Parte II:

Ottimalità, rilassamenti e bound

Ottimalità, rilassamenti e bound

Consideriamo il seguente problema

$z^* = \max \{c^T x : x \in X, X \subseteq \{0,1\}^n\}$

dove z^* è il valore della soluzione ottima x^*.

<u>Domanda</u>: In che modo è possibile certificare che la soluzione x^* è ottima?

In generale, se disponessimo di un algoritmo che genera le due sequenze di soluzioni:

$$z_1^{UB} \geq z_2^{UB} \geq \ldots \geq z_h^{UB} \geq z^*$$

$$z_1^{LB} \leq z_2^{LB} \leq \ldots \leq z_k^{LB} \leq z^*$$

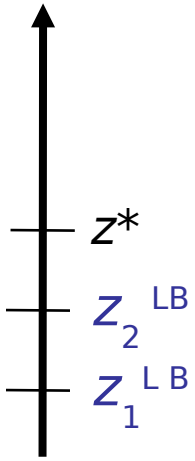
potremmo fornire come criterio di arresto

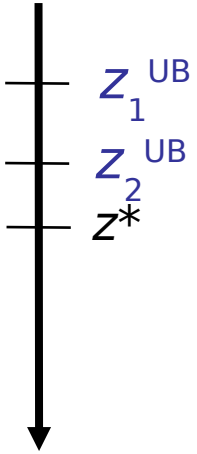
$$z_h^{UB} - z_k^{LB} \leq \varepsilon \ (> 0)$$

Lower (upper) bound

Bound primali

Ogni soluzione $x \in X$ ammissibile è un *lower* (*upper*) *bound* per un problema di massimizzazione (minimizzazione)





Upper (lower) bound

Bound duali

Al contrario dei bound *primali*, trovare upper (lower) bound di buona qualità per problemi di massimo (minimo) è tipicamente difficile.

Buoni bound *duali* si ottengono attraverso lo studio delle proprietà strutturali del problema di OC.

Le proprietà di un problema di OC si caratterizzano tramite:

1. Rilassamenti del problema
2. Definizione e studio di problemi *duali*.

Bound duali per il TSP

Rilassamento

Definizione: Il problema

(RP) $z^R = \max \{f(x) : x \in T, T \subseteq \boldsymbol{R}^n\}$ ($z^R = \min \{f(x) : x \in T, T \subseteq \boldsymbol{R}^n\}$)

si definisce rilassamento del problema

(P) $z = \max \{c^T x : x \in X, X \subseteq \{0,1\}^n\}$ ($z = \min \{c^T x : x \in X, X \subseteq \{0,1\}^n\}$)

se e solo se:

i) $X \subseteq T$,

ii) $f(x) \geq c^T x$ per ogni $x \in X$ ($f(x) \leq c^T x$ per ogni $x \in X$)

Proprietà 1: Se (RP) è un rilassamento di (P), allora $z^R \geq z^*$ ($z^R \leq z^*$).

Proprietà 2: Se x^R sol. ottima di (RP) è ammissibile per (P) allora $x^R = x^*$.

Esempi di rilassamenti

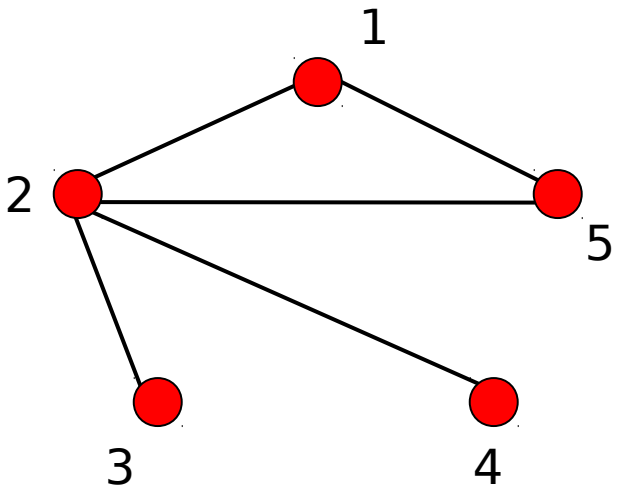
Problema del commesso viaggiatore simmetrico (archi non orientati).

Dati: grafo $G = (V, E)$; pesi sugli archi c_e per ogni arco $e \in E$.

Domanda: trovare un *ciclo hamiltoniano* (ciclo che tocca tutti i nodi del grafo una ed una sola volta) di peso minimo.

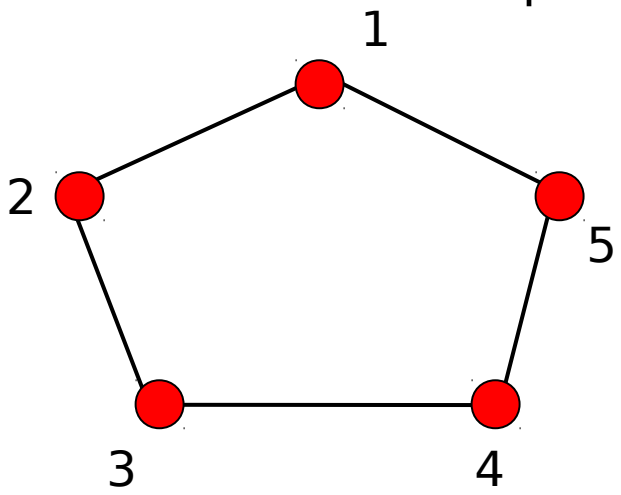
Definizione: Un 1-*albero* è un sottografo di G che consiste di due archi adiacenti al nodo 1 più gli archi di un albero ricoprente i nodi $\{2, \ldots, n\}$.

Esempio



Rilassamenti per il TSP

Osservazione: Un ciclo hamiltoniano è un particolare 1-albero.



Pertanto, il problema di determinare un 1-albero di peso minimo su un grafo $G = (V,E)$ con peso c_e associato ad ogni arco $e \in E$ è un rilassamento del problema del TSP. Infatti

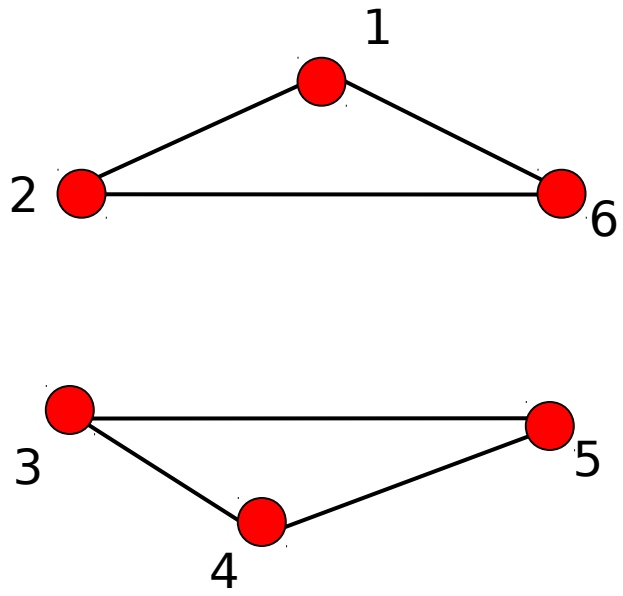
- L'insieme X di tutti i cicli hamiltoniani è contenuto nell'insieme T di tutti gli 1-alberi
- Poiché $T \supseteq X$ il valore della soluzione trovata $f(T)$ sarà sempre $\leq$ al valore della soluzione ottima del TSP $f(X)$.

Rilassamenti per il TSP

Il problema di determinare un 1-albero di peso minimo su un grafo è risolvibile in tempo polinomiale.

Rilassamenti per il TSP

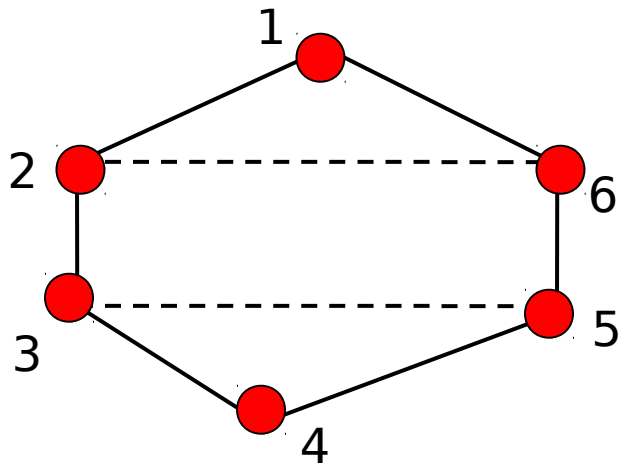
Definizione: Dato un grafo $G = (V, E)$, un 2-*abbinamento* su G è un insieme di archi M tale che ogni nodo in V è estremo di esattamente due archi in M.



Un 2-abbinamento forma un insieme di cicli disgiunti.

Rilassamenti per il TSP

Osservazione: Un ciclo hamiltoniano è un particolare 2-abbinamento, difatti è un 2-abbinamento privo di sottocicli (subtour).



Pertanto, il problema di trovare un 2-abbinamento di peso minimo su un grafo $G = (V,E)$ con peso c_e associato ad ogni arco $e \in E$ è un rilassamento del problema del TSP. Infatti

- L'insieme X di tutti i cicli hamiltoniani è contenuto nell'insieme T di tutti i 2-abbinamenti.
- Poiché $T \supseteq X$ il valore della soluzione trovata $f(T)$ sarà sempre $\leq$ al valore della soluzione ottima del TSP $f(X)$.

Rilassamenti per il TSP

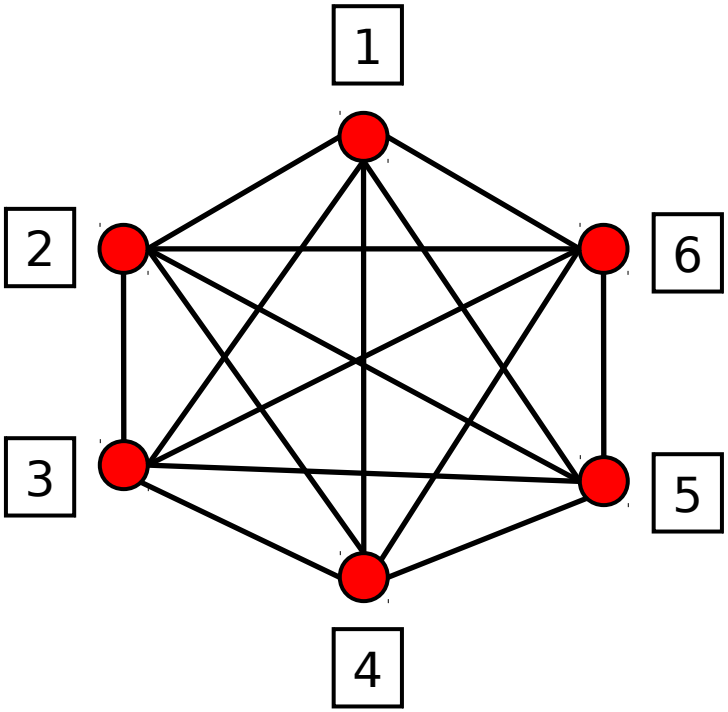
Il problema di determinare un 2-abbinamento di peso minimo su un grafo è risolvibile in tempo polinomiale.

Un esempio

Consideriamo la seguente istanza del problema del TSP (grafo completo):

	1	2	3	4	5	6
1	-	1	99	99	99	1
2	1	-	10	99	99	1
3	99	10	-	1	1	99
4	99	99	1	-	1	99
5	99	99	1	1	-	10
6	1	1	99	99		-

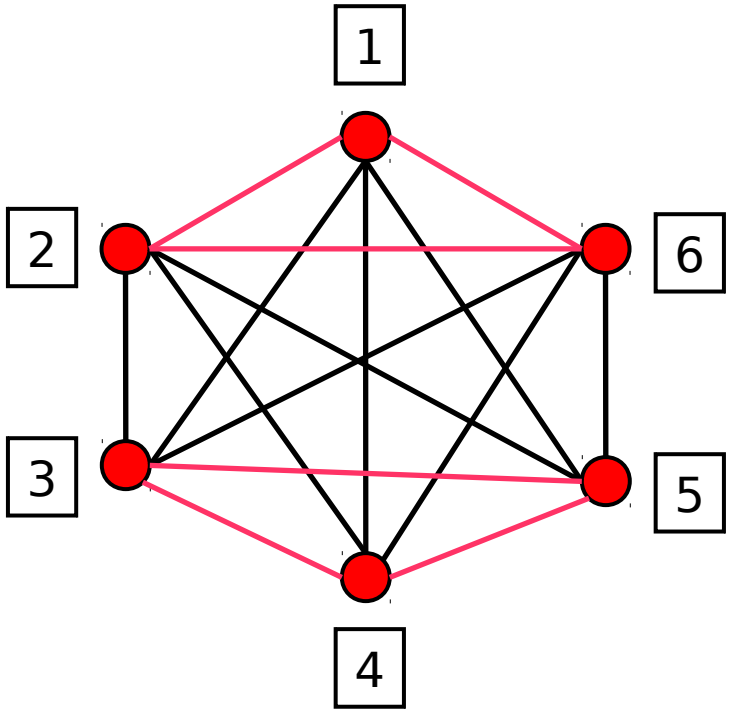
Un esempio



	1	2	3	4	5	6
1	-	1	99	99	99	1
2	1	-	10	99	99	1
3	99	10	-	1	1	99
4	99	99	1	-	1	99
5	99	99	1	1	-	10
6	1	1	99	99	10	-

Un esempio

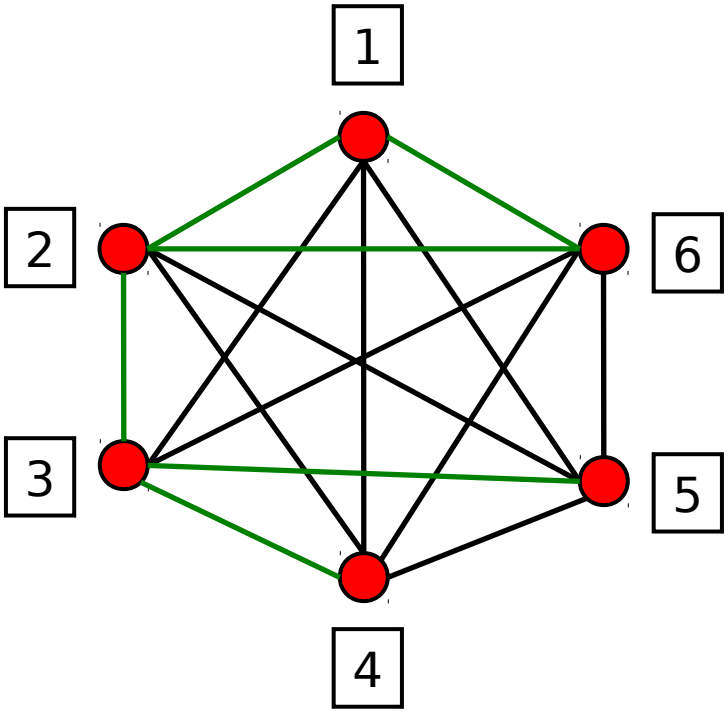
Il 2-abbinamento di peso minimo ha valore 6.



	1	2	3	4	5	6
1	-	1	99	99	99	1
2	1	-	10	99	99	1
3	99	10	-	1	1	99
4	99	99	1	-	1	99
5	99	99	1	1	-	10
6	1	1	99	99	10	-

Un esempio

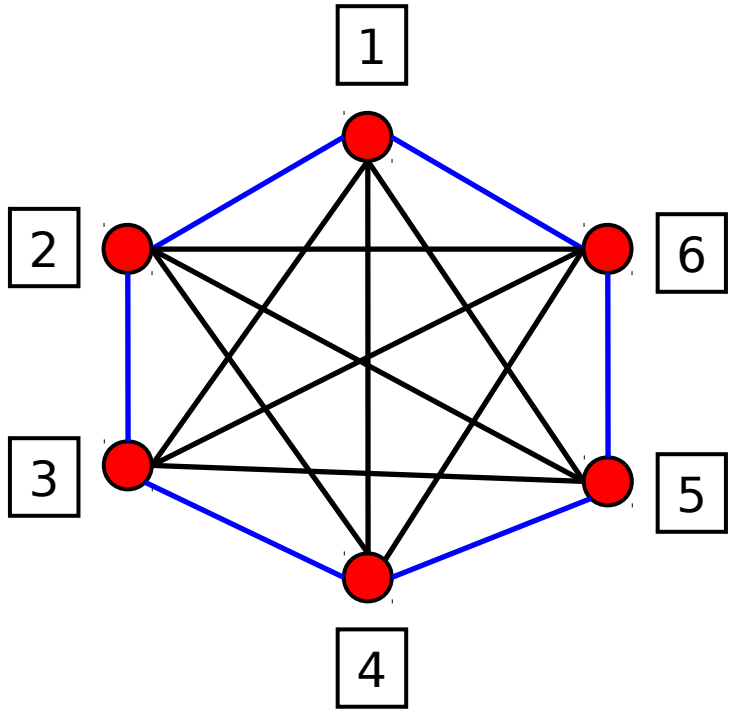
L’1-albero di peso minimo ha valore 15.



	1	2	3	4	5	6
1	-	1	99	99	99	1
2	1	-	10	99	99	1
3	99	10	-	1	1	99
4	99	99	1	-	1	99
5	99	99	1	1	-	10
6	1	1	99	99	10	-

Un esempio

Il ciclo hamiltoniano di peso minimo ha valore 24.



	1	2	3	4	5	6
1	-	1	99	99	99	1
2	1	-	10	99	99	1
3	99	10	-	1	1	99
4	99	99	1	-	1	99
5	99	99	1	1	-	10
6	1	1	99	99	10	-

Ricapitolando…

Abbiamo definito per il TSP due lower bound con le seguenti proprietà:

1. Sono lower bound “combinatorici”, ovvero si ottengono tramite la soluzione di un problema di OC.
2. Tutti e due i problemi la cui soluzione genera i lower bound sono “facili”, ovvero hanno complessità polinomiale.

La proprietà 2. è una proprietà chiave per ogni bound duale.

Infatti, se il calcolo del bound fosse un problema NP-completo sappiamo che calcolare il bound diventa almeno tanto difficile quanto risolvere il problema stesso.

Domanda: Come si possono calcolare bound primali (ovvero soluzioni ammissibili di buona qualità) per il problema del TSP?

Bound primali per il TSP

Algoritmi euristici

Vogliamo individuare un metodo generale per calcolare bound primali (ovvero soluzioni ammissibili di buona qualità) per il problema del TSP.

Definizione: Un algoritmo *A* si dice *euristico* per un problema *P* se restituisce una soluzione ammissibile z^A che non è garantito essere la soluzione ottima.

Definizione: Sia *P* un problema in forma di minimo. Un algoritmo euristico *A* si dice δ-*approssimato* se

1. ha complessità polinomiale, e
2. per ogni istanza *I* di *P* con soluzione ottima $z^*(I)$, si ha

$$z^A(I) / z^*(I) \leq \delta \qquad (1)$$

Osservazione: Se *P* è un problema di massimo, allora $\delta \leq 1$ e la condizione (1) vale con il segno di $\geq$.

Euristiche per il TSP

Sia $G = (V, E)$ un grafo completo e sia c_{uv} il costo di ciascun arco $uv \in E$.

Euristiche costruttive: tentano di costruire un “buon” ciclo hamiltoniano a partire da un sottociclo eventualmente vuoto.

Euristiche migliorative: a partire da una soluzione ammissibile, si tenta di migliorarla attraverso miglioramenti “locali”.

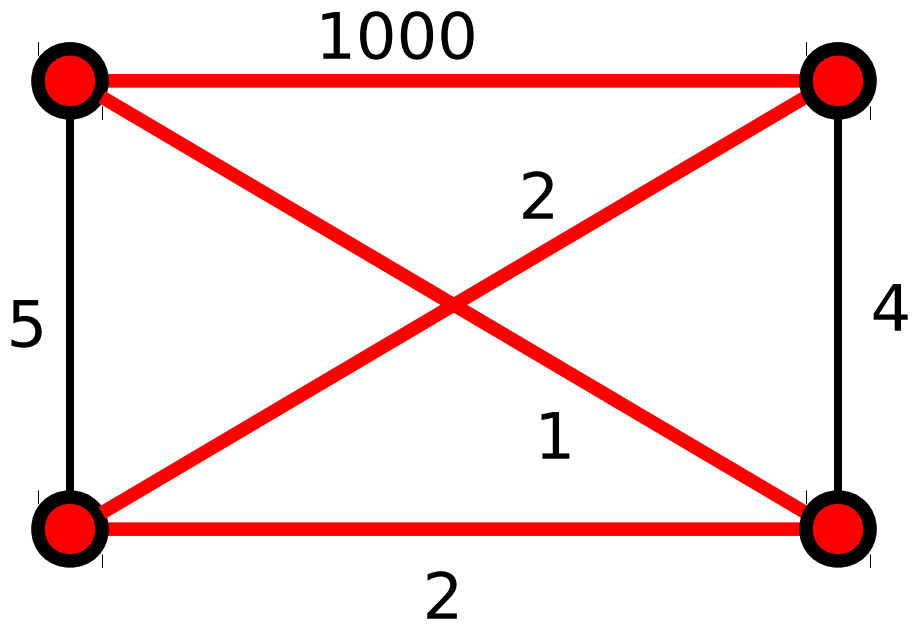
Euristica *Nearest Neighbor*

Input: $G = (V,E)$.

Output: ciclo hamiltoniano C_H.

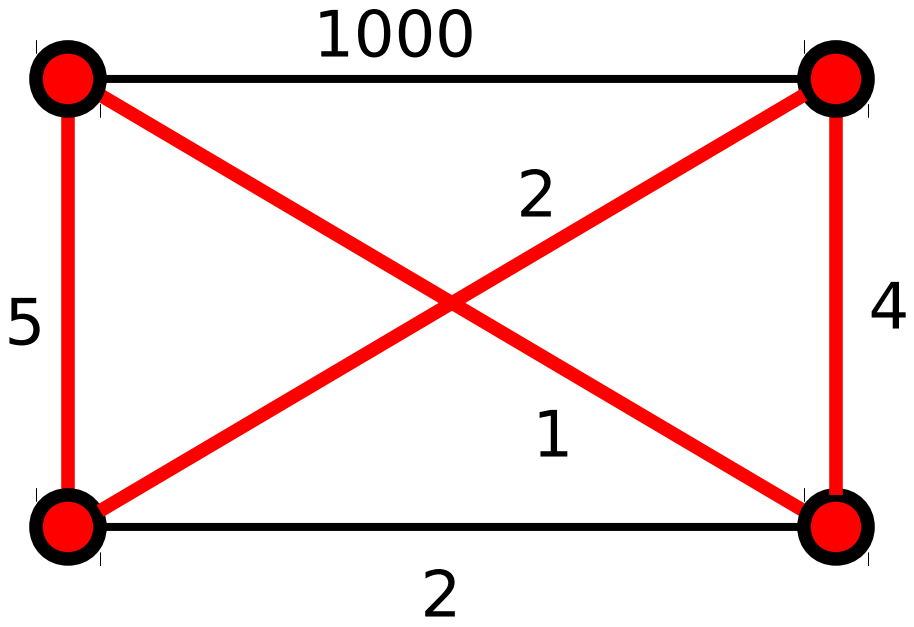
```
procedure nearest_neighbor ()

   Scegli un vertice u ∈ V qualsiasi;

   W=V\{u}, aggiungi u a C_H

   while |W| > 0 {
       scegli v ∈ W tale che c_uv = min {c_uj : j ∈ W}
       e v non forma sottocicli in W
       Aggiungi {v} a C_H
       W = W \ {v}
       u = v
   }
```

Esempio *Nearest Neighbor*



La soluzione di valore 1005 è ottima?

Esempio *Nearest Neighbor*



La soluzione ottima vale 12.

Osservazione: il rapporto $z^A(I)/z^*(I)$ può essere grande a piacere.

Euristiche di inserimento

Input: $G=(V,E)$.

Output: ciclo hamiltoniano C_H.

```
procedure insertion_heuristic ()
        Inizializza C_H con un sottociclo
        W = V/C_H;
        while (|W| > 0) {
                scegli un vertice u ∈ W;
                scegli la posizione in cui inserire u in C_H;
                inserisci u in C_H;
                elimina u da W;
        }
```

Selezione del vertice da inserire

Definizione: Si definisce distanza di un vertice u da un ciclo C_H il peso del più piccolo arco che collega il vertice ad un altro qualsiasi vertice del ciclo.

$$C_H = (1, 2, .., k) \Rightarrow \text{dist}\,(u, C_H) = \min\{\, c_{uv} : v \in C_H \}.$$

Nearest Insertion:

Inserisci il vertice u che minimizza dist (u, C_H), $u \notin C_H$.

Farthest Insertion:

Inserisci il vertice u che massimizza dist (u, C_H), $u \notin C_H$.

Selezione della posizione di inserimento

Osservazione: Scegliere la posizione equivale a scegliere l'arco da eliminare nel ciclo C_H. Un vertice può essere inserito in ogni posizione di C_H.

Siano $C_H = (v_1, v_2, \ldots, v_n)$ il ciclo corrente e $c(C_H)$ il costo di C_H e sia u il nodo da aggiungere al ciclo.

Se $c(C_H(i))$ è il costo del ciclo ottenuto da C_H inserendo il nodo u nella posizione i-esima, il costo di inserimento è pari a

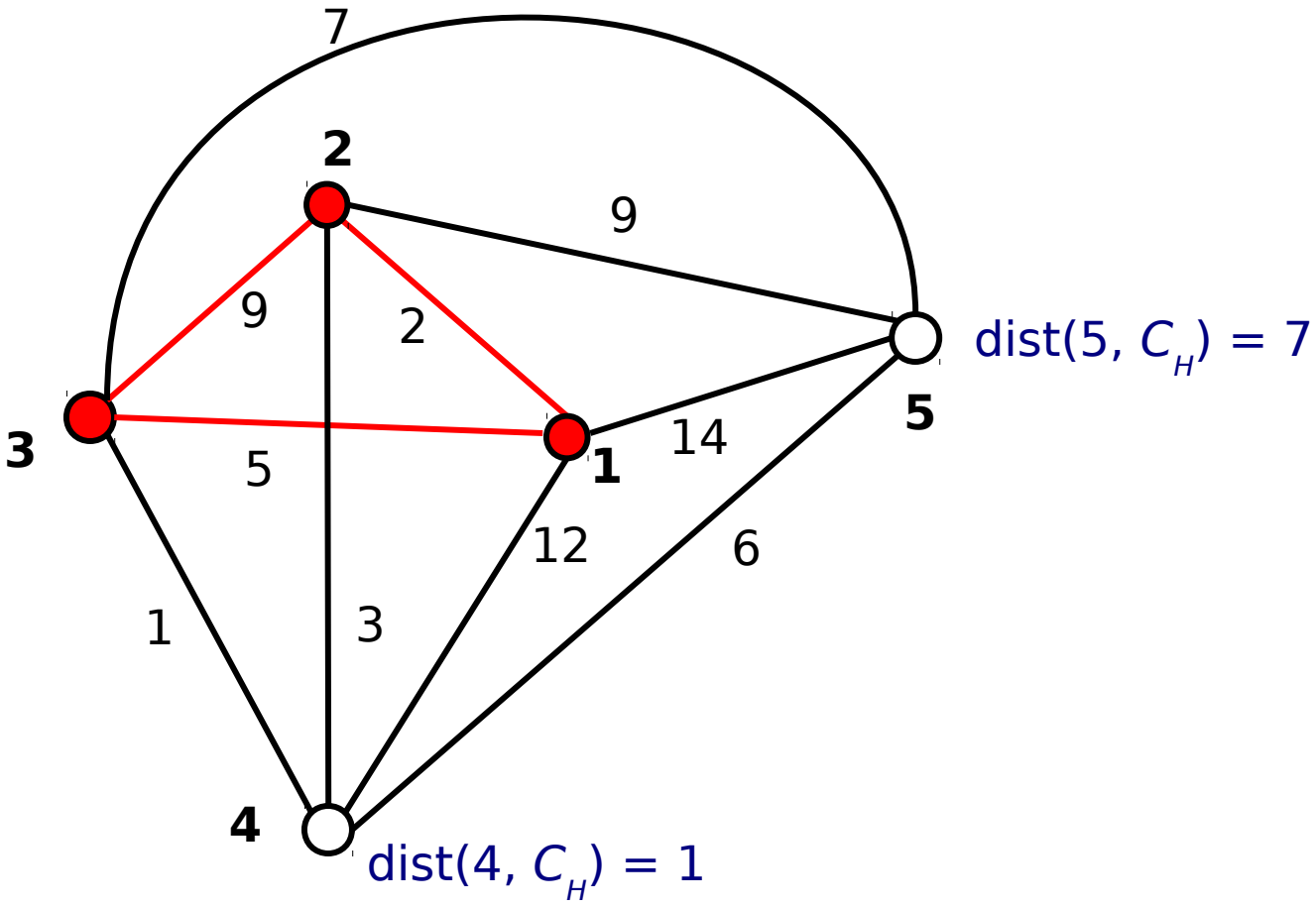
$$c_i = c(C_H(i)) - c(C_H).$$

Si seleziona la posizione che minimizza c_i.

Esempio

$C_H = \{1,2,3\}$

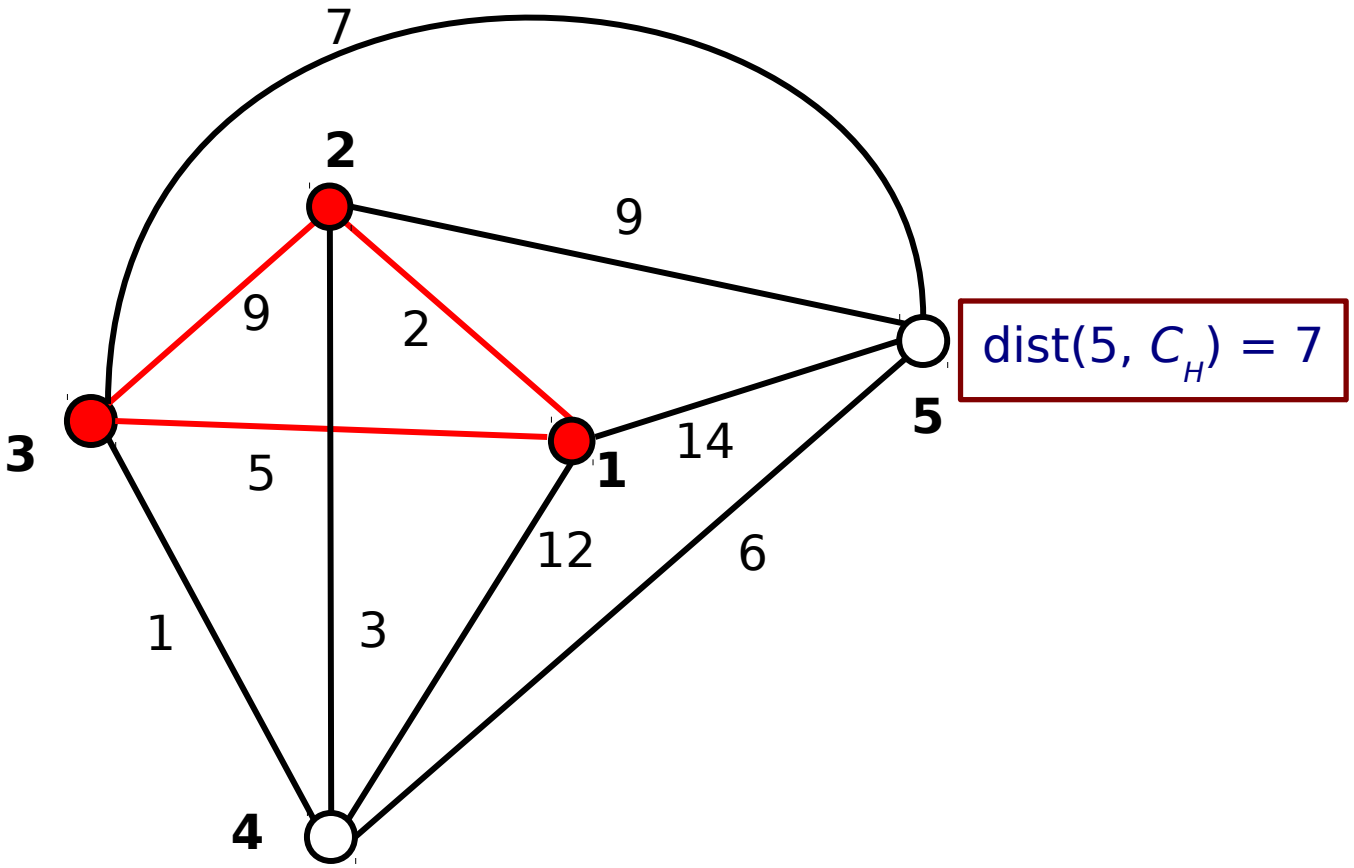
$c(C_H) = 16$



Esempio

$C_H = \{1,2,3\}$

$c(C_H) = 16$



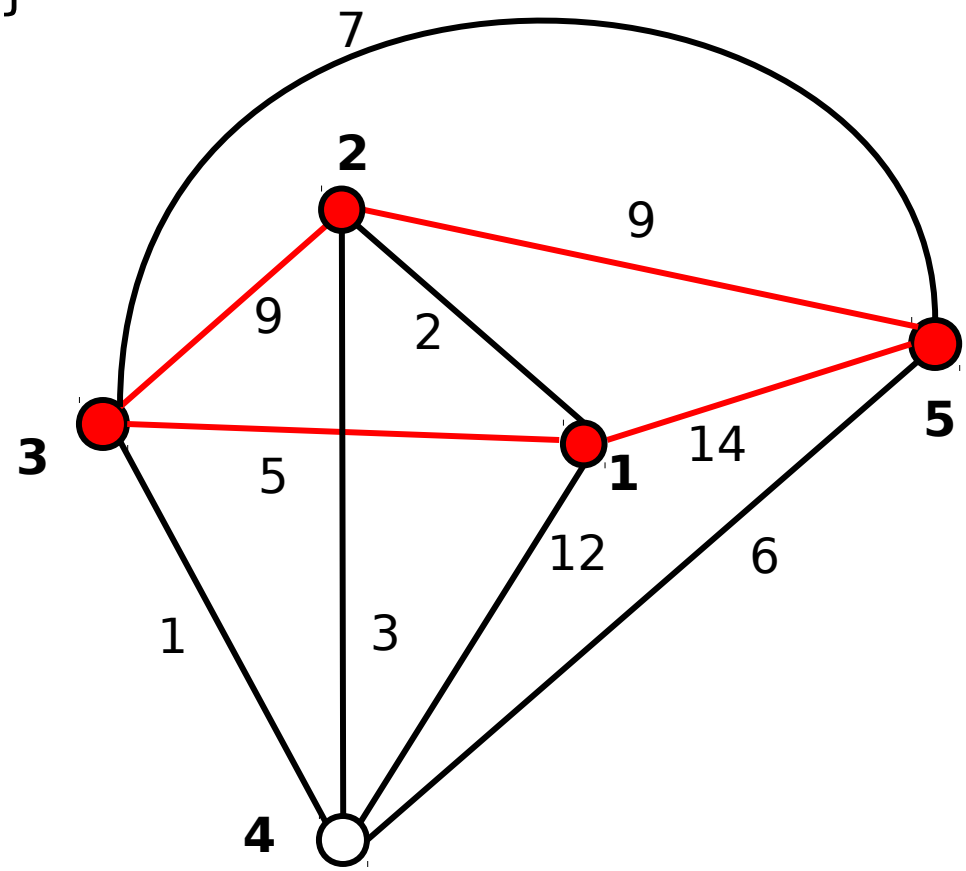
Esempio

$C_H = \{1,2,3,5\}$

$c(C_H(2)) = 37$

$c(C_H) = 16$

$c_I = 21$



Tentativo 1:

14 + 9 - 2 = 21

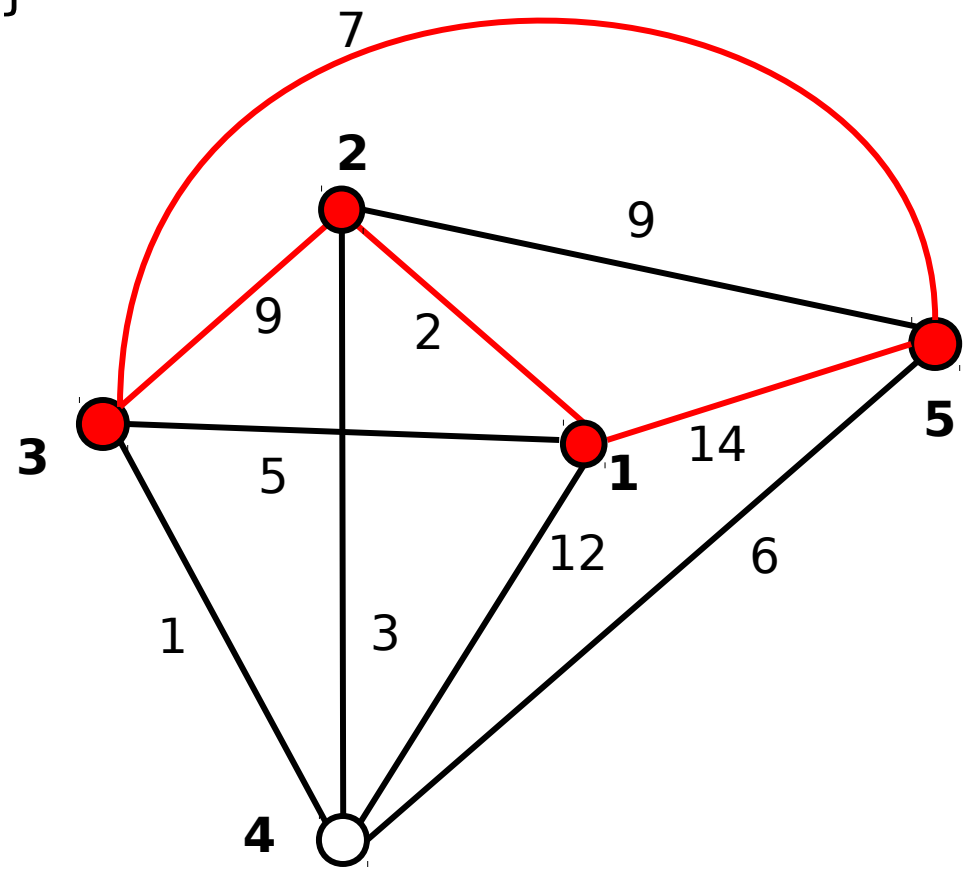
Esempio

$C_H = \{1,2,3,5\}$

$c(C_H(4)) = 32$

$c(C_H) = 16$

$c_I = 16$



Tentativo 2:

14 + 7 - 5 = 16

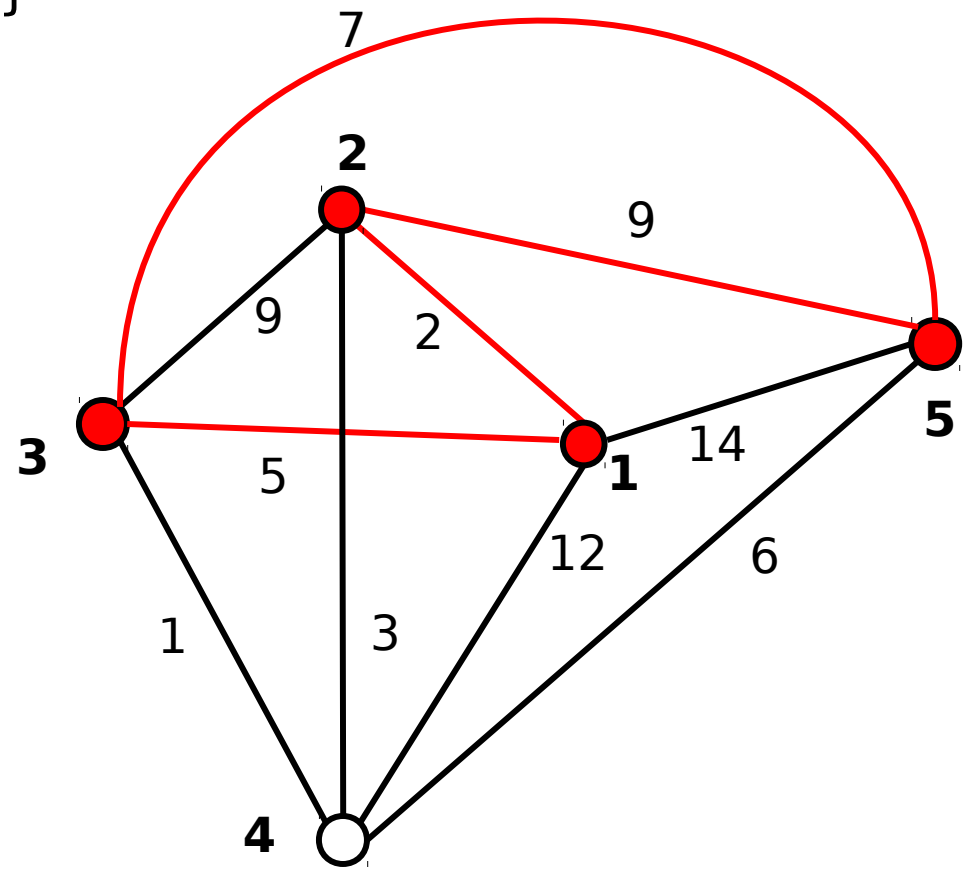
Esempio

$C_H = \{1,2,3,5\}$

$c(C_H(3)) = 23$

$c(C_H) = 16$

$c_I = 7$

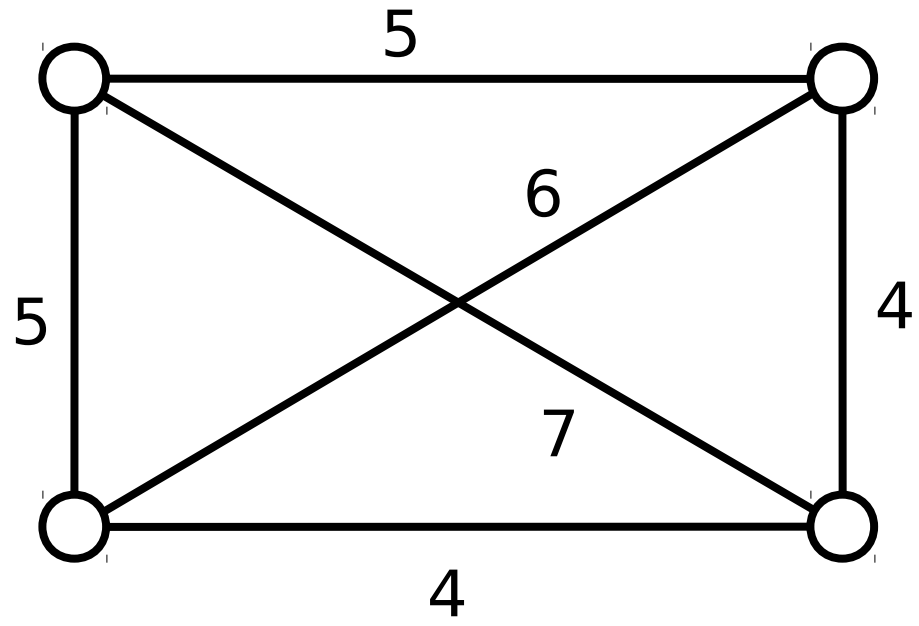


Tentativo 3:

9 + 7 - 9 = 7

Una proprietà strutturale

Si dice che la matrice delle distanze di un grafo G soddisfa la *disuguaglianza triangolare* se per ogni terna di nodi i, j, k in G, $i \neq j \neq k$, si ha $c_{ij} + c_{jk} \geq c_{ik}$.



Richiamo

Definizione: Un multigrafo $G = (V, E)$ (cioè un grafo con ripetizione di archi) è *euleriano* se, a partire da un vertice v di G, è possibile costruire un ciclo che inizia e finisce in v e che attraversa ogni arco esattamente una volta.

Teorema: Un multigrafo $G = (V, E)$ è euleriano se e solo se

1. G è connesso, e
2. tutti i nodi di G hanno grado pari.

Richiamo

Dimostrazione: ($\Rightarrow$) Se G è euleriano allora contiene un ciclo euleriano. Necessariamente deve essere connesso (altrimenti non potrei partire da un nodo in una componente connessa e tornare nello stesso nodo passando per altre componenti connesse) ed avere nodi di grado pari.

($\Leftarrow$) Sia G connesso con tutti i nodi di grado pari. Procediamo per induzione

PASSO BASE: $E = \emptyset$. Sicuramente G è euleriano.

PASSO GENERICO: Supponiamo che tutti i multigrafi $G' = (V' E')$ con $|E'| < |E|$ siano euleriani. Scegliamo un nodo $v \in V$. Partendo da v, costruiamo un ciclo senza passare mai su un arco più di una volta (questo è possibile perché ogni nodo in G ha grado pari).

Richiamo

($\Leftarrow$) Cancelliamo da G gli archi del ciclo appena calcolato e consideriamo le componenti connesse ottenute. Ognuna di tali componenti

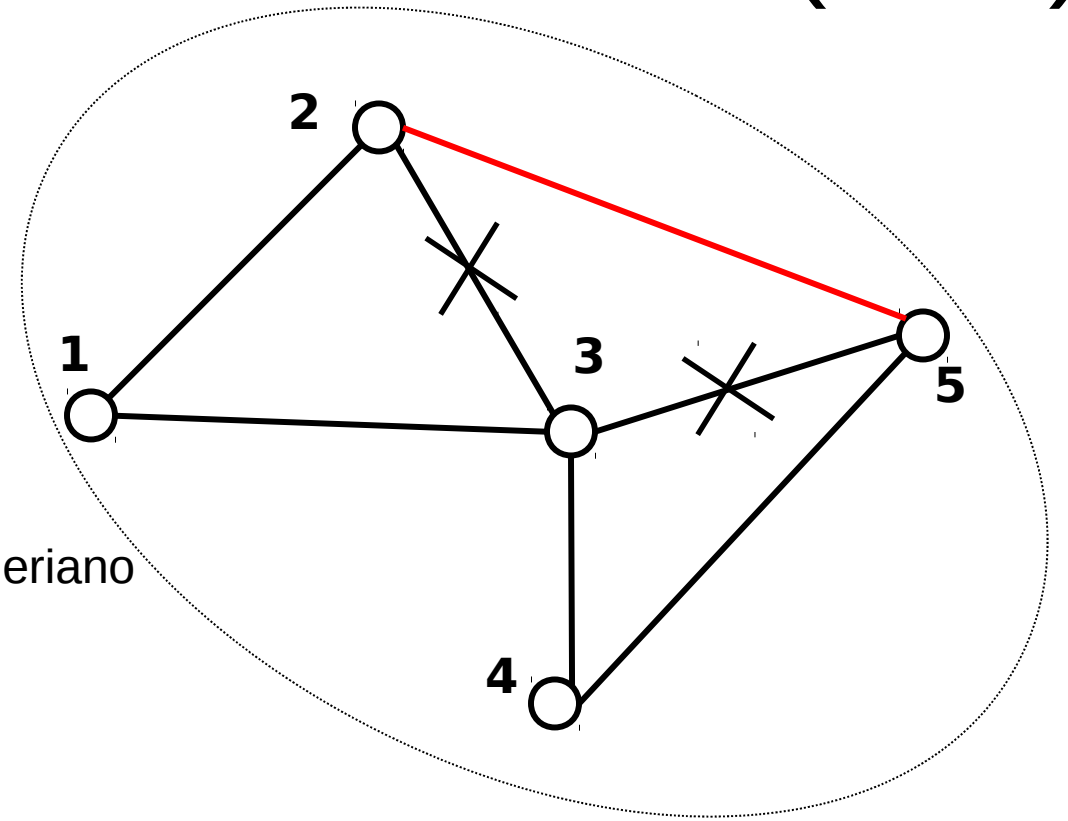
- soddisfa le condizioni 1. e 2.
- è euleriana (dall'ipotesi induttiva).

Ricomponendo i percorsi euleriani di ciascuna componente connessa otteniamo un ciclo euleriano su G. Pertanto possiamo concludere che G è euleriano.

Richiamo

Teorema: Sia $H = (V, F)$ un grafo completo con la matrice dei costi che soddisfa la disuguaglianza triangolare. Sia $G = (V,E)$ un sottografo euleriano connesso di H. Allora H contiene un ciclo hamiltoniano di lunghezza al più $\sum_{e \in E} c_e$.

Dimostrazione (idea)



Sottografo euleriano

Come posso ricavare un ciclo hamiltoniano?

Poiché vale la disuguaglianza triangolare, $c_{25} \leq c_{23} + c_{35}$.

Euristica *Double Tree*

1. Calcola un minimo albero ricoprente *K* (Kruskal/Prim).

2. Raddoppia gli archi di *K*, formando un ciclo euleriano.

3. Ricava un ciclo hamiltoniano dal ciclo euleriano.

Indichiamo con Z^H_{DT} il valore della soluzione che si ottiene applicando l'euristica Double Tree.

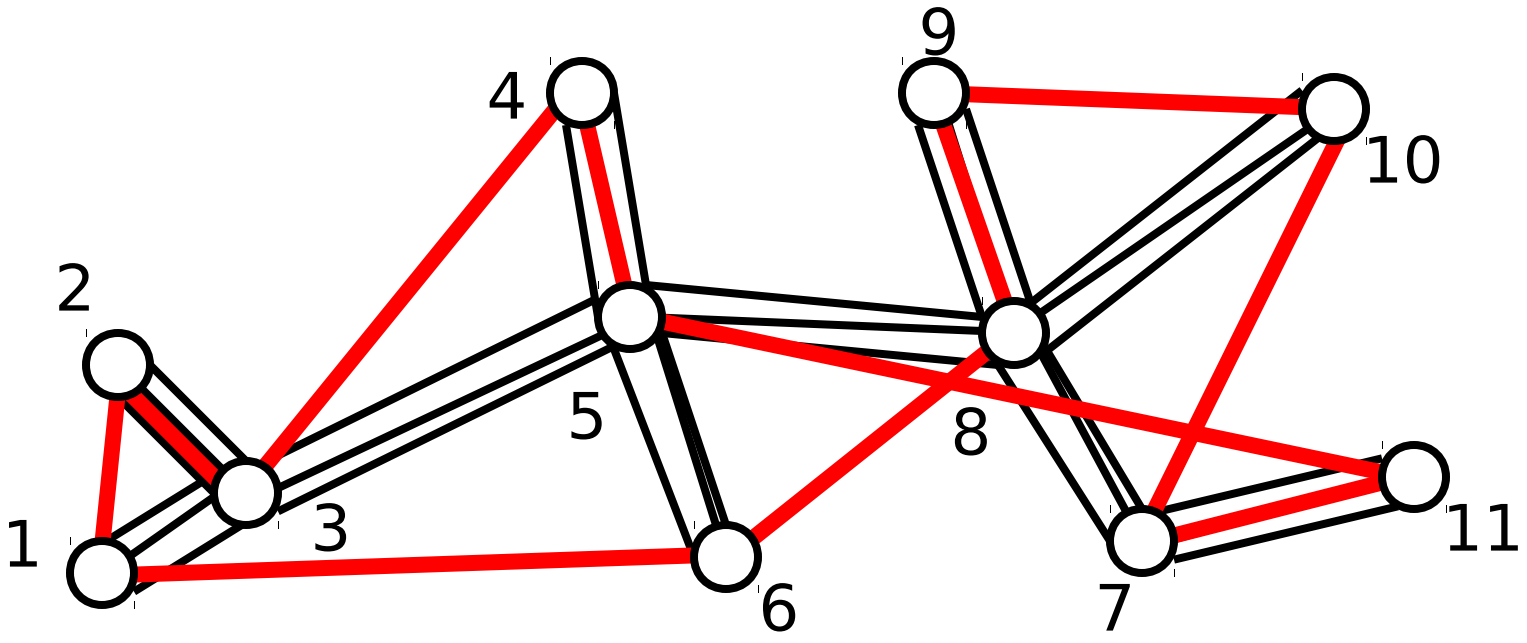
Da un percorso euleriano ad un ciclo hamiltoniano

Consideriamo un ciclo euleriano $(v_1, \ldots, v_k, v_1)$.

```
procedure obtain_hamiltonian ()
   T = {v1}, i=2, v = v1
   while |T| < n {
      if vi ∉ T
      T = T ∪ {vi}
      collega v a vi
      v = vi
      i ++
   }
   collega v a v1
```

T è un ciclo hamiltoniano.

Esempio



Tour euleriano (5,4,5,3,2,3,1,3,5,6,5,8,9,8,10,8,7,11,7,8,5)

Tour hamiltoniano (5,4,5,3,2,3,1,3,5,6,5,8,9,8,10,8,7,11,7,8,5)

Proprietà

Double tree è un algoritmo 2-approssimato per il TSP.

Dimostrazione:

1. $z^* \geq z(\text{TREE})$(1-albero è un rilassamento per TSP).
2. Per costruzione, la lunghezza del “doppio albero” è $2*z(\text{TREE})$.
3. $z^H(\text{DT}) \leq 2*z(\text{TREE})$.

Pertanto:

$z^H(\text{DT}) / z^* \leq 2*z(\text{TREE})/z(\text{TREE}) = 2.$ □

Euristica di Christofides

1. Calcola un minimo albero ricoprente K.

2. Sia $V' \subseteq V$ l'insieme dei vertici che hanno grado dispari in K.

3. Trova il matching perfetto M di peso minimo sui nodi V'.

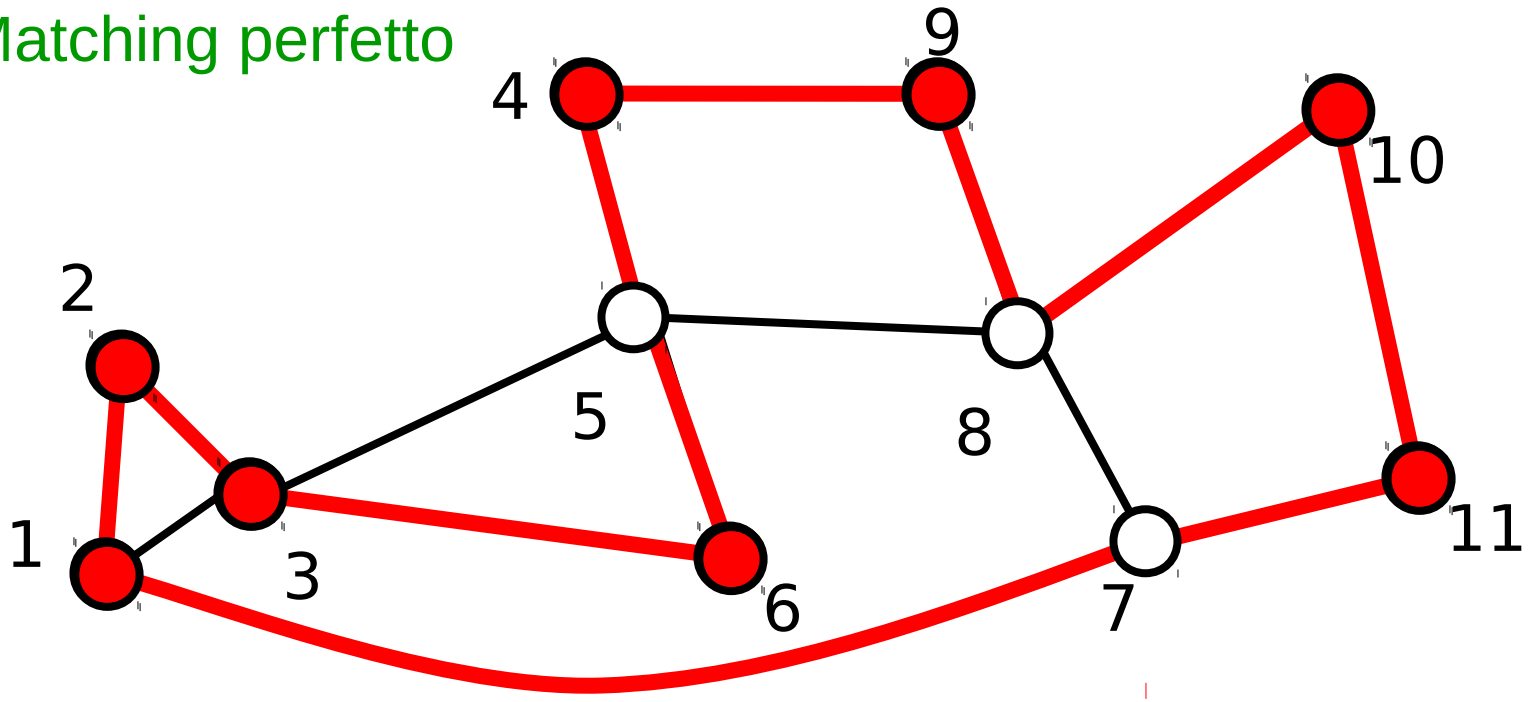
4. $M \cup K$ è un percorso euleriano.

5. Ricava un ciclo hamiltoniano dal percorso euleriano.

Indichiamo con $z^H(CH)$ il valore della soluzione che si ottiene applicando l'euristica di Christofides.

Esempio

Matching perfetto



Tour euleriano (1,2,3,6,5,4,9,8,10,11,7,8,5,3,1)

Nodi di grado dispari

Tour hamiltoniano (1,2,3,6,5,4,9,8,10,11,7,~~8,5,3~~,1)

Proprietà

L’algoritmo di Christofides per il TSP è 3/2-approssimato.

Dimostrazione:

1. $z^* \geq z(\text{TREE})$ (1-albero è un rilassamento per TSP).

1. Per costruzione, il peso di un ciclo euleriano è pari a $z(\text{TREE}) + z_M$. Grazie alla disuguaglianza triangolare e dato che il ciclo hamiltoniano è stato ottenuto cancellando alcuni nodi del ciclo euleriano, si ha

$z^H(\text{CH}) \leq z(\text{TREE}) + z_M$.

Proprietà

Dimostrazione: Sia C^* il ciclo hamiltoniano ottimo e sia Q l'insieme di vertici di grado dispari nel minimo albero ricoprente. Ricordiamo che $|Q| = q$ è sempre pari.

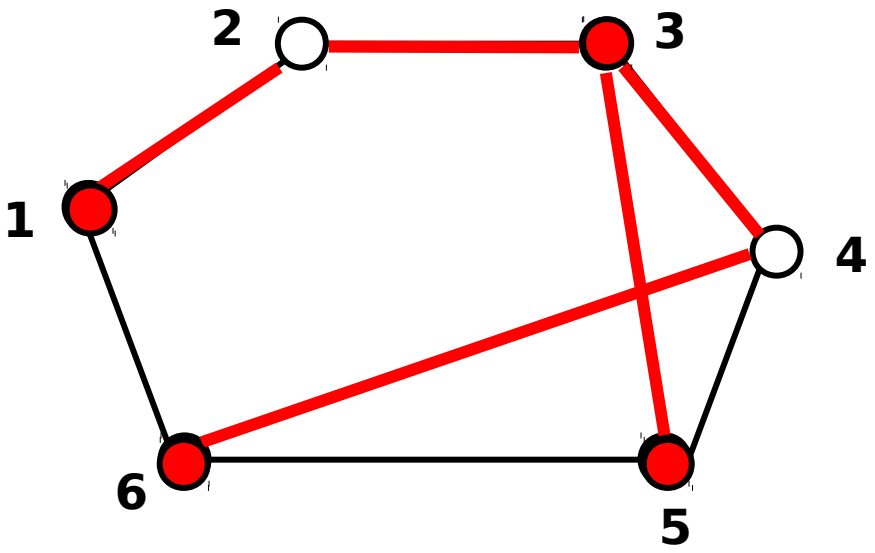
C^* tocca tutti i vertici del grafo e quindi anche i vertici in Q.

Esempio:

$C^* = \{(1,2),(2,3),(3,4),(3,5),(4,6)\}$

Min albero ricoprente in rosso

$Q = \{1, 3, 5, 6\}$



Dimostrazione

Cancelliamo da C^* tutti i vertici che non sono in Q e supponiamo invece di collegare i vertici in Q direttamente tra loro. In questo modo otteremo un ciclo C_Q di costo $z(C_Q)$.

Si ha che:

1. $z(C_Q) \leq z^*$ per la disuguaglianza triangolare. Difatti ognuno degli archi di C_Q ha un costo minore o uguale al corrispondente sottocammino in C^*.
2. C_Q è l'unione di 2 matching perfetti. Infatti, ricordando che $|Q| = q$ è un numero pari, possiamo dividere gli archi del ciclo C_Q in due insiemi alternati: gli archi pari (il secondo, il quarto,..., il q-esimo) e gli archi dispari (il primo, il terzo, ..., il $(q – 1)$-esimo). Ciascuno di questi insiemi costituisce un matching perfetto tra i nodi di Q, e quindi $z(C_Q) \geq 2z_M$.

Pertanto, $z^* \geq z(C_Q) \geq 2\, z_M$, ovvero $z_M \leq z^*/2$.

Quindi:

$$z^H(\text{CH}) \leq z(\text{TREE}) + z_M \leq z^* + z^*/2 \leq 3/2\, z^*$$

□

Bound primale e duale per il Knapsack 0-1

□

Problema di Knapsack

Avete a disposizione un budget *b* per gli investimenti dell'anno 2008.

Ad ogni progetto è associato

- un costo a_j (> 0)
- un guadagno atteso p_j (>0).

Problema: Scegliere l'insieme di progetti in modo che sia massimizzato il guadagno atteso senza eccedere il budget *b*.

Se ogni progetto può essere attivato non solo per intero ma anche in parte si parla di *knapsack continuo*.

Il knapsack continuo

Consideriamo il seguente problema (KRL):

$$\max \sum_{j=1}^{n} p_j x_j$$

$$\sum_{j=1}^{n} a_j x_j \leq b$$

$$0 \leq x_j \leq 1 \text{ per } j = 1, \ldots, n$$

(KRL) è un rilassamento del problema di knapsack 0-1.

Infatti, la collezione degli insiemi ammissibili del problema di knapsack 0-1 è contenuta nella regione ammissibile del problema di knapsack continuo.

La soluzione di (KRL) fornisce un upper bound per il problema di knapsack 0-1.

Il knapsack continuo

Come si risolve il problema di knapsack continuo?

Essendo formulato come problema di Programmazione Lineare,

si può risolvere utilizzando il metodo del simplesso.

In alternativa: Supponiamo di riordinare gli elementi del problema in modo che:

$\frac{p_1}{a_1} \geq \frac{p_2}{a_2} \geq \ldots \geq \frac{p_n}{a_n}$ e sia *h* l'indice minimo per cui $\sum_{j=1}^{h} a_j > b$

La soluzione $x^1=1, x^2=1, \ldots, x^{h-1}=1, x^h=f, x^{h+1}=0, \ldots, x^n=0$ con

$f = \frac{\left(b - \sum_{j=1}^{h-1} a_j\right)}{a_h}$ è ottima per (KLR).

Dimostrazione

Supponiamo, senza perdita di generalità, che gli elementi del problema soddisfino

$$\frac{p_1}{a_1} \geq \frac{p_2}{a_2} \geq \ldots \geq \frac{p_n}{a_n}$$

e sia $x^*_{LP} = (x^*_1, \ldots, x^*_n)$ la soluzione ottenuta con la formula precedente. Consideriamo una soluzione x' ottima, diversa da x^*_{LP}.

x' differisce da x^*_{LP} per almeno un elemento x'_i con $i \leq h$ e per almeno un elemento x'_k con $k > h$. Ciò significa che esiste un indice $i \leq h$ tale che $x'_i < 1$ e un indice $k > h$ tale che $x'_k > 0$.

Sia

$$d = \min\left\{a_k x'_k, a_i\left(1 - x'_i\right)\right\}$$

Per costruzione, $d > 0$.

Dimostrazione

Consideriamo un nuovo vettore x'' con le seguenti componenti:

$x''_j = x'_j$ per ogni $j \neq i, k$

$x''_i = x'_i + d\backslash a_i$

$x''_k = x'_k - d\backslash a_k$

x'' è ammissibile per KRL. Infatti:

$$\sum a_j x''_j = \sum a_j x'_j + \frac{a_i d}{a_i} - \frac{a_k d}{a_k} \leq b$$

Inoltre, $px'' > px'$. Infatti:

$$\sum p_j x''_j = \sum p_j x'_j + d\left(\frac{p_i}{a_i} - \frac{p_k}{a_k}\right) > \sum p_j x'_j$$

Ma, allora, la soluzione *x'* non è ottima (contraddizione).

□

Bound primale per il knapsack

Per determinare un bound primale per il problema di Knapsack è possibile utilizzare un algoritmo Greedy.

Tale algoritmo, applicato agli elementi della bisaccia riordinati secondo il criterio

$$\frac{p_1}{a_1} \geq \frac{p_2}{a_2} \geq \ldots \geq \frac{p_n}{a_n}$$

restituisce una soluzione ammissibile, e quindi un lower bound, per il problema di knapsack:

Bound primale per il knapsack

Algoritmo Greedy_knapsack ()

```
   d = 0; z = 0;
   for (j = 1; j < n; j ++) {
      if (d + a_j ≤ b) then
        x_j = 1;
        d = d + a_j;
          z = z + p_j;
      else x_j = 0; }
return z;
```

Riassumendo…

Abbiamo definito per il knapsack 0-1 un upper bound con le seguenti proprietà:

1. L'upper bound è "continuo", nel senso che si ottiene dalla soluzione di un problema di Programmazione Lineare e non da un problema di OC.
2. L'upper bound può essere calcolato con un algoritmo polinomiale più efficiente rispetto al metodo del simplesso.

Domanda: Può essere generalizzata questa tecnica di rilassamento?

\

Sostituendo la "stipula" $x \in \{0,1\}$ con il vincolo $0 \leq x \leq 1$ di una formulazione di un problema di PL-{0,1}, si ottiene sempre un rilassamento denominato *Rilassamento Lineare*.

Bound duali ottenuti tramite la teoria della dualità

Dualità

Definizione: Due problemi di ottimizzazione

$$z = \max \{f(x) : x \in X\}$$

$$w = \min \{g(y) : y \in Y\}$$

formano una coppia duale "debole" se $f(x) \leq g(y)$ per ogni $x \in X$ e $y \in Y$.

Se $z = w$ si dice che formano una coppia duale "forte".

<u>Vantaggio fondamentale rispetto al rilassamento:</u>

Per ottenere un bound attraverso il rilassamento, il problema rilassato va risolto all'ottimo. Invece, per una coppia duale **OGNI** soluzione ammissibile $y \in Y$ ($x \in X$) è un upper (lower) bound per z (w).

Formulazioni di PLI

Massimo insieme stabile

Dati: $G = (V,E)$, $|V| = n$, $|E| = m$.

Variabili decisionali:

$$x_i = \begin{cases} 1 & \text{se il vertice } i \text{ appartiene allo stabile } S \\ 0 & \text{altrimenti} \end{cases}$$

Formulazione:

$$\max \sum_{i \in V} x_i$$

$$x_i + x_j \leq 1 \qquad \forall\, (i,j) \in E$$

$$x_i \in \{0,1\} \qquad \forall\, i \in V$$

Rilassamento lineare

Rilassamento lineare del massimo insieme stabile

$$\max \sum_{i \in V} x_i$$

$$x_i + x_j \leq 1 \qquad \forall\, (i, j) \in E \qquad \text{STAB}_{\text{RL}}$$

$$x_i \geq 0 \qquad \forall\, i \in V$$

Osservazione: La limitazione $x_i \leq 1$ può essere omessa.

Indichiamo con $\alpha_{\text{RL}}(G)$ il valore della soluzione ottima di STAB_{RL}.

Formulazioni di PLI

Minimo edge cover

Dati: $G = (V,E)$, $|V| = n$, $|E| = m$.

Variabili decisionali:

$$y_{ij} = \begin{cases} 1 & \text{se l'arco } ij \text{ appartiene all'edge cover } C \\ 0 & \text{altrimenti} \end{cases}$$

Formulazione:

$$\min \sum_{ij \in E} y_{ij}$$

$$\sum_{j \in \partial(i)} y_{ij} \geq 1 \qquad \forall\, i \in V$$

$$y_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in E$$

Rilassamento lineare

Rilassamento lineare del minimo edge cover

$$\min \sum_{ij \in E} y_{ij}$$

$$\sum_{j \in \partial(i)} y_{ij} \geq 1 \qquad \forall\, i \in V$$

$$y_{ij} \geq 0 \qquad \forall\, (i, j) \in E$$

EDGE-C_{RL}

Osservazione: La limitazione $y_{ij} \leq 1$ può essere omessa.

Indichiamo con $\rho_{RL}(G)$ il valore della soluzione ottima di EDGE-C_{RL}.

Dualità

Osservazione: I problemi $STAB_{RL}$ e $EDGE\text{-}C_{RL}$ costituiscono una coppia duale forte per ogni grafo *G*.

Inoltre

$$\alpha(G) \leq \alpha_{RL}(G) \text{ e } \rho_{RL}(G) \leq \rho(G)$$

Pertanto

$$\alpha(G) \leq \alpha_{RL}(G) = \rho_{RL}(G) \leq \rho(G)$$

I problemi STAB e EDGE-C costituiscono una coppia duale debole per ogni grafo *G*.

Formulazioni di PLI

Massimo matching

Dati: $G = (V,E)$, $|V| = n$, $|E| = m$.

Variabili decisionali:

$$y_{ij} = \begin{cases} 1 & \text{se l'arco } ij \text{ appartiene al matching } M \\ 0 & \text{altrimenti} \end{cases}$$

Formulazione:

$$\max \sum_{ij \in E} y_{ij}$$

$$\sum_{j \in \partial(i)} y_{ij} \leq 1 \qquad \forall\, i \in V$$

$$y_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in E$$

Rilassamento lineare

Rilassamento lineare del massimo matching

$$\max \sum_{ij \in E} y_{ij}$$

$$\sum_{j \in \partial(i)} y_{ij} \leq 1 \qquad \forall\, i \in V$$

$$y_{ij} \geq 0 \qquad \forall\, (i, j) \in E$$

$\text{MATCHING}_{\text{RL}}$

Osservazione: La limitazione $y_{ij} \leq 1$ può essere omessa.

Indichiamo con $\mu_{\text{RL}}(G)$ il valore della soluzione ottima di $\text{MATCHING}_{\text{RL}}$.

Formulazioni di PLI

Minimo trasversale

Dati: $G = (V,E)$, $|V| = n$, $|E| = m$.

Variabili decisionali:

$$x_i = \begin{cases} 1 & \text{se il vertice } i \text{ appartiene al trasversale } T \\ 0 & \text{altrimenti} \end{cases}$$

Formulazione:

$$\min \sum_{i \in V} x_i$$

$$x_i + x_j \geq 1 \qquad \forall\, (i, j) \in E$$

$$x_i \in \{0,1\} \qquad \forall\, i \in V$$

Rilassamento lineare

Rilassamento lineare del minimo trasversale

$$\min \sum_{i \in V} x_i$$

$$x_i + x_j \geq 1 \qquad \forall\, (i, j) \in E \qquad \text{TRASV}_{\text{RL}}$$

$$x_i \geq 0 \qquad \forall\, i \in V$$

Osservazione: La limitazione $x_i \leq 1$ può essere omessa.

Indichiamo con $\tau_{\text{RL}}(G)$ il valore della soluzione ottima di TRASV_{RL}.

Dualità

Osservazione: I problemi MATCHING_{RL} e TRASV_{RL} costituiscono una coppia duale forte per ogni grafo *G*.

Inoltre

$$\mu(G) \leq \mu_{RL}(G) \text{ e } \tau_{RL}(G) \leq \tau(G)$$

Pertanto

$$\mu(G) \leq \mu_{RL}(G) = \tau_{RL}(G) \leq \tau(G)$$

I problemi MATCHING e TRASV costituiscono una coppia duale debole per ogni grafo *G*.

Dualità

Dal teorema di König segue che MATCHING e TRASV godono della dualità forte se *G* è un grafo bipartito.

Dal teorema di Gallai e dal teorema di König segue che STAB e EDGE-C godono della dualità forte se G è un grafo bipartito.