-4-1$, di costo 124.4. La soluzione è anche ammissibile: $1-6-3-2-5-4-1$ diventa la soluzione corrente, P_9 viene chiuso così come P_8 perché non migliorante.

Rimangono aperti P_5, P_6 e P_4 e il best nodo è P_4, con sotto-ciclo $1-6-3-1$ e figli:

$\underline{P_{10}}$: non ammissibile (i vincoli $x_{16}=1$ e $x_{16}=0$ sono in conflitto);

$\underline{P_{11}}$: soluzione $1-6-2-4-5-3-1$, di costo 91.7. La soluzione è anche ammissibile: $1-6-2-4-5-3-1$ migliora la soluzione corrente e P_{11} viene chiuso, ma non permette di chiudere altri nodi.

$\underline{P_{12}}$: soluzione $1-6-3-5-2-4-1$, di costo 90.4. La soluzione è anche ammissibile: $1-6-3-5-2-4-1$ migliora la soluzione corrente e P_{12} viene chiuso, così come P_5 e P_6 perché non miglioranti.

Non essendo rimasti altri nodi aperti, $1-6-3-5-2-4-1$ è la soluzione ottima, di costo 90.4.

2 TSP simmetrico

Il TSP simmetrico (Symmetric TSP - STSP) è definito su un grafo non orientato $G = (V, E)$ con costi c_e associati ad ogni spigolo $e \in E$. Il problema può essere formulato e risolto come un TSP asimmetrico trasformando G in un grafo orientato con due archi per ogni spigolo: per ogni spigolo $e \in E$ di estremi $i \in V$ e $j \in V$ si considerano due archi (i, j) e (j, i) con $c_{ij} = c_{ji} = c_e$. La simmetria nei costi, però, rende il modello e i metodi visti migliorabili, non essendo più rilevante l'ordine in cui si visitano coppie di nodi (il costo è uguale). Una formulazione alternativa per il TSP simmetrico utilizza infatti delle variabili per ogni spigolo (quindi la metà delle variabili che verrebbero usate in caso di passaggio al corrispondente grafo simmetrico):

$$\forall\, e \in E,\ x_e = \begin{cases} 1, & \text{se } e \text{ è nel ciclo;} \\ 0, & \text{altrimenti.} \end{cases}$$

$$\begin{array}{lrcll} \min & \displaystyle\sum_{e\in E} x_e & & & \\ s.t. & \displaystyle\sum_{e\in\delta(v)} x_e & = & 2 \quad \forall\ v \in V & (STSP:1) \\ & \displaystyle\sum_{e\in E(S)} x_e & \leq & |S|-1 \quad \forall\ S \subset V : 3 \leq |S| \leq |V|-1 & (STSP:2) \\ & x_e & \in & \{0,1\} \quad \forall\ e \in E & (STSP:3) \end{array}$$

Si noti come, per ogni nodo $v \in V$, sia sufficiente scegliere esattamente due spigoli in $\delta(v)$, cioè la stella di spigoli che hanno v come estremo, senza necessità di esplicitare il verso di percorrenza dei singoli spigoli. Il secondo gruppo di vincoli equivale all'eliminazione di tutti i sotto-cicli: $E(S) \subseteq E$ è l'insieme degli spigoli che hanno entrambi gli estremi in $S \subset V$. Rispetto al caso asimmetrico, non è necessario eliminare i sotto-cicli di ordine 2, che sono esclusi dal primo insieme di vincoli. Anche in questo caso, la messa a punto di tecniche risolutive per STSP deve considerare la presenza di un numero esponenziale di vincoli.

2.1 Generazione di vincoli per STSP

Come per l'ATSP, si può pensare di risolvere iterativamente un problema rilassato, con un numero limitato di vincoli di eliminazione di sotto-ciclo. Dopo aver risolto il problema

rilassato, si individuano gli eventuali sotto-cicli nella soluzione ottima, si aggiungono i vincoli per l'eliminazione dei sotto-cicli individuati e si itera, fino ad ottenere una soluzione ottima che non contiene sotto-cicli ed è quindi ammissibile e ottima per il STSP. La procedura è riportata in Figura 2. Si noti che *la procedura è del tutto simile al caso ATSP*.

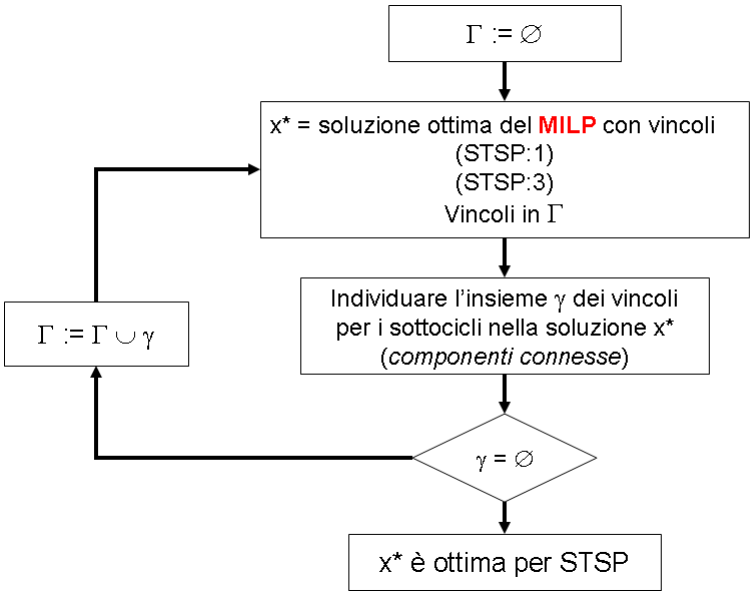


Figura 2: Generazione di vincoli per (S)TSP.

Si noti come il problema che restituisce x^* contiene i vincoli di interezza delle variabili ed è pertanto un problema di programmazione lineare intera. In realtà, come per il caso asimmetrico, è possibile limitare i vincoli di interezza (binarietà) delle variabili alle sole variabili nei vincoli di sotto-ciclo (le restanti variabili saranno automaticamente binarie). Pertanto, x^* è ottenibile con la soluzione di un problema di programmazione lineare intera mista (*Mixed Integer Linear Program* - MILP).

Assumendo che $x^* \in \{0,1\}^{|E|}$, l'individuazione dei sotto-cicli è molto semplice e consiste nell'individuazione delle componenti connesse nel grafo $G_{x^*} = (V, E_{x^*})$, cioè il grafo con i soli spigoli corrispondenti a $x_e > 0$. G_{x^*} è detto grafo di supporto di x^*.

Anche in questo caso, però, la procedura complessiva risulta molto poco efficiente, e di scarso aiuto se il problema è definito su un grafo con un numero molto elevato di nodi. In effetti, il numero di iterazioni è esponenziale (nel caso peggiore si dovranno aggiungere i vincoli di eliminazione di tutti i sotto-cicli) e ad ogni iterazione si deve risolvere un MILP!

Come sopra osservato, tale limitazione può essere superata trasformando il STSP in un particolare ATSP e applicando il metodo del Branch-and-bound sopra esposto. Nel caso del STSP, però, può risultare vantaggioso applicare dei metodi diversi: anziché limitarsi all'introduzione di vincoli molto semplici (fissaggio di variabili a 0 o a 1) per eliminare

dei sotto-cicli in modo implicito senza perdere la totale unimodularità della matrice dei vincoli, si permette l'introduzione di vincoli di eliminazione esplicita dei sotto-cicli. In questo caso si perde la totale unimodularità della matrice dei vincoli, ma, invece di risolvere dei MILP ad ogni iterazione, si considerano dei metodi diversi, basati sulla soluzione del rilassamento continuo del problema con un numero limitato di vincoli di sub-tour elimination e sull'individuazione di "sotto-cicli" a partire da soluzioni del rilassamento continuo.

2.2 Separazione dei vincoli di eliminazione di sotto-ciclo e soluzione del rilassamento continuo

Si supponga di risolvere il *rilassamento continuo* di un STSP con un numero limitato di vincoli di eliminazione di sotto-cicli e sia $x_R^* \in [0,1]^{|E|}$ la soluzione ottima. In questo caso, il grafo di supporto $G_{x_R^*}$ non ci aiuta a individuare dei sotto-cicli: non basta individuare le componenti connesse su questo grafo e, di conseguenza, dei vincoli di eliminazione di sotto-ciclo violati. Il problema è quindi il seguente:

> *Data la soluzione ottima $x_R^* \in [0,1]^{|E|}$ del rilassamento continuo di un STSP con un numero limitato di vincoli di eliminazione di sotto-cicli, individuare un vincolo di eliminazione di sotto-ciclo violato da x_R^*.*

Questo problema è detto *Problema di separazione* perché ci permette di *separare* la soluzione ottima corrente x_R^* dall'insieme delle soluzioni che non violano nessun vincolo di sotto-ciclo. Per poterlo risolvere, conviene scrivere i vincoli $(STSP : 2)$ nella seguente forma equivalente:

$$\sum_{e \in \delta(S)} x_e \quad \geq \quad 2 \quad \forall \quad S \subset V : 3 \leq |S| \leq |V| - 1 \quad (STSP : 2')$$

$\delta(S)$ è l'insieme degli spigoli con un solo estremo in S e l'altro estremo in $V \setminus S$, e il vincolo impone che almeno due di questi spigoli siano presenti in una soluzione. In questo modo si impone che non ci siano sotto-insiemi isolati dal resto dei nodi e, quindi, si impone che gli spigoli selezionati ($x_e = 1$) formino un unico grafo connesso. Infatti, se esistesse un sotto-ciclo (vedi Figura 3), allora si può scegliere $\hat{S}$ come il sottoinsieme dei nodi nel sotto-ciclo e $\sum_{e \in \delta(\hat{S})} x_e = 0 < 2$.

Se, invece, non esistono sotto-cicli, allora, comunque si scelga un sottoinsieme di nodi $\hat{S}$, da questo sottoinsieme ci saranno almeno due spigoli che permettono di passare da $\hat{S}$ a $V \setminus \hat{S}$ e viceversa (vedi Figura 4).

Consideriamo adesso la soluzione frazionaria x_R^*. Per come è stata ottenuta, questa soddisfa tutti i vincoli sul grado (pari a 2) di ogni nodo, ma potrebbe non soddisfare uno dei vincoli di eliminazione di sotto-ciclo. Consideriamo l'esempio in Figura 5, dove x_R^* è pari a 1 sugli spigoli con linea continua, 1/2 sugli spigoli con linea tratteggiata e 0 sugli spigoli non rappresentati.

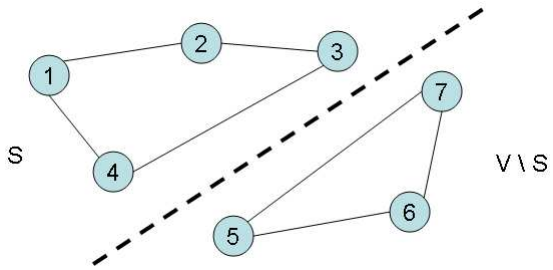


Figura 3: Presenza di sotto-cicli.

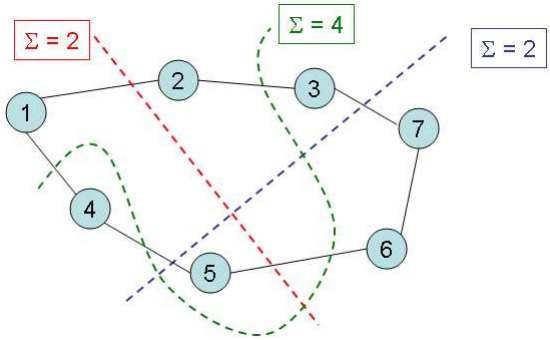


Figura 4: Esclusione di sotto-cicli. Σ indica il numero di archi che collegano $\hat{S}$ a $V \setminus \hat{S}$.

Chiaramente, la somma delle x_R^* su ogni nodo è pari a 2, ma il vincolo di eliminazione del sotto-ciclo relativo al sottoinsieme $S' = \{1, 2, 3, 4\}$ è violato. Per escludere (separare) tale soluzione basterebbe includere il vincolo:

$$\sum_{e \in \delta(S')} x_e \geq 2$$

Si può mostrare che, se costruiamo un grafo $G_{sep} = (V, E)$ in cui ad ogni spigolo $e \in E$ associamo una *capacità* $c_e = x_R^*(e)$, allora $\sum_{e \in \delta(S)} x_e$ è esattamente il valore del massimo flusso da un nodo in S ad un nodo in $V \setminus S$. Quindi:

> *Il problema di separazione per individuare un vincolo di eliminazione di sotto-ciclo violato è riconducibile ad un problema di massimo flusso.*[3]

In particolare, bisognerà scegliere un nodo s da considerare come sorgente e risolvere $|V| - 1$ problemi di massimo flusso in cui ciascuno dei rimanenti nodi è il pozzo. In corrispondenza di un massimo flusso di capacità inferiore a 2 è possibile individuare[4] un

[3] Si tratta di un problema si massimo flusso su una rete non orientata che può essere risolto, in base alle nostre conoscenze, trasformando la rete nella corrispondente orientata, attraverso la duplicazione di ogni spigolo e con due archi con la stessa capacità c_e.

[4] Omettiamo i dettagli.

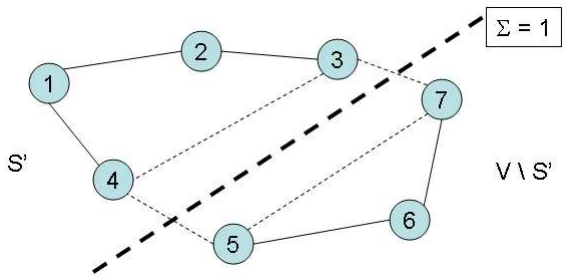


Figura 5: Un vincolo violato da una soluzione frazionaria.

sottoinsieme di nodi S che contiene s e che è connesso ai restanti nodi in $V \setminus S$ da "meno di 2" spigoli e, quindi, un vincolo di sotto-ciclo violato.[5]

La disponibilità di un metodo di separazione dei vincoli di eliminazione di sotto-ciclo a partire da una soluzione frazionaria ci permette di risolvere il rilassamento continuo del STSP, attraverso il metodo in Figura 6.

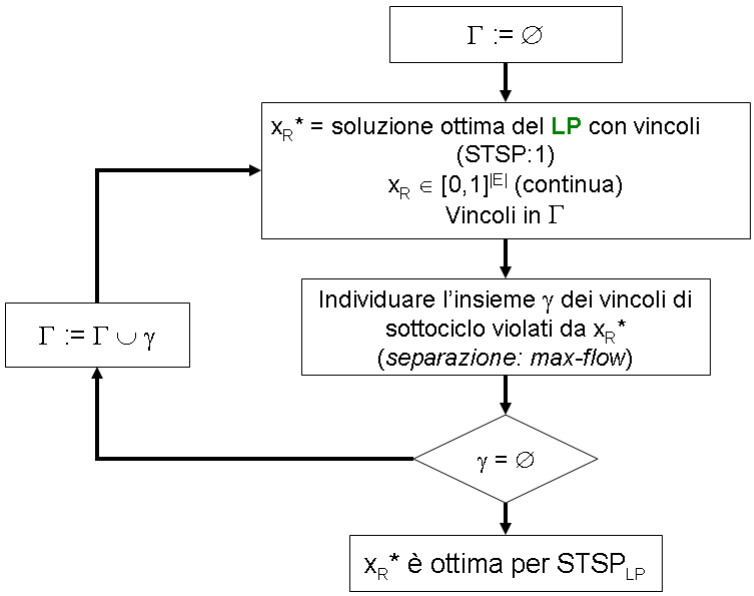


Figura 6: Soluzione del rilassamento continuo del STSP.

Si parte dal problema con variabili continue e senza vincoli di eliminazione di sotto-ciclo e, ad ogni iterazione, si risolve un problema di programmazione lineare (*Linear Program*

[5]Esistono comunque metodi più efficienti per risolvere questo particolare problema di massimo flusso, sia in base alla definizione su un grafo non orientato, sia in base al fatto che, in realtà, s non è definita e basta individuare due sottoinsiemi di nodi tali che la capacità degli archi (e quindi del flusso tra i due sottoinsiemi) sia inferiore a 2.

- LP). Si applica quindi l'algoritmo di separazione basato sul massimo flusso e, se si individuano dei vincoli di sotto-ciclo violati, si aggiungono alla formulazione e si itera la soluzione di un nuovo LP. Si noti che, ad ogni iterazione, si applicano due algoritmi *efficienti* per la soluzione di un LP (ad esempio il simplesso) e del problema di separazione. In particolare, se il problema di separazione non individua vincoli violati, vuol dire che *tutti* i vincoli di eliminazione di sotto-ciclo sono rispettati e, quindi, la soluzione corrente è la soluzione ottima del rilassamento continuo. Si noti che la convergenza ad una soluzione ottima è garantita dalla finitezza del numero di vincoli si sub-tour elimination.

2.3 Generazione di vincoli per STSP

La soluzione ottima del rilassamento continuo, anche se sono rispettati tutti i vincoli $(STSP:1)$ e $(STSP:2)$, non è, in generale, una soluzione ammissibile per il STSP, non essendo garantita l'interezza di tutte le variabili. Si consideri l'esempio in Figura 7.

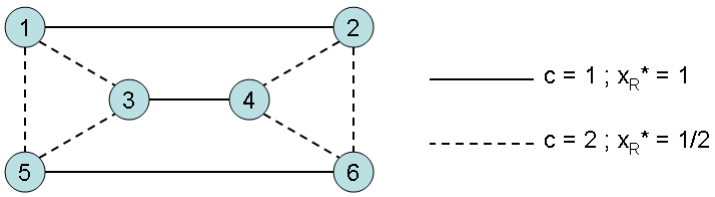


Figura 7: Soluzione frazionaria del rilassamento continuo.

La soluzione rappresentata è una soluzione ottima del rilassamento continuo che soddisfa tutti i vincoli $(STSP:1)$ e $(STSP:2)$. Il valore della soluzione è $z_R^* = 9$. Una soluzione ottima intera del problema sullo stesso grafo contiene al massimo 2 spigoli di costo 1[6] e, quindi, almeno 4 spigoli di costo 2 (ogni ciclo hamiltoniano contiene $|V| = 6$ spigoli): il costo di un ciclo hamiltoniano è almeno pari a 10. Quindi, in questo caso, non è possibile trovare soluzioni ottime frazionarie che siano anche ammissibili per il STSP.

L'esempio mette in evidenza che non è sufficiente separare i vincoli $(STSP:2)$ sul rilassamento continuo e che, dopo tale fase, è necessario procedere alla soluzione del problema che comprende i vincoli di interezza delle variabili. La nuova soluzione ottenuta imponendo l'interezza delle variabili potrebbe comunque contenere dei nuovi sotto-cicli, non compresi nei vincoli generati attraverso la separazione di soluzioni frazionarie. Pertanto, per arrivare a una soluzione del STSP si potrebbe applicare, lo schema di generazione di vincoli, a partire però dall'insieme di vincoli Γ ottenuto nella fase di soluzione del rilassamento continuo. Si ottiene lo schema di soluzione in Figura 8.

[6]Non è possibile costruire cicli hamiltoniani che contengano tutti e 3 gli spigoli di costo 1: questi sono tutti e soli gli spigoli tra i due sottoinsiemi di nodi $D = \{1, 3, 5\}$ e $P = \{2, 4, 6\}$. Supponiamo di avere un ciclo C hamiltoniano con i tre spigoli e percorriamolo, ad esempio, a partire da un nodo in D. Utilizzando uno degli spigoli di costo 1 passeremmo da D a P; con un secondo da S a D e con il terzo di nuovo a P. Non avendo altri spigoli tra D e P non potremmo tornare al nodo di partenza in D e quindi C non sarebbe un ciclo.

I vantaggi rispetto allo schema precedente risiedono nel fatto che molti vincoli di eliminazione di sotto-ciclo sono generati attraverso la soluzione *efficiente* di problemi di programmazione lineare e problemi di flusso massimo. Questo permette alla seconda fase, in cui si risolvono iterativamente dei MILP, di convergere più velocemente a soluzioni senza sotto-cicli. Complessivamente, quindi, i tempi di calcolo sono, mediamente, ridotti: si risolvono dei LP in più (prima fase), ma si considerano dei MILP in meno (seconda fase). Tuttavia, tale schema non è pienamente soddisfacente, potendoci essere molte iterazioni nella Fase MILP.

2.4 Branch-and-Bound per STSP

Il miglior metodo ad oggi disponibile per la soluzione del STSP si basa sulla seguente osservazione: avendo a disposizione un metodo *efficiente* per la soluzione del rilassamento continuo del STSP, possiamo applicare lo schema classico del Branch-and-Bound per problemi di programmazione lineare intera in cui si costruisce un albero basandosi sulle seguenti regole:

- la valutazione del *lower bound* si ottiene risolvendo il rilassamento continuo del problema rappresentato da un nodo dell'albero di Branch-and-Bound. Per risolvere il rilassamento continuo si applica lo schema di generazione di vincoli sopra esposto;
- si utilizza un *branching* dicotomico: sia x_R^* la soluzione ottima del rilassamento e $x_R^*(e)$ una variabile frazionaria in corrispondenza dello spigolo e: si generano due nodi figli aggiungendo il vincolo di branching $x_R^*(e) = 0$ o $x_R^*(e) = 1$.

Lo schema è molto efficiente in quanto i nodi figli ereditano i vincoli di eliminazione dei sotto-cicli individuati nei nodi padre e questo permette di ridurre drasticamente il numero di iterazioni (e quindi di vincoli aggiuntivi) per ottenere il lower bound nei nodi figli.

Inoltre, lo schema può essere integrato dall'aggiunta di disuguaglianze valide specifiche per il STSP: si tratta di disuguaglianze che non cambiano l'insieme delle soluzioni intere ammissibili ma che *tagliano* delle soluzioni frazionarie (in analogia alle *cover inequalities* per il problema dello zaino, che vedremo successivamente). Nella valutazione del lower bound di ogni nodo, quindi, oltre alla separazione dei vincoli di sotto-ciclo violati, si separano anche delle ulteriori disuguaglianze valide che aiutano ad ottenere delle soluzioni senza sotto-cicli e, comunque, a rendere più stringente il bound. In questo modo, si è arrivati a risolvere istanze di STSP con *milioni (!!!)* di nodi (in tempi di calcolo dell'ordine di pochi giorni).

3 Cenni al metodo del Branch-and-Cut

La tecnica di migliorare il bound attraverso l'introduzione *dinamica* di disuguaglianze valide violate (tagli o *cuts*) è generalizzabile a tutti i problemi di programmazione lineare intera (mista), a condizione di:

- disporre di una classe di disuguaglianze valide in grado di fornire una *formulazione migliore* del problema dato (al limite la formulazione ideale), che non conviene introdurre tutte fin dall'inizio, ad esempio per la loro numerosità (si pensi alle cover inequalities per il problema dello zaino);
- disporre di algoritmi di separazione efficienti per le stesse disuguaglianze.

Un caso particolare è l'introduzione delle disuguaglianze valide che sono violate al solo nodo radice, dopodiché si procede con un Branch-and-Bound classico in cui, in ogni nodo, si ereditano tali disuguaglianze e non se ne separano di ulteriori. Più in generale, lo schema è ripetibile ad ogni nodo, per migliorare il bound ottenuto dal rilassamento continuo attraverso disuguaglianze valide violate dalla soluzione di uno specifico nodo (e non dalle soluzioni dei nodi pro-genitori). Tale tecnica è stata applicata con successo a diverse classi di problemi di ottimizzazione combinatoria. La chiave del successo è lo studio e la scoperta di particolari classi di disuguaglianze valide in grado di riprodurre o almeno approssimare la formulazione ideale di un problema, insieme a delle tecniche efficienti per la loro separazione.

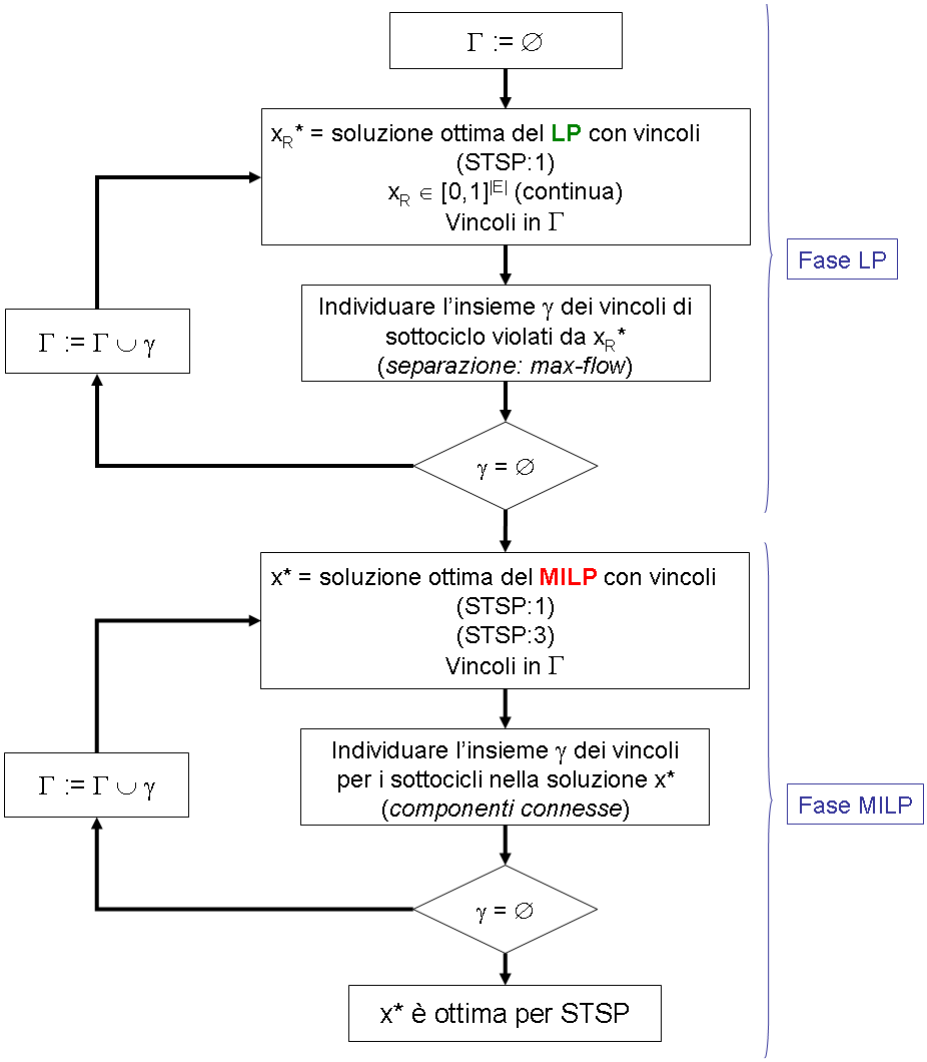


Figura 8: Generazione di vincoli su LP e su MILP per STSP