Problema del Commesso Viaggiatore

Michele Monaci

Dipartimento di Ingegneria dell'Informazione, Università di Padova
Viale Gradenigo, 6/A - 35131 - Padova
monaci@dei.unipd.it

1 Introduzione

Dato un grafo $G = (V, A)$ con $n = |V|$ vertici, pesato sugli archi, il problema del commesso viaggiatore (*Travelling Salesman Problem*, TSP) richiede di determinare un circuito nel grafo in modo che:

- ogni vertice $i \in V$ sia visitato esattamente una volta nel ciclo;
- il costo degli archi selezionati sia minimo.

Un circuito che visita ogni vertice una ed una sola volta si dice *circuito hamiltoniano*. Nel caso in cui il grafo sia orientato, si parla di problema del commesso viaggiatore asimmetrico (ATSP); nel caso particolare di grafo non orientato si parla del problema del commesso viaggiatore simmetrico (STSP).

Una interpretazione alternativa del TSP consiste nel problema di sequenziare n lavori su una unica macchina. Gli n lavori devono essere processati in un ordine qualunque e l'obiettivo è quello di completare tutti i lavori il prima possibile. Supponiamo che la macchina debba trovarsi in certe condizioni (ad esempio ad una certa temperatura, o corredata da particolari utensili, o in una certa posizione, ...) per svolgere ciascun lavoro, e che esista un tempo tecnico t_{ij} di set-up che bisogna aspettare per poter processare il lavoro j subito dopo il lavoro i. Dato che il tempo totale di processamento di tutti i lavori è costante, il problema consiste nel minimizzare la somma dei tempi di set-up tra una lavorazione e la successiva.

Nella definizione del problema non poniamo nessuna assunzione relativamente ai costi degli archi; questo significa che tutte le proprietà che deriveremo per il problema valgono anche nel caso di costi negativi. Infatti, la soluzione ottima x^* non cambia se tutti i costi vengono modificati della stessa quantità k: cambia ovviamente il valore della soluzione, di una quantità costante pari a $n \cdot k$

2 Modello Matematico ATSP

Nel seguito indicheremo con c_{ij} il costo del generico arco $(i, j) \in A$, ed immagineremo sempre di lavorare con un grafo completo, eventualmente ponendo $c_{ij} = +\infty \; \forall (i, j) \notin A$.

Introducendo le seguenti variabili binarie

$$x_{ij} = \begin{cases} 1 & \text{se l'arco } (i,j) \text{ viene selezionato} \\ 0 & \text{altrimenti} \end{cases} \qquad (i,j \in V)$$

un possibile modello di PLI potrebbe essere il seguente:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij}\ x_{ij} \tag{1}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \tag{2}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad i \in V \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad i,j \in V \tag{4}$$

La funzione obiettivo (1) richiede di miminizzare la somma dei costi degli archi scelti nel circuito, mentre i vincoli (2) e (3) impongono che ciascun vertice abbia esattamente un arco entrante ed un arco uscente, rispettivamente; infine, i vincoli (4) impongono che le variabili siano binarie.

I vincoli del modello (1)–(4) impongono che qualunque soluzione contenga almeno un circuito, ma non necessariamente uno solo; infatti sono ammissibili soluzioni composte da più di un circuito, come quella in figura 1.

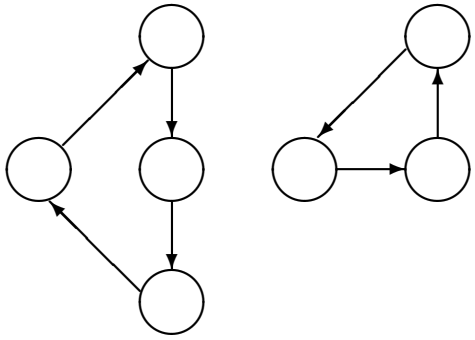


Figura 1: Significato dei Subtour Elimination Constraints.

Per ottenere un modello valido per l'ATSP occorre aggiungere dei vincoli che impongano che la soluzione sia composta da un solo circuito; questi vincoli, noti come *Subtour Elimination Constraints* (SECs), hanno la seguente struttura:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad S \subset V,\ |S| \geq 1$$

Dato un qualunque insieme (non nullo) $S \subset V$ di vertici, i SECs impongono che il numero di archi scelti con entrambi gli estremi in S sia al più pari a $|S| - 1$, ossia che non ci siano dei cicli in S; ovviamente S deve essere un sottoinsieme non vuoto di vertici e non deve coincidere con V (altrimenti sarebbero proibiti anche i circuiti hamiltoniani).
Si noti che i SECs sono simmetrici, nel senso che, dato un insieme S^*, il vincolo scritto per S^* e quello per $V \setminus S^*$ coincidono; pertanto possiamo sempre ipotizzare che un vertice a

scelta (ad esempio il vertice 1) appartenga all'insieme S.
Infine, si noti che non è necessario imporre i SECs quando $|S| = 1$, se si evita di introdurre le variabili del tipo x_{ii} (inutili per il TSP); questo significa che anche i vincoli con $|S| = |V|-1$ sono ridondanti.
Pertanto è possibile scrivere i SECs nel seguente modo:

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1 \qquad S \subset V, 1 \in S, 2 \leq |S| \leq |V| - 2 \tag{5}$$

Un modo alternativo di proibire i sottocicli è quello di rimpiazzare i (5) con i seguenti vincoli

$$\sum_{i\in S}\sum_{j\notin S} x_{ij} \geq 1 \qquad S \subset V, 1 \in S, 2 \leq |S| \leq |V| - 2 \tag{6}$$

che impongono la connessione tra l'insieme dei vertici S (con $1 \in S$) e l'insieme $V \setminus S$. Si può dimostrare che, se i vincoli (2)-(4) sono imposti, scrivere i (5) oppure scrivere i (6) è assolutamente indifferente (anche a livello di rilassamento continuo).
Infatti sia $S \subset V$ e supponiamo che i vincoli (2)-(4) siano imposti. Allora si ha che

$$\sum_{i\in S}\sum_{j\in S} x_{ij} = \sum_{i\in S}\sum_{j\in S} x_{ij} + \sum_{i\in S}\sum_{j\notin S} x_{ij} - \sum_{i\in S}\sum_{j\notin S} x_{ij} = \sum_{i\in S}(\sum_{j\in V} x_{ij}) - \sum_{i\in S}\sum_{j\notin S} x_{ij} = |S| - \sum_{i\in S}\sum_{j\notin S} x_{ij}.$$

Se vale il vincolo (5) associato all'insieme S, si ha che $\sum_{i\in S}\sum_{j\in S} x_{ij} = |S| - \sum_{i\in S}\sum_{j\notin S} x_{ij} \leq |S| - 1$,
ossia $\sum_{i\in S}\sum_{j\notin S} x_{ij} \geq 1$; quindi il vincolo (6) corrispondente è soddisfatto.
Viceversa, se vale il vincolo (6) associato all'insieme S si ha che $\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1$, cioè il corrispondente vincolo (5) è soddisfatto.

Il numero di SECs, sia nella forma (5) che nella (6), è esponenziale. Pertanto, non solo il problema del commesso viaggiatore in quanto tale è NP-hard (in quanto vi sono i vincoli di interezza delle variabili), ma la dimensione della formulazione (intesa come numero di variabili e vincoli) rende impraticabile anche la soluzione del rilassamento continuo (almeno per istanze di dimensioni interessanti).

2.1 Rilassamento per eliminazione dei SECs

Eliminando i SECs si ottiene il modello (1)–(4) che corrisponde al problema dell' assegnamento, risolubile in tempo polinomiale con l'algoritmo ungherese. Dato che il numero di possibili assegnamenti distinti è $n!$ e che di questi $(n-1)!$ sono dei tour, ciò significa che mediamente un assegnamento su n è un tour. Si noti poi che tra gli assegnamenti ammissibili ci sono anche quelli che utilizzano gli elementi della diagonale principale, corrispondenti ad archi senza alcun interesse per il problema del commesso viaggiatore. Modificando gli elementi della matrice dei costi in modo da eliminare questi elementi diagonali, il numero di possibili assegnamenti distinti si riduce a $\lfloor n!/e + 1/2 \rfloor$, per cui aumenta la probabilità che l'assegnamento trovato sia anche un tour. Questo suggerisce il fatto che il rilassamento per eliminazione dei SECs è molto vicino al valore della soluzione ottima (su alcune classi di istanze il lower bound fornito è circa pari al 99% del valore della soluzione ottima).

2.1.1 Possibile schema di branching

La soluzione del rilassamento per eliminazione dei vincoli di subtour porta ad un problema di assegnamento, la cui soluzione ottima contiene, in generale, dei sottocicli: un algoritmo branch-and-bound basato sul rilassamento per eliminazione dei SECs deve prevedere uno schema di branching che consenta di "rompere" i sottocicli.

Per eliminare una eventuale soluzione rilassata non ammissibile si può considerare il seguente schema di branching: si prende il ciclo più corto (di lunghezza k) e si generano k figli vietando, a turno, uno degli archi del ciclo. Nell'esempio di figura 2 il ciclo più corto coinvolge 3 archi, per cui dal nodo corrente vengono generati 3 figli nei quali, a turno, ciascun arco è imposto essere non in soluzione. Si può inoltre rafforzare il branching imponendo che, nel generico nodo figlio j ($j = 1, \ldots, k$), gli archi di posizione $1, \ldots, j-1$ nel ciclo siano in soluzione. Ovviamente si evita di generare i nodi corrispondenti a condizioni che vietano archi fissati in soluzione ai livelli precedenti.

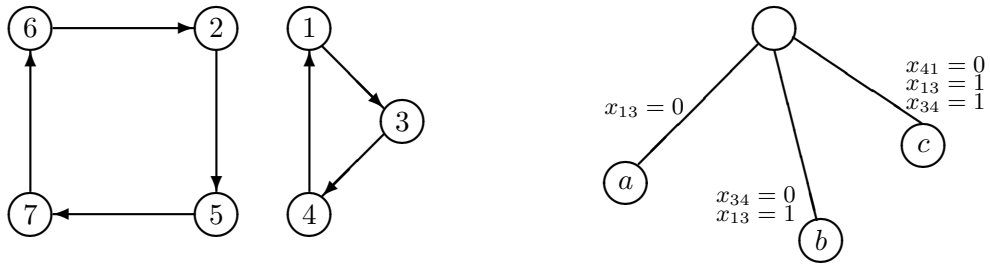


Figura 2: Schema di branching per eliminazione di subtour.

Il sottoproblema da risolvere ad ogni nodo è sempre un problema di assegnamento con alcuni archi fissati ed altri archi proibiti. Per poter tenere conto di questo fatto basta modificare la matrice dei costi del problema di assegnamento nel seguente modo: se $x_{ij} = 0 \rightarrow c_{ij} = \infty$ (in modo da penalizzare la scelta di un arco proibito), se $x_{ij} = 1 \rightarrow c_{ik} = \infty \; \forall k \neq j$ (in modo da penalizzare la mancata scelta di un arco fissato).

La soluzione del rilassamento del nodo corrente non è ammissibile per il nodo figlio in quanto un arco (presente nella soluzione rilassata del nodo corrente) è stato proibito dal branching. Il nodo figlio quindi eredita dal padre un assegnamento ammissibile parziale di dimensione $n - 1$, cui deve aggiungere un solo cammino aumentante; pertanto, ad ogni nodo dell'albero decisionale (a parte il nodo radice), il rilassamento può essere calcolato in maniera incrementale in tempo $O(n^2)$.

2.2 Rilassamento continuo del modello

Considerando un modello per l'ATSP, il rilassamento continuo consiste nel rimpiazzare i vincoli (4) con i vincoli

$$x_{ij} \geq 0 \quad i, j \in V$$

Si noti che la condizione $x_{ij} \leq 1$ è implicata dai vincoli (2) e (3), che impongono che la somma di un sottoinsieme di variabili sia 1.

Come già detto, il numero dei SECs è esponenziale in n, per cui è impossibile pensare di utilizzare un LP solver per risolvere direttamente il rilassamento continuo del modello. Per poter sperare di risolvere il rilassamento continuo è opportuno definire un algoritmo di tipo *cutting plane* che aggiunga i vincoli all'LP solo quando necessario.
Quindi l'idea di base è quella di risolvere un ulteriore rilassamento del rilassamento continuo, ottenuto eliminando tutti i SECs, e di *separare* i SECs violati dalla soluzione ottima $\tilde{x}$ del rilassamento corrente. Aggiungendo questi vincoli al modello ed iterando il procedimento si giunge ad una situazione nella quale tutti i SECs, seppur non presenti esplicitamente nel modello, sono rispettati; in tal caso la soluzione $\tilde{x}$ è la soluzione ottima del rilassamento continuo del modello comprendente tutti i vincoli. È da notare comunque che, ad ogni iterazione, la soluzione ottima del rilassamento corrente fornisce un lower bound sul valore della soluzione ottima.

2.2.1 Separazione dei SECs

Il problema di separazione dei SECs può essere formulato nel seguente modo: data una soluzione $\tilde{x}$, determinare, se esiste, un sottoinsieme $S^* \subseteq V$ che viola il vincolo di subtour associato. Immaginando di lavorare sulla formulazione nella quale i SECs sono imposti nella forma (6), il problema di separazione consiste nel trovare un insieme S^* tale che

(a) $2 \leq |S^*| \leq |V| - 2$

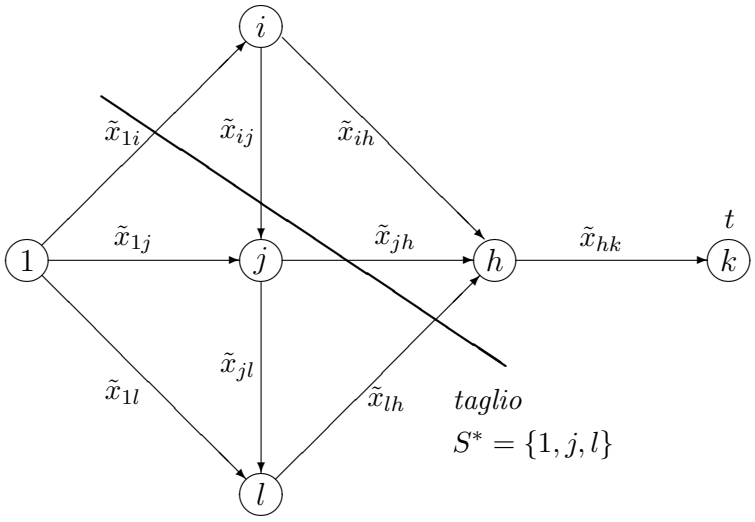
(b) $1 \in S^*$

(c) $\sum_{i \in S^*} \sum_{j \notin S^*} x_{ij} < 1$

Ovviamente si potrebbe pensare di enumerare tutti i sottoinsiemi S^* che soddisfano le condizioni (a) e (b), selezionando tutti quelli che soddisfano anche la (c). Però il numero esponenziale di sottoinsiemi da controllare ad ogni iterazione renderebbe il problema di separazione (e l'intero procedimento di soluzione dell'LP) estremamente lento. L'alternativa consiste nel risolvere il problema di separazione definendo (e risolvendo) un opportuno problema (spesso di PLI). Il fatto di dovere risolvere un PLI all'interno di una procedura per la soluzione di un LP (che a sua volta risulta spesso essere il rilassamento continuo di un PLI) può sembrare estremamente inefficiente, ma la realtà è che anche risolvendo un PLI per la separazione il procedimento funziona bene in pratica.

Alla prima iterazione, il problema rilassato è un problema di assegnamento, per cui la soluzione ottima $\tilde{x}$ è intera. In tal caso è facile verificare se esistono dei SECs violati: il problema consiste nel verificare se la soluzione contiene dei sottocicli. Questo problema può essere formulato come problema di raggiungibilità: partendo da un vertice a piacere (ad esempio il vertice 1), è possibile visitare tutti gli altri vertici utilizzando la soluzione $\tilde{x}$ prima di ritornare al vertice di partenza? Se la risposta è affermativa, allora vuol dire che la soluzione trovata è effettivamente un cammino hamiltoniano e nessun vincolo è violato. Se la risposta è no possiamo considerare come insieme S^* l'insieme di vertici raggiungibili dal vertice di partenza ed aggiungere alla formulazione il vincolo corrispondente a tale insieme. In questo caso quindi il problema di separazione è risolubile in tempo $O(n)$.

Il problema di separazione è più complesso alle iterazioni successive, quando la soluzione $\tilde{x}$ non è intera. I vincoli (6) impongono che la soluzione $\tilde{x}$ induca un sottografo connesso, ossia tale da contenere un cammino dal vertice 1 ad ogni vertice $k \in V \setminus \{1\}$. Immaginiamo di aver fissato il vertice $k \in V \setminus \{1\}$; la condizione di connessione equivale a richiedere che il valore di qualunque taglio della forma $(S_k, V \setminus S_k)$ con $1 \in S_k$ e $k \notin S_k$ sia almeno 1. Quindi, se il vertice 1 non è connesso al vertice k (cioè se esiste un vincolo (6) violato) esiste un insieme S_k tale che $1 \in S_k$, $k \notin S_k$ ed il valore del taglio $(S_k, V \setminus S_k)$ è minore di 1. Per trovare un taglio con queste caratteristiche cerchiamo il taglio di valore minimo dal vertice 1 al vertice k e verifichiamo se questo valore è maggiore o minore di 1. Il problema del taglio minimo corrisponde al problema del flusso massimo che è possibile inviare dal nodo 1 al nodo k, utilizzando come capacità di ciascun arco (i, j) il valore della corrispondente variabile $\tilde{x}_{ij}$ nella soluzione corrente. Se il valore del flusso massimo è minore di 1, allora abbiamo identificato un taglio di valore minore di 1: l'insieme dei vertici cui corrisponde il taglio è l'insieme S^* che soddisfa le condizioni (a), (b) e (c), per il quale esiste un vincolo violato. Viceversa, se il valore del taglio è maggiore o uguale a 1 $\forall k \in V \setminus \{1\}$, allora tutti i SECs sono rispettati.



Il problema di separazione consiste quindi nel risolvere $n-1$ problemi di flusso (uno per ogni possibile nodo k di destinazione); ciascun sottoproblema è risolubile in tempo $O(n^3)$ attraverso una versione modificata dell'algoritmo di Ford-Fulkerson. Per poter risolvere il problema di separazione con un algoritmo standard di flusso massimo, bisogna eliminare il flusso entrante nel vertice 1, ossia eliminare i corrispondenti archi nella rete, oppure togliere il flusso entrante in 1 dal valore del flusso uscente.

Ad ogni iterazione almeno un vincolo violato (se esiste) viene aggiunto alla formulazione corrente. Il numero di iterazioni necessarie per risolvere esattamente il rilassamento continuo del modello potrebbe, in linea di principio, essere esponenziale in n. È stato però dimostrato che per descrivere completamente il politopo del rilassamento continuo dell'AT-SP bastano un numero polinomiale di SECs. In particolare, risolvendo ad ogni iterazione il problema rilassato con l'algoritmo dell'ellissoide, si riesce a dimostrare che il rilassamento continuo del TSP può essere risolto in tempo polinomiale (Grötschel, Lovász & Schrijver). Quindi la soluzione del rilassamento continuo del modello richiede un numero polinomiale di iterazioni di separazione, ciascuna delle quali richiede un tempo polinomiale; quindi l'intero procedimento di soluzione del rilassamento continuo richiede un tempo polinomiale.

3 Formulazioni compatte

Il risultato della sezione precedente è estremamente importante dal punto di vista teorico, ma ha anche un riscontro pratico (in quanto il metodo funziona effettivamente in modo efficiente), e conferma i seguenti risultati di validità più generale:

Teorema: *Se il problema di separazione (o di generazione di colonne) è risolubile in tempo polinomiale (nella taglia del problema), allora anche il numero di iterazioni è polinomiale (nella taglia del problema).*

Proposizione: *Se il problema di separazione (o di generazione di colonne) è formulabile come problema di programmazione di dimensione polinomiale (nella taglia del problema) in cui $\tilde{x}$ compare solo nel termine noto o solo nei costi, allora anche il problema originale è formulabile come LP di dimensione polinomiale (nella taglia del problema).*

Infatti, si consideri il problema (P)

$$
\begin{aligned}
(\mathtt{P}) \qquad & \min c^T\, x && (7)\\
& A\, x \le b && (8)\\
& D\, x \le e && (9)\\
& x \ge 0 && (10)
\end{aligned}
$$

nel quale immaginiamo che i vincoli (8) siano "pochi" mentre i vincoli (9) siano "molti". Immaginiamo di voler risolvere il problema con un algoritmo di tipo cutting-plane, risolvendo ad ogni iterazione il problema di separazione dei vincoli (9), e che questo problema abbia la seguente forma.
Problema di separazione: dato $\widetilde{x}$, verificare se $v \ge 1$ dove

$$
\begin{aligned}
(\mathtt{S}) \qquad & v = \max f^T\, y\\
& G\, y \le H\\
& L\, y \le M(\widetilde{x})\\
& y \ge 0
\end{aligned}
$$

dove $M(x)$ è lineare in x ed il problema di separazione (S) ha taglia polinomiale in n. Questo corrisponde a dire che

$$D\, \widetilde{x} \le e \Longleftrightarrow v \ge 1.$$

Allora il problema (P) può essere riformulato nel seguente modo:

$$
\begin{aligned}
(\overline{\mathtt{P}}) \qquad & \min c^T\, x\\
& A\, x \le b\\
& x \ge 0\\
& f^T\, y \ge 1\\
& G\, y \le H\\
& L\, y \le M(x)\\
& y \ge 0
\end{aligned}
$$

cioè esiste una formulazione di dimensione polinomiale per (P).

Se invece il problema di separazione è definito nel seguente modo:

$$(\mathtt{S}') \qquad \begin{aligned} v' = \max \widetilde{x}^T \; y \\ G \; y \leq H \\ y \geq 0 \end{aligned}$$

allora si ha anche che

$$(\mathtt{S}'') \qquad \begin{aligned} v' = \min H^T \; u \\ \widetilde{x}^T \geq u^T \; G \\ u \geq 0 \end{aligned}$$

e quindi $(\overline{\mathtt{P}}')$ è formulabile come:

$$(\overline{\mathtt{P}}') \qquad \begin{aligned} \min c^T \; x \\ A \; x \leq b \\ x \geq 0 \\ H^T \; u \geq 1 \\ u^T \; G \leq x^T \\ u \geq 0 \end{aligned}$$

In maniera speculare, vale il seguente risultato "negativo".

Teorema: *Se il problema di separazione (o di generazione di colonne) è NP-completo, allora anche risolvere il rilassamento continuo è un problema NP-completo.*

3.1 Modello Alternativo ATSP

Il fatto che il rilassamento continuo del modello (1)–(5) possa essere risolto in tempo polinomiale anche se il numero di vincoli è esponenziale, suggerisce che esista una formulazione alternativa di PLI nella quale il numero di variabili e vincoli sia polinomiale. L'idea di base per ottenere questa formulazione è quella di esprimere i vincoli di subtour tramite un insieme di vincoli lineari, basandosi sulla formulazione del problema di separazione come problema di flusso.

Oltre alle variabili x_{ij}, occorre introdurre le seguenti variabili continue

$$f_{ij}^k = \text{flusso lungo l'arco } (i,j) \text{ utilizzando il nodo } k \text{ come terminale} \quad (i,j \in V,\ k \in V')$$

dove $V' = V \setminus \{1\}$.

Il corrispondente modello matematico è:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} \; x_{ij} \tag{11}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \tag{12}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad i \in V \tag{13}$$

$$x_{ij} \in \{0,1\} \qquad i,j \in V \tag{14}$$

$$\sum_{j \in V} f^k_{1j} - \sum_{i \in V} f^k_{i1} \geq 1 \qquad k \in V' \tag{15}$$

$$\sum_{j \in V} f^k_{hj} - \sum_{i \in V} f^k_{ih} = 0 \qquad h,k \in V',\ h \neq k \tag{16}$$

$$f^k_{ij} \leq x_{ij} \qquad i,j \in V,\ k \in V' \tag{17}$$

$$f^k_{ij} \geq 0 \qquad i,j \in V,\ k \in V' \tag{18}$$

Il modello equivale al modello precedente dove i SECs sono rimpiazzati dai vincoli (15)–(18) che impongono che, per qualunque vertice terminale $k \in V'$, il flusso che può essere inviato da 1 a k sia almeno pari a 1. In particolare, per ogni vertice terminale k, i vincoli (15) impongono che il flusso netto uscente dal vertice 1 sia almeno 1, i vincoli (16) impongono il bilanciamento del flusso per tutti i nodi intermedi, ed i vincoli (17) e (18) corrispondono ai classici vincoli di capacità e di non negatività delle variabili, rispettivamente.

Questa formulazione, coinvolge un numero polinomiale di variabili e vincoli: questa è una dimostrazione alternativa del fatto che il rilassamento continuo di ATSP può essere risolto in tempo polinomiale.

3.2 Formulazione di Muller, Tucker e Zemlin

Un ulteriore modello per l'ATSP con un numero polinomiale di variabili e vincoli è quello proposto da Muller, Tucker e Zemlin. L'idea di base è quella di esprimere i SECs tramite un insieme di variabili che indichino, per ciascun vertice, l'ordine di visita nel ciclo.

Il modello prevede che, oltre alle variabili x_{ij}, siano definite le seguenti variabili intere

$$u_i = \text{ordine di visita del vertice } i \text{ in soluzione} \quad (i \in V)$$

Il corrispondente modello di PLI è

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij}\ x_{ij} \tag{19}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \tag{20}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad i \in V \tag{21}$$

$$x_{ij} \in \{0,1\} \qquad i,j \in V \tag{22}$$

$$u_i - u_j + n\ x_{ij} \leq n-1 \qquad i \in V,\ j \in V',\ i \neq j \tag{23}$$

$$u_1 = 0 \tag{24}$$

$$u_i \geq 0 \ \text{ (intere)} \qquad i \in V' \tag{25}$$

dove al solito $V' = V \setminus \{1\}$.

I vincoli (23) sono espressi per ogni coppia di nodi i, j distinti con $i, j \neq 1$. Ciascun vincolo è disattivato se $x_{ij} = 0$; in tal caso infatti si impone solo che la differenza tra u_i e u_j non superi $n - 1$ (imponendo che la soluzione abbia un ciclo). Viceversa, se $x_{ij} = 1$, allora il nodo i precede il nodo j ed il vincolo $u_i - u_j \leq n - 1 - n = -1$ impone che la variabile u_j sia (almeno) pari a $u_i + 1$. È da notare che le variabili u_i potrebbero anche essere definite come variabili continue (non vincolate a valori interi) e che bisogna fissare arbitrariamente il valore di una di queste variabili (ad esempio $u_1 = 0$).

4 Modello Matematico STSP

Nel caso particolare di grafo $G = (V, E)$ non orientato, indicheremo con c_e il costo del generico arco $e = (i, j) \in E$. Inoltre, dato un vertice $i \in V$, indicheremo con $\delta(i)$ l'insieme dei lati con un estremo in i. Più in generale, dato un sottoinsieme di vertici $S \subseteq V$, indicheremo con $\delta(S)$ l'insieme dei lati con esattamente un estremo in S e con $E(S)$ l'insieme dei lati con entrambi gli estremi in S.

Associando a ciascun arco $e \in E$ una variabile binaria

$$x_e = \begin{cases} 1 & \text{se l'arco } e \text{ viene selezionato} \\ 0 & \text{altrimenti} \end{cases} \qquad (e \in E)$$

si ottiene un possibile modello di PLI:

$$\min \sum_{e \in E} c_e \, x_e \tag{26}$$

$$\sum_{e \in \delta(i)} x_e = 2 \qquad i \in V \tag{27}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \qquad S \subset V, 1 \in S, 2 \leq |S| \leq |V| - 2 \tag{28}$$

$$x_e \in \{0, 1\} \qquad e \in E \tag{29}$$

Analogamente a quanto accade per l'ATSP, i vincoli (28) impediscono la presenza di sottocicli in soluzione e possono essere rimpiazzati dai seguenti vincoli di connessione

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad S \subset V, 1 \in S, 2 \leq |S| \leq |V| - 2. \tag{30}$$

L'unica differenza rispetto al caso dell'ATSP è data dal fatto che la non orientazione degli archi permette di mettere a 2 il termine noto dei vincoli (30).

4.1 Rilassamento 1-albero

Consideriamo una qualunque soluzione ammissibile per il TSP simmetrico: il numero di lati incidenti su ciascun vertice è pari a 2 e, immaginando di rimuovere i due lati incidenti su un vertice a piacere (ad esempio il vertice 1), quel che rimane è un albero ricoprente per il sottografo indotto da $V \setminus \{1\}$. Pertanto, qualunque soluzione ha la seguente struttura:

(a) l'insieme dei lati individuato forma un ciclo hamiltoniano;

(b) ci sono due lati incidenti nel vertice 1;

(c) togliendo il vertice 1 rimane un albero.

Il vincolo (a) è difficile da imporre, per cui lo si può rimuovere ed ottenere una soluzione rilassata nel seguente modo:

- calcolando lo SST sul sottografo ottenuto eliminando il vertice 1;
- aggiungendo i due lati di costo minimo incidenti in 1.

La soluzione così ottenuta si chiama *1-albero*.

Avendo rimosso il vincolo (a) abbiamo ottenuto la possibilità di ottimizzare sulla famiglia degli 1-alberi anzichè sulla famiglia dei cicli hamiltoniani; è evidente che la prima contiene la seconda come sottoinsieme stretto, per cui il valore della soluzione trovata sarà sempre minore o uguale al valore della soluzione ottima del problema di partenza.

Matematicamente questo rilassamento può essere ottenuto a partire dal modello matematico (26)–(27), (29)–(30), aggiungendo il vincolo (33) (che sarebbe ridondante), che impone di selezionare esattamente n lati in soluzione, ed eliminando i vincoli di grado (27) tranne che per il vertice 1, ottenendo il seguente problema rilassato:

$$\min \sum_{e \in E} c_e \, x_e \tag{31}$$

$$\sum_{e \in \delta(1)} x_e = 2 \tag{32}$$

$$\sum_{e \in E} x_e = n \tag{33}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad S \subset V, 1 \in S, 2 \leq |S| \leq |V| - 2 \tag{34}$$

$$x_e \in \{0, 1\} \qquad e \in E \tag{35}$$

Il problema (31)–(35) ha la seguente interpretazione: selezionare n lati di costo complessivo minimo, di cui 2 incidenti sul vertice 1, che garantiscano la connessione. Il problema risultante è quindi l'1-albero.

La qualità (in termini di valore del bound) fornito da questo rilassamento è generalmente abbastanza scarsa per le istanze della letteratura. È possibile migliorare la qualità dei bound ottenuti tenendo conto in modo lagrangiano dei vincoli rilassati. Dato un vettore $(\lambda_1, \ldots, \lambda_n)$ di moltiplicatori (non vincolati in segno) il problema lagrangiano associato è

$$\mathcal{L}_\lambda(TSP) = \min \left[\sum_{e \in E} c_e x_e + \sum_{i \in V} \lambda_i \, (2 - \sum_{e \in \, \delta(i)} x_e) \right]$$

con i vincoli (32)–(35).

La funzione obiettivo del problema lagrangiano può essere riscritta nel seguente modo

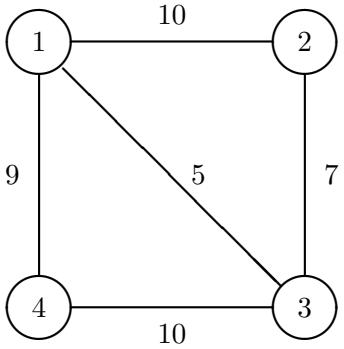
$$2 \sum_{i \in V} \lambda_i + \sum_{e \in E} \bar{c_e} \; x_e$$

dove il *costo lagrangiano* di ciascun lato $e = (i, j) \in E$ è definito come $\bar{c_e} := c_e - \lambda_i - \lambda_j$.

Le stesse considerazioni possono essere fatte nel caso del problema del commesso viaggiatore asimmetrico: il corrispondente problema rilassato sarà quello di determinare una *1-arborescenza*.

• **Esempio**

Consideriamo il seguente grafo



La soluzione ottima del rilassamento 1-albero è composta dall'albero $(2, 3), (3, 4)$ di costo 7+10=17 e dai lati $(1, 3)$ e $(1, 4)$, di costo pari a 5 e 9, rispettivamente. Il corrispondente lower bound è quindi pari a 31. La soluzione rilassata non è però ammissibile per il problema originale, in quanto non è soddisfatto il vincolo di grado per i vertici 2 e 3.

L'idea del rilassamento lagrangiano è di "pesare" l'importanza dei vari vertici nella soluzione; in particolare, si dovrà cercare di aumentare il moltiplicatore del vertice 2 (in modo da favorire i lati incidenti su questo nodo) e di diminuire il moltiplicatore associato al vertice 3 (in modo da penalizzare l'utilizzo di lati incidenti sul nodo).

Considerando il seguente vettore di moltiplicatori lagrangiani $\lambda = (0, 1, -1, 0)$, il costo degli archi viene modificato nel seguente modo:

i	j	c_e	λ_i	λ_j	$\bar{c_e}$
1	2	10	0	1	9
1	3	5	0	-1	6
1	4	9	0	0	9
2	3	7	1	-1	7
3	4	10	-1	0	11

Una soluzione ottima del rilassamento è identica a quella ottenuta dal rilassamento per eliminazione, in quanto composta dall'albero $(2, 3), (3, 4)$ e dai lati $(1, 3)$ e $(1, 4)$ (si noti che la soluzione ottenuta prendendo il lato $(1, 2)$ anzichè il lato $(1, 4)$ era altrettanto ottima).

Essendo però cambiati i costi dei lati, il bound fornito dal rilassamento è pari a $2\ (0+1-1+0)+7+11+6+9=33$.

Prendendo infine il seguente vettore di moltiplicatori $\lambda=(0,2,-4,1)$ i costi lagrangiani dei lati sono i seguenti:

i	j	c_e	λ_i	λ_j	$\bar{c_e}$
1	2	10	0	2	8
1	3	5	0	-4	9
1	4	9	0	1	8
2	3	7	2	-4	9
3	4	10	-4	1	13

L'albero di costo minimo è composto dai lati $(2,3),(3,4)$, di costo pari a 9 e 13, rispettivamente. I due lati di costo minimo incidenti nel vertice 1 sono i lati $(1,2)$ e $(1,4)$, entrambi di costo pari a 8, per cui il corrispondente bound è pari a $2\ (0+2-4+1)+13+9+8+8=36$. Questa soluzione è ammissibile per il problema di partenza ed è quindi la soluzione ottima (si ricordi che sono stati rilassati solo dei vincoli di uguaglianza).

4.1.1 Possibile schema di branching

Una soluzione ottima del rilassamento 1-albero non soddisfa, in generale, i vincoli di grado per qualche nodo, per cui un algoritmo branch-and-bound basato sul rilassamento 1-albero deve prevedere uno schema di branching che consenta di ripristinare tali vincoli.

Generalmente un algoritmo branch-and-bound basato su questo rilassamento richiede di calcolare, al nodo radice, un "buon" lower bound tramite rilassamento lagrangiano 1-albero utilizzando diversi vettori di moltiplicatori e di memorizzare il vettore di moltiplicatori cui corrisponde il miglior lower bound e la corrispondente soluzione rilassata. Se la soluzione rilassata non è ammissibile, si scelgono un vertice v con grado ≥ 3 e due lati a e b incidenti su v; il problema è suddiviso in tre sottoproblemi ottenuti vietando, a turno, il lato a, il lato b, oppure tutti gli altri lati incidenti su v (vedi figura 3)

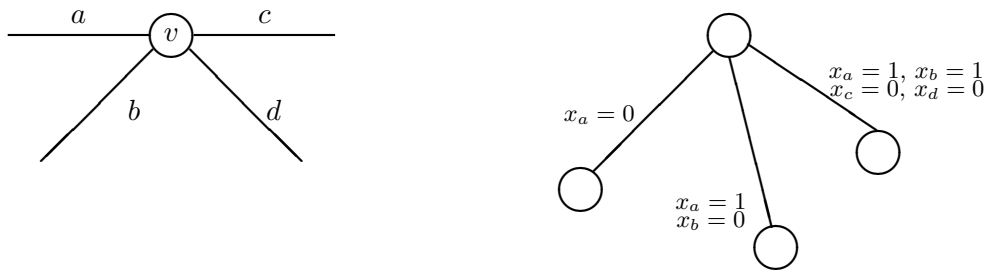


Figura 3: Schema di branching per rispettare i vincoli di grado.

Ai nodi successivi dell'albero decisionale il calcolo del lower bound deve essere effettuato in modo molto veloce, per cui il numero di iterazioni di subgradiente è generalmente molto ridotto; al limite si possono utilizzare ad ogni nodo i moltiplicatori ottimi trovati al nodo radice, in quanto i moltiplicatori ottimi del generico nodo non saranno troppo diversi da quelli trovati al nodo radice.

5 Algoritmi euristici per il TSP

Gli algoritmi euristici per il TSP si dividono essenzialmente in due categorie:

1. *Tour construction procedures*
   Sono algoritmi di costruzione di una soluzione che definiscono una soluzione approssimata ex-novo.
2. *Tour improvement procedures*
   Sono procedure di miglioramento della soluzione che, partendo da una soluzione ammissibile, cercano di migliorarla attraverso una sequenza di scambi.

Spesso gli algoritmi sono definiti per TSP simmetrico: è ovviamente possibile definire estensioni immediate di tali algoritmi al caso asimmetrico.

5.1 Tour Construction Procedures

Definiscono iterativamente la soluzione espandendo un ciclo (o un cammino) fino ad ottenere una soluzione ammissibile. Alla generica iterazione la soluzione corrente (parziale) è un cammino (ciclo) che include solo un sottoinsieme di vertici $S \subset V$; l'algoritmo termina quando $S = V$.

Gli ingredienti fondamentali nella costruzione di una soluzione sono:

1. La scelta del subtour iniziale (o del vertice di partenza).
   - scegliere un vertice h a caso, formando come subtour l'autoanello h, h, oppure il subtour h, k con il vertice k tale che c_{hk} è minima (o massima);
   - scegliere i vertici h, k cui corrisponde l'arco di costo minimo;
   - per problemi euclidei: il subtour iniziale è dato dalla convex hull dell'insieme V dei vertici (più piccolo insieme convesso che contiene tutti i vertici).
2. Il criterio di scelta del successivo vertice da inserire nel subtour.
   - scegliere un vertice k a caso tra quelli non ancora presenti nel subtour;
   - selezionare il vertice k non presente nel subtour che minimizza la distanza rispetto al vertice più vicino nel subtour
3. Il criterio di inserzione, per stabilire tra quali vertici i e j del subtour inserire il vertice scelto. Questo è il passo critico dell'algoritmo, anche se spesso la scelta su dove inserire il vertice viene fatta assieme alla scelta del prossimo vertice da inserire.
   - Trova l'arco (i, j) del subtour che minimizza la quantità $c_{ik} + c_{kj} - c_{ij}$ (che indica di quanto cresce il costo della soluzione)
   - Trova l'arco (i, j) del subtour che massimizza l'angolo formato dagli archi (i, k) e (k, j)

Gli algoritmi euristici proposti in letteratura che (mediamente) sembrano funzionare meglio sono:

- *Nearest Neighbour Algorithm*
  Il vertice iniziale è scelto a caso; poi si sceglie sempre il vertice non raggiunto più vicino all'ultimo vertice inserito e lo si inserisce alla fine del cammino. Il tempo di calcolo dell'algoritmo è $O(n^2)$.
  La figura 4 mostra la soluzione prodotta dall'algoritmo partendo dal vertice 1; la sequenza di visita dei vertici è $1, 4, 5, 3, 2, 6, 1$ ed il valore della soluzione è 35.

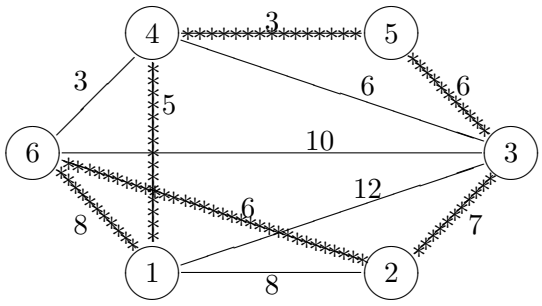


Figura 4: Algoritmo Nearest Neighbour partendo dal vertice 1.

- *Cheapest Insertion Algorithm*
  Il vertice iniziale h è scelto a caso, ed il subtour iniziale è determinato dall'arco uscente da h di costo massimo; poi si sceglie sempre il vertice che minimizza il costo di inserzione tra qualunque coppia di vertici del subtour corrente. Il tempo di calcolo dell'algoritmo è $O(n^2 \log n)$.
  Questo non è tecnicamente un algoritmo greedy, in quanto alcune delle scelte compiute (relativamente ai lati selezionati) possono essere messe in discussione nelle iterazioni successive.

It.	lati	2	4	5	6
1	(1,3)	3	-1 √	∞	6
	(3,1)	3	-1	∞	6
2	(1,4)	∞		∞	6
	(4,3)	∞		3	7
	(3,1)	3 √		∞	6
3	(1,4)			∞	6
	(4,3)			3 √	7
	(3,2)			∞	9
	(2,1)			∞	10
4	(1,4)				6 √
	(4,5)				∞
	(5,3)				∞
	(3,2)				9
	(2,1)				10

Tabella 1: Evoluzione dell'algoritmo Cheapest Insertion.

La Tabella 1 riporta la sequenza delle operazioni compiute dall'algoritmo partendo dal vertice 1. La corrispondente soluzione, di valore pari a 35, è indicata in figura 5.

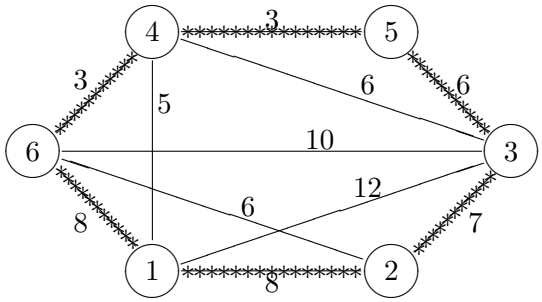


Figura 5: Algoritmo Cheapest Insertion partendo dal vertice 1.

- *Multi-Path Algorithm*
  L'algoritmo lavora in maniera analoga all'algoritmo di Kruskal per la determinazione di un albero ricoprente di costo minimo. I lati vengono ordinati secondo valori non decrescenti di costo e vengono considerati in questo ordine. Il generico arco i, j viene inserito solo se non crea un ciclo parziale e se nessuno dei due vertici associati ha già due archi incidenti. L'algoritmo termina quando sono stati selezionati n lati. Il tempo di calcolo dell'algoritmo è $O(n^2 \log n)$.
  La figura 6 mostra la corrispondente soluzione, di valore pari a 38.

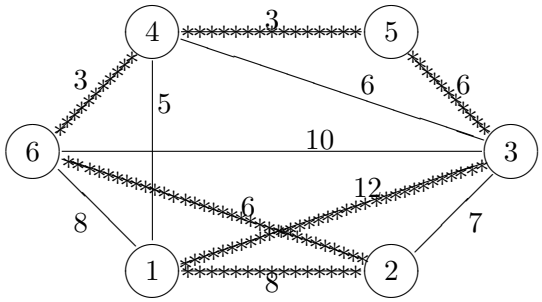


Figura 6: Algoritmo Multi-Path.

Altri algoritmi euristici possono essere ottenuti partendo da soluzioni ottime di rilassamenti del problema, cercando di modificare queste soluzioni in modo da renderle ammissibili; l'algoritmo Patching parte dalla soluzione del rilassamento per eliminazione dei SECs, cercando di "unire" a costo minimo eventuali sottocicli presenti nella soluzione rilassata.

5.2 Tour Improvement Procedures

La più naturale procedura di ricerca locale per il problema del commesso viaggiatore consiste nell'effettuare degli scambi di archi a partire da una generica soluzione s, che immaginiamo descritta mediante il vettore dei successori $\sigma(i)$ e/o quello dei predecessori $\pi(i)$ ($i \in V$). L'idea di base è quella di rimuovere dalla soluzione k archi, sostituendoli con k archi precedentemente non in soluzione.

La soluzione cui si converge utilizzando questo tipo di scambi è detta k-ottima.

Figura 7: Esempio di scambio di k archi con $k = 2$.

La figura 7 descrive una soluzione iniziale ed un possibile scambio di k archi con $k = 2$; lo scambio prevede la rimozione degli archi $(i, \sigma(i))$ e $(j, \sigma(j))$ e l'inserimento degli archi (i, j) e $(\sigma(i), \sigma(j))$. La variazione di costo a seguito della mossa è pari a

$$\Delta C := -c_{i,\sigma(i)} - c_{j,\sigma(j)} + c_{i,j} + c_{\sigma(j),\sigma(i)}$$

e la valutazione della convenienza dello scambio può essere effettuata in tempo costante, dato che dopo l'eliminazione dei due archi $(i, \sigma(i))$ e $(j, \sigma(j))$, la scelta degli archi da inserire è univoca. Il numero di scambi di questo tipo da considerare è dell'ordine $O(n^2)$.

Nel caso orientato bisogna invertire la direzione degli archi da j a $\sigma(i)$: per poter valutare la variazione del costo dello scambio in tempo costante, occorre memorizzare, per ciascun nodo $i \in V$, il costo del cammino dal nodo 1 al nodo i e quello del cammino dal nodo i al nodo 1.

In generale, la dimensione dell'intorno definito dallo scambio di k archi è $O(n^k)$. L'intorno è *connesso* già con $k = 2$. Una implementazione particolarmente efficiente dell'algoritmo è stata proposta da Lin & Kernighan, che hanno realizzato un algoritmo k-opt con k variabile (ad ogni iterazione viene deciso il numero di archi da scambiare).

Un altro tipo di intorno che si può pensare di definire riguarda lo scambio di gruppi di vertici, inserendo tra due vertici adiacenti un gruppo di vertici consecutivi. Una soluzione s può essere descritta indicando, per ciascun vertice $i \in V$, l'ordine $p(i)$ con il quale il vertice i viene visitato: un possibile intorno consiste nel considerare tutte le soluzioni ottenibili da s mantenendo inalterato il numero $p(i)$ associato ad un sottoinsieme di vertici (ad esempio quelli di indice pari), provando a re-inserire in soluzione i restanti (nelle posizioni rimaste libere). Ovviamente la dimensione dell'intorno è esponenziale, per cui si deve stabilire qualche criterio euristico per una esplorazione efficiente dell'intorno.

6 Approssimabilità del TSP

Per il problema del commesso viaggiatore non si conosce nessun algoritmo approssimato; addirittura si pensa che un tale algoritmo non possa esistere, dato che l'esistenza di un algoritmo approssimato equivarrebbe a provare che $P = NP$. Questo risultato negativo relativamente a problemi di approssimabilità è contenuto nel seguente teorema.

Teorema *Non esiste un algoritmo polinomiale A per il TSP con worst-case performance ratio $r(A)$ finita, a meno che non sia $P = NP$.*

Dim: Ragioniamo per assurdo supponendo che esista un algoritmo A per il quale si ha

$$r(A) \geq \frac{z^A(I)}{z^*(I)}$$

per ogni istanza I: dimostriamo che questo ipotetico algoritmo A sarebbe in grado di stabilire se un grafo qualunque possiede o meno un circuito hamiltoniano.
Supponendo che sia dato un qualunque grafo orientato e non pesato $G = (V, A)$, definiamo l'istanza di TSP sul grafo completo e pesato $G' = (V, A')$ nel quale i costi degli archi in A' sono definiti nel seguente modo:

$$c_{ij} = \begin{cases} 1 & \text{se } (i,j) \in A \\ r(A)\ n & \text{altrimenti} \end{cases} \qquad (i, j = 1, \ldots, n)$$

Applicando l'algoritmo A al grafo G' così definito si ha la seguente situazione: se $z^A \leq r(A)\ n$ allora deve essere $z^A = n$, dato che ciascun arco costa 1 oppure più di n. Dato poi che $z^* \leq z^A$ e che nessuna soluzione ammissibile può costare meno di n, allora $z^* = n$; questo significa che la soluzione ottima utilizza solo archi a costo unitario, cioè presenti in G, ossia che G possiede un cammino hamiltoniano.
Viceversa, supponiamo che sia $z^A > r(A)\ n$. Dato che $z^A \leq r(A)\ z^*$, allora deve essere $z^* > n$, per cui nella soluzione ottima bisogna utilizzare almeno un arco a costo non unitario, cioè non presente in G: di conseguenza G non possiede un circuito hamiltoniano. Quindi potremmo utilizzare l'algoritmo A per verificare se un qualunque grafo G possiede un circuito hamiltoniano oppure no: questo è impossibile a meno che non sia $P = NP$. □

Teorema *Sia A un algoritmo di ricerca locale per il quale l'esplorazione dell'intorno richiede tempo polinomiale. Allora, anche effettuando un numero esponenziale di iterazioni, A non può garantire di fornire una approssimazione finita, a meno che non sia $P = NP$.*

I due risultati negativi di non approssimabilità (che valgono ovviamente anche nel caso di TSP simmetrico) stabiliscono che, nel caso generale, non può esistere nessun algoritmo polinomiale con approssimazione garantita per il problema del commesso viaggiatore e che nessuna approssimazione garantita possa essere ottenuta neanche utilizzando per un numero esponenziale di iterazioni un algoritmo di ricerca locale per il quale l'intorno è polinomiale.

6.1 Caso particolare: TSP Metrico

Entrambi i risultati di non approssimabilità visti valgono per la versione generale del TSP. Esistono però dei casi particolari per i quali il problema del commesso viaggiatore (simmetrico) può essere approssimato; questo è quel che succede quando vale la *proprietà di triangolarità*, ossia quando per ogni $i, j, k \in V$ risulta $c_{ij} \leq c_{ik} + c_{kj}$ (il percorso diretto dal vertice i al vertice j costa non di più del percorso che passa per il vertice intermedio k). In tal caso si parla del problema del commesso viaggiatore metrico (indicato con ΔTSP); è da notare che il problema (sia nella versione simmetrica che in quella asimmetrica) resta NP-hard anche se vale la condizione di triangolarità. È anche da evidenziare il fatto che se il grafo è connesso (condizione indispensabile per l'esistenza di una soluzione ammissibile) e vale la condizione di triangolarità, allora il grafo è completo; quindi, sotto queste condizioni, il problema di determinare un circuito hamiltoniano è un problema "facile" (risolubile in tempo lineare).

La condizione di triangolarità è sempre verificata per le istanze euclidee, nelle quali i costi associati agli archi rappresentano delle distanze tra punti del piano; questo si verifica assai spesso nelle applicazioni reali. A volte può però accadere che i costi associati agli archi non rappresentino delle distanze, ma piuttosto dei tempi di percorrenza o dei costi (ad esempio, dei pedaggi da pagare per percorrere certe strade), o una qualche combinazione di questi parametri; oppure può accadere che alcuni di questi costi siano negativi, per modellare qualche "premio" da attribuire alle soluzioni che utilizzano certi archi. In tutte queste situazioni non è garantita la condizione di triangolarità, per cui non si possono applicare i risultati che seguono. Infine, un altro caso nel quale vale certamente la condizione di triangolarità è quello in cui, per ogni coppia di vertici i e j, la quantità c_{ij} rappresenta il costo del cammino minimo per andare dal vertice i al vertice j. Questo è quel che accade se la matrice dei costi è stata ottenuta calcolando il cammino minimo tra ogni coppia di vertici, ad esempio tramite l'algoritmo di Floyd-Warshall.

Infine è da notare che, agendi sui costi degli archi, è sempre possibile rendere triangolare qualunque grafo; aggiungendo una quantità opportuna M al costo di ciascun lato la soluzione ottima non cambia ma il grafo diventa triangolare. Però questo procedimento ha due inconvenienti: (i) ci obbliga a lavorare con lati che hanno costi elevati (il che può dare problemi numerici); (ii) anche se non cambia il cammino hamiltoniano ottimo, cambia il valore della soluzione ottima, per cui l'approssimabilità ottenuta sulla nuova istanza non è detto che valga sulla istanza di partenza (una soluzione ϵ-approssimata sulla nuova istanza potrebbe essere arbitrariamente cattiva se valutata relativamente all'istanza di partenza).

- **Algoritmo SST**

L'algoritmo SST (Shortest Spanning Tree) fornisce una approssimazione pari a 2 per il ΔTSP; l'idea di base è quella di costruire un grafo *euleriano* (nel quale cioè ogni vertice ha grado pari) a partire da un albero ricoprente di costo minimo.

1. determina un albero ricoprente T di costo minimo;
2. duplica ciascun lato di T, ottenendo un grafo euleriano;
3. definisci un cammino euleriano sul grafo appena creato;
4. costruisci un cammino ammissibile che visita ciascun nodo una ed una sola volta utilizzando degli "shortcut".

Teorema *Per l'algoritmo SST si ha* $r(SST) = 2$.

Dim: Ogni soluzione ammissibile del TSP è costituita da un albero ricoprente cui viene aggiunto un lato; pertanto un lower bound valido sul valore della soluzione ottima è dato dal valore di un albero ricoprente T di costo minimo. Quindi, per una qualunque istanza I di TSP, il costo $CT(I)$ di uno shortest spanning tree è tale che $CT(I) \leq z^*(I)$. Se duplichiamo i lati dell'albero ricoprente T trovato, otteniamo un grafo euleriano, per il quale è immediato trovare un ciclo: basta infatti partire da un nodo qualunque (ad esempio, il vertice 1) e seguire un lato uscente dal nodo, evitando di utilizzare due volte lo stesso lato. Il costo di tale ciclo è pari a 2 $CT(I)$ ma la soluzione trovata non è ammissibile per TSP in quanto ciascun vertice è visitato due volte. Per rendere la soluzione ammissibile si utilizza la proprietà di triangolarità e si effettuano degli *shortcut*: ogni volta che ci si dovrebbe

recare ad un vertice che è già stato visitato, si salta il vertice e ci si reca al successivo vertice non ancora visitato. È facile verificare che, se vale la condizione di triangolarità, il costo del cammino così ottenuto non può mai aumentare e che quindi la soluzione trovata ha costo

$$z^H(I) \leq 2\ CT(I) \leq 2\ z^*(I)$$

L'algoritmo richiede tempo $O(n^2)$ per trovare un albero ricoprente di costo minimo; le altre operazioni possono essere svolte in tempo $O(n)$.

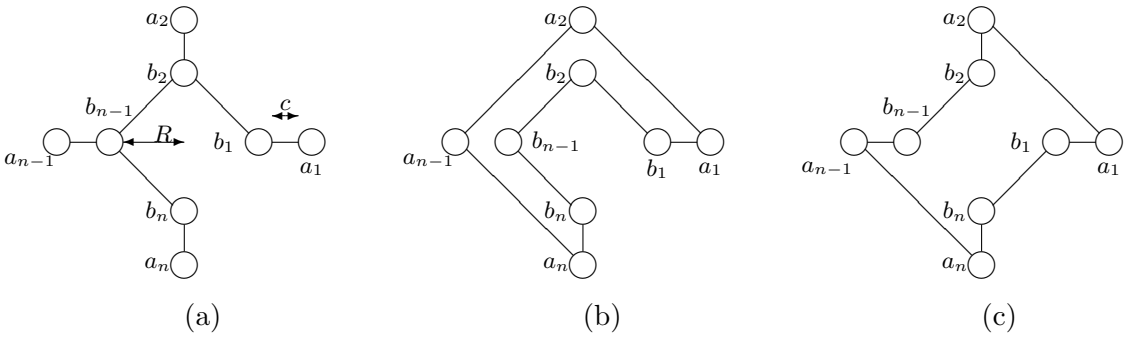


Figura 8: Istanza critica per l'algoritmo SST.

La figura 8 presenta un esempio di istanza su grafo completo con distanze euclidee per la quale l'algoritmo SST può raggiungere la propria worst case. Il grafo ha $2n$ nodi (con n pari):

- n nodi di tipo a; il generico nodo a_i ha coordinate $((R+c)\cdot\cos(i\frac{2\pi}{n}),\ (R+c)\cdot\sin(i\frac{2\pi}{n}))$;
- n nodi di tipo b; il generico nodo b_i ha coordinate $(R\cdot\cos(i\frac{2\pi}{n}),\ R\cdot\sin(i\frac{2\pi}{n}))$.

Un albero ricoprente di costo minimo è riportato nella parte (a) della figura, mentre le parti (b) e (c) mostrano una soluzione prodotta da SST e la soluzione ottima, rispettivamente. La lunghezza di ciascun lato (b_i, b_{i+1}) e (a_i, a_{i+1}) $(i = 1, \ldots, n-1)$ è pari a $2R\sin(\pi/n)$ e $2(R+c)\sin(\pi/n)$, rispettivamente, per cui il valore della soluzione euristica è

$$z^H = 2(n-1)(2R+c)\sin(\pi/n) + 2c$$

mentre la soluzione ottima vale

$$z^* = n(2R+c)\sin(\pi/n) + nc$$

Prendendo $R = 1$ e $c = \dfrac{1}{n^2}$ si ha

$$\lim_{n\to\infty} \frac{z^H}{z^*} = \frac{4\pi}{2\pi} = 2 \qquad \square$$

- **Algoritmo di Christofiedes**

Per migliorare l'approssimazione fornita dall'algoritmo SST, l'idea di base è che, per creare un grafo euleriano a partire dall'albero ricoprente T di costo minimo trovato al punto 1., non occorre duplicare tutti i lati, ma basta rendere pari il grado dei vertici che hanno grado dispari in T. È da notare che il numero di vertici con grado dispari è sempre pari, in quanto il grado complessivo di T è pari.

Quello che occorre fare per migliorare l'algoritmo SST è generare un grafo euleriano migliore. Non possiamo pretendere di determinare il miglior grafo euleriano, in quanto questo equivarrebbe a determinare il ciclo ottimo (per via della disuguaglianza triangolare). L'idea è quella di trovare il matching ottimo (di peso minimo) tra tutti i vertici che hanno grado dispari. Questa operazione, che può essere effettuata con opportuni algoritmi in tempo $O(n^3)$, permette di identificare un insieme $M = \{(i_1, i_2), (i_3, i_4), \ldots, (i_{2p-1}, i_{2p})\}$ di lati che, se aggiunti all'albero T, formano un grafo euleriano. L'algoritmo di Christofiedes funziona nel seguente modo:

1. determina un albero ricoprente T di costo minimo;
2. trova il matching perfetto di peso minimo sul grafo indotto dai vertici con grado dispari in T, ed aggiungi a T i lati del matching ottimo, così da ottenere un grafo euleriano;
3. definisci un cammino euleriano sul grafo appena creato;
4. costruisci un cammino ammissibile che visita ciascun nodo una ed una sola volta utilizzando degli "shortcut".

Teorema *Per l'algoritmo di Christofides si ha* $r(A) = 3/2$.

Dim: Data una istanza I di TSP, consideriamo i vertici $i_1, i_2, \ldots, i_{2p}$ coinvolti nel matching; questi vertici sono quelli di grado dispari nell'albero T trovato, ad esempio quelli evidenziati in figura 9.

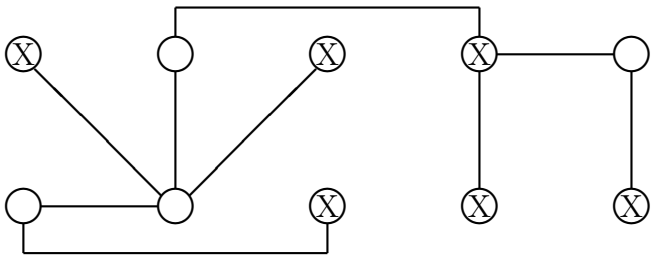


Figura 9: Possibile albero ricoprente e vertici coinvolti nel matching.

L'ordine con cui questi vertici sono visitati nella soluzione ottima del problema produce due matching ammissibili $M_1(I)$ e $M_2(I)$ (in generale diversi dal matching $M(I)$ di costo minimo), come mostrato in figura 10. Dato che i vertici sono visitati nello stesso ordine con

il quale sono visitati nella soluzione ottima e che vale la proprietà di triangolarità, segue che

$$CM_1(I) + CM_2(I) \leq z^*(I)$$

Considerando il matching $M(I)$ di costo minimo, si ha che

$$CM(I) \leq CM_1(I) \qquad \text{e} \qquad CM(I) \leq CM_2(I)$$

e quindi

$$2\ CM(I) \leq CM_1(I) + CM_2(I) \leq z^*(I) \qquad \Rightarrow \qquad CM(I) \leq z^*(I)/2$$

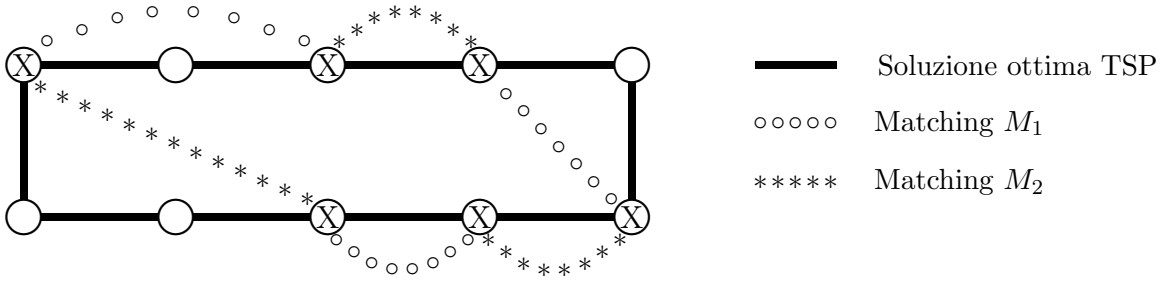


Figura 10: Utilizzo del matching nell'algoritmo di Christofidies.

Aggiungendo a T il matching di costo minimo nel sottografo indotto dai vertici di grado dispari in T, si ottiene un grafo euleriano che, se percorso completamente, ha costo pari a

$$CT(I) + CM(I) \leq z^*(I) + \frac{z^*(I)}{2} = \frac{3}{2} z^*(I)$$

da cui segue una soluzione ammissibile di valore non superiore a $\frac{3}{2} z^*(I)$ e la dimostrazione che la ratio dell'algoritmo non può eccedere 3/2.

La figura 11 presenta un esempio di istanza su grafo completo con distanze euclidee per la quale l'algoritmo di Christofides può raggiungere la propria worst case. Il grafo ha $2n + 1$ nodi:

- $n + 1$ nodi di tipo a; il generico nodo a_i ha coordinate $(i,\ 0)$;
- n nodi di tipo b; il generico nodo b_i ha coordinate $(i + \frac{1}{2}, \frac{\sqrt{3}}{2})$.

Un albero ricoprente di costo minimo è riportato nella parte (a) della figura; a questo viene aggiunto, tramite soluzione di un problema di matching triviale, il lato a_1, a_{n+1}, per cui il valore della soluzione euristica è pari a

$$z^H = 2n + n = 3n$$

La parte (b) della figura mostra la soluzione ottima, di valore pari a

$$z^* = 1 + (n - 1) + 1 + n = 2n + 1$$

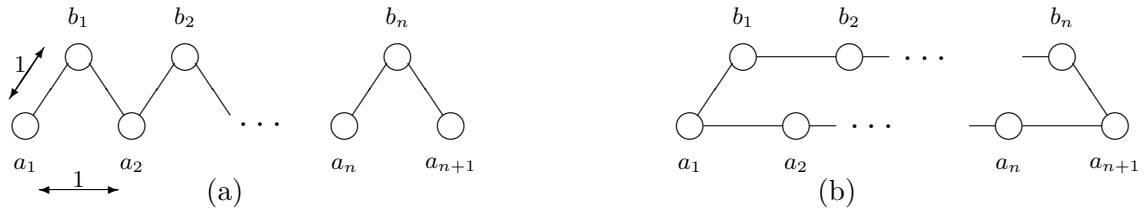


Figura 11: Istanza critica per l'algoritmo di Christofides.

per cui il rapporto $\frac{z^H}{z^*}$ è arbitrariamente vicino a 3/2 per n sufficientemente grande. □

Quindi l'algoritmo di Christofidies porta ad una approssimazione pari a 3/2 per il ΔTSP. È da ricordare che invece, nel caso generale, non esiste nessun algoritmo approssimato per il TSP, neanche con rapporto di caso peggiore pari a 1000.
Fermo restando che attualmente non si conosce nessun algoritmo polinomiale in grado di fornire una approssimazione migliore di quella dell'algoritmo di Christofides, resta ancora in piedi la seguente questione: con quale criterio effettuare gli shortcut? È stato provato che il problema di trovare il miglior insieme di shortcut è un problema NP-difficile.