ALGORITMI DI APPROSSIMAZIONE E PROBLEMA TSP

[Deme, seconda edizione] cap. 16

Sezione 16.4 fino a 16.4.1 inclusa

[Cormen]

Paragrafi 35.1, 35.2.0 e 35.2.1

Algoritmi di Approssimazione

È sempre possibile **associare** ad un **problema** di **ottimizzazione** un **problema decisionale**, che ci dica se una determinata soluzione per un problema è ottima.

È facile capire che **il relativo problema decisionale** è **più facile** del **problema di ottimizzazione**.

Ma se già il problema decisionale è NP-completo, sembra davvero difficile trovare algoritmi efficienti per risolvere il problema di ottimizzazione.

A volte, **privilegiando l'efficienza a costo dell'ottimalità**, si utilizzano **algoritmi di approssimazione** tali per cui la soluzione restituita non è per forza ottima, ma ha costo C che si discosta dal costo C* di quella ottima di al massimo un **fattore di approssimazione** moltiplicativo $\boldsymbol{\rho(n)}$.

$$\max\left\{\frac{C}{C^*}, \frac{C^*}{C}\right\} \leq \rho(n)$$

Copertura (minima) di Vertici

Vediamo un esempio.

Dato un grafo non orientato G = (V, E), una **copertura di vertici** è un sottoinsieme **$\mathbf{V'} \subseteq \mathbf{V}$ tale che, se (u,v) $\in \mathbf{E}$, allora $\mathbf{u} \in \mathbf{V'} \vee \mathbf{v} \in \mathbf{V'}$**.

La **dimensione** di una **copertura** V’ è il **numero** di **vertici** che contiene.

Il problema della copertura di vertici consiste nel trovare **V’ ottima**, cioè tale che la sua **dimensione** sia **minima**.

Decidere se una certa copertura V’’ è ottima è un problema **NP-completo**, quindi difficilmente possiamo risolvere il nostro problema «efficientemente».

Possiamo però trovare facilmente una soluzione «**approssimativamente** ottima».

Copertura di Vertici – Algoritmo Approssimato

```
APPROX-COVER (G)
  C <- {}     //C conterrà la soluzione
  E’ <- E
  while E’ ≠ Ø
    sia (u,v) un arco in E’ //E’∈ A
    C <- C ∪ {u,v}
    rimuovi da E’ ogni arco incidente in u o v
  return C
```

La complessità è **O(n+m)** se G è rappresentato con liste di adiacenza.

Fattore di Approssimazione

APPROX-COVER ha un **fattore di approssimazione di 2**. Si dice anche che è **2-approssimato**

DIMOSTRAZIONE: Sia A l'insieme di archi (u,v) scelti all'inizio di ogni ciclo.

Per coprire gli archi in A, una **copertura ottima** deve contenere **almeno un vertice di ciascun arco in A**.

Due archi in A non possono avere un vertice in comune, perché una volta scelto (u,v) tutti gli archi incidenti ad u o v sono rimossi da E'.

Pertanto, non ci sono due archi in A coperti dallo stesso vertice e **|A| è un limite inferiore** per la dimensione di una **copertura ottima C*.**

$$|\boldsymbol{C}^*| \geq |\boldsymbol{A}|$$

Ma $|C| = 2|A|$, poiché nel risultato metto sia u che v, quindi

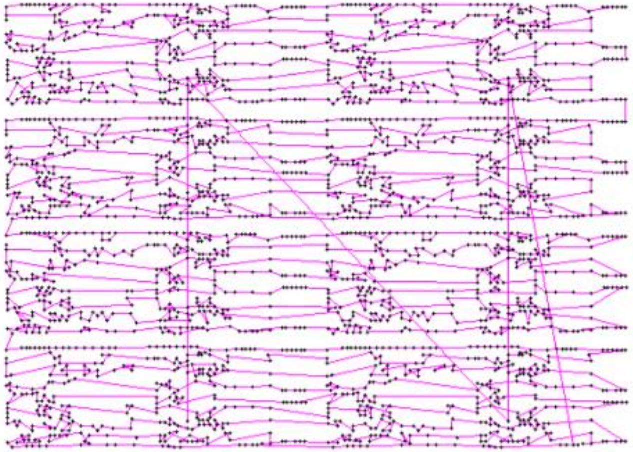
$$|\boldsymbol{C}| = 2|A| \leq \mathbf{2}|\boldsymbol{C}^*|$$

Problema del Commesso Viaggiatore (TSP)

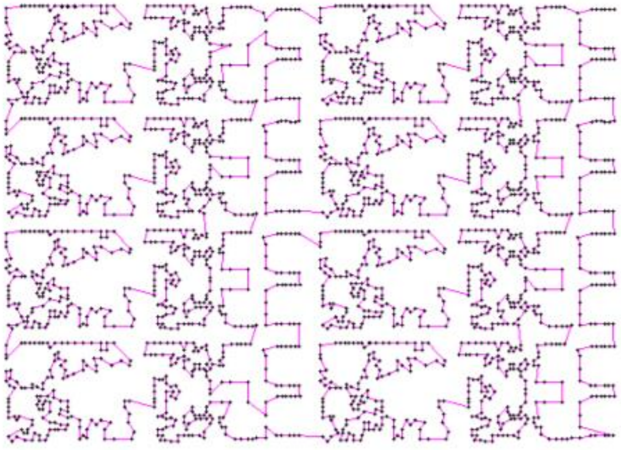
Il **Travel Salesman Problem** (da cui TSP) è uno dei problemi di **ottimizzazione** più famosi, con molte applicazioni pratiche.

Data una mappa (modellata con un grafo), il commesso viaggiatore deve passare per **tutte le città** in essa contenute **una sola volta**, e **tornare** al **punto di partenza** facendo il percorso **migliore**.

Il problema è uno dei più conosciuti e studiati in informatica. Ha moltissime applicazioni pratiche (es. macchina foratrice per i circuiti elettronici).



industry solution



optimal solution

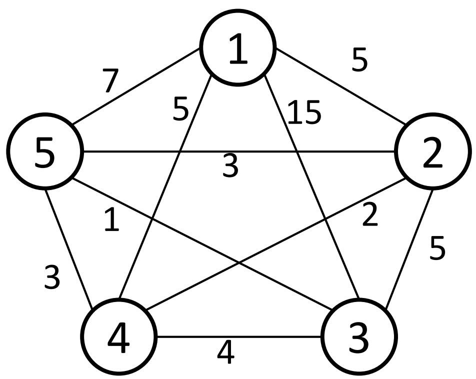
Problema del Commesso Viaggiatore (TSP)

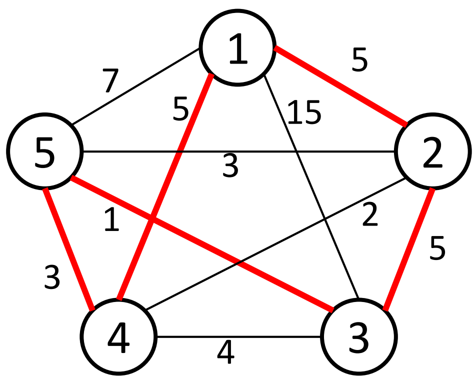
Più formalmente, dato un grafo **pesato**, **completo** e **non orientato** G=(V,E), bisogna trovare un **ciclo Hamiltoniano** (un ciclo semplice che contenga tutti i vertici) di **costo minimo**.

Noi consideriamo la versione «semplice» (ma realistica, ad esempio in una mappa), in cui i pesi degli archi rispettano la **disuguaglianza triangolare**

$$W(u,w) \leq W(u,v) + W(v,w)$$

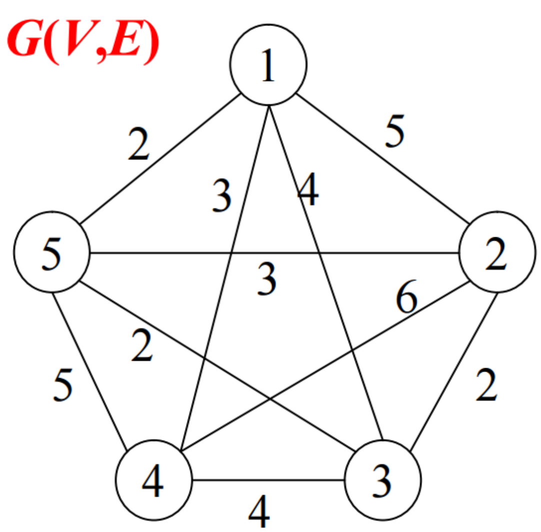
Il problema di capire se una soluzione è ottima è esso stesso NP-completo. Di conseguenza, cerchiamo soluzioni **approssimate**.

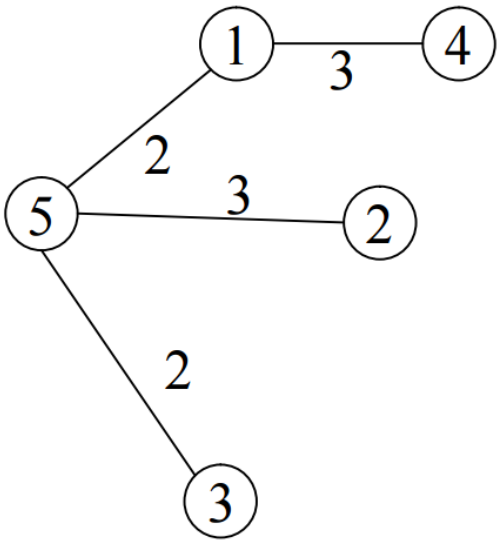




Algoritmo Approssimato per TSP

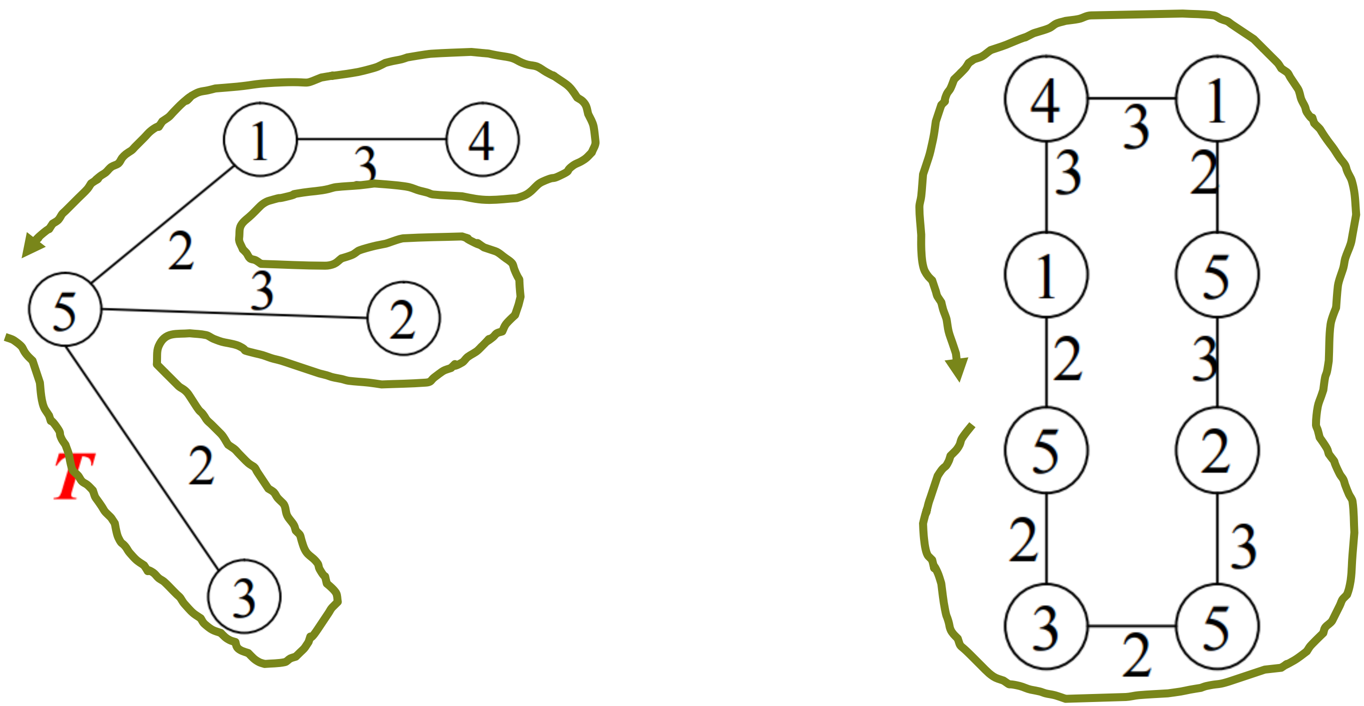
IDEA: un **minimo albero ricoprente (MAR)** è un insieme di archi di peso minimo che tocca tutti i vertici, ma **non è un ciclo**. Tuttavia, il **peso di un MAR** è un **limite inferiore** per il **peso di un circuito Hamiltoniano**.





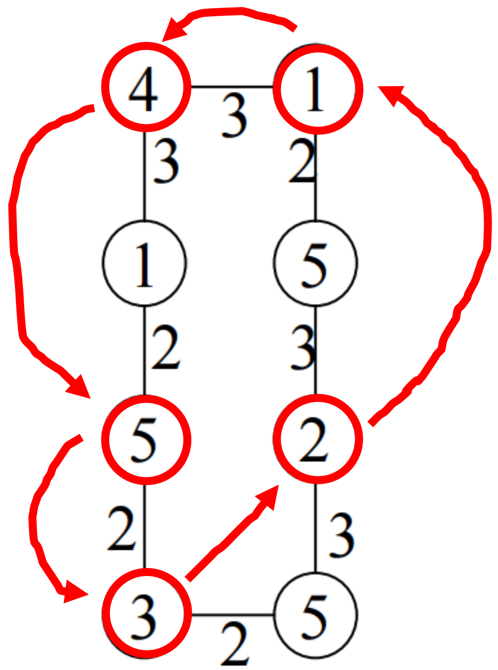
Algoritmo Approssimato per TSP - II

Circumnavigando il MAR otteniamo un ciclo (**con peso doppio rispetto al MAR**), che però non è Hamiltoniano perché passa sullo stesso nodo più volte.



Algoritmo Approssimato per TSP - III

Per trasformarlo in Hamiltoniano, possiamo «tagliare» le ripetizioni dei nodi, sfruttando la **disuguaglianza triangolare** che ci assicura che se sostituiamo (v,w), (w,z) (dove **w** è una **ripetizione** nel ciclo) con (v,z) otteniamo un ciclo di peso non maggiore.



Algoritmo Approssimato per TSP - IV

IDEA: la circumnavigazione + il taglio delle ripetizioni, altro non sono che una visita in profondità del MAR con traccia di inizio visita!

Possiamo usare una **visita in profondità dell'albero** (restituendo i vertici in **ordine di inizio visita**) per passare direttamente dall'albero al circuito.

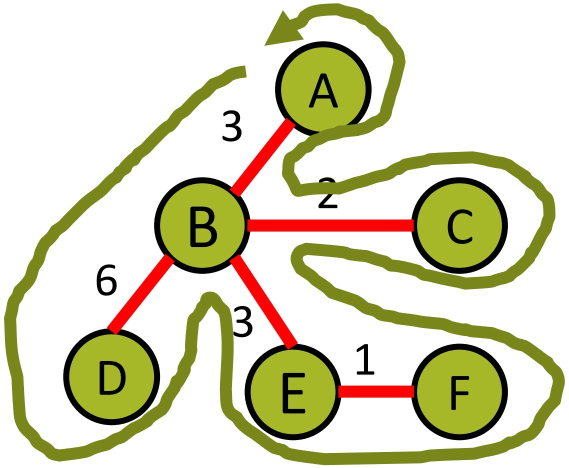
APPROX-TSP (G, W, r)

scegli un vertice r casualmente
A <- Prim (G, W, r)
ord <- DFS-ric-INIZIO (A, r)
return ord

sia ord = [V_1, V_2, … V_n], il ciclo Hamiltoniano è **V_1, V_2, … V_n, V_1**

DFS-ric-INIZIO restituisce un vettore contenente i vertici in ordine di inizio visita di una visita in profondità.

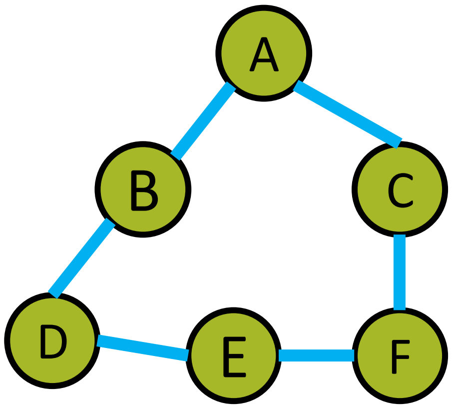
L'algoritmo è **2-approssimato** (la dimostrazione è sul Cormen, ma è facile da capire osservando i passaggi fatti).



Il rosso il MAR calcolato con Prim

In verde la visita DFS che restituisce
ord = [A,B,D,E,F,C]

A destra il ciclo Hamiltoniano
ottenuto. Siamo sicuri che abbia
costo minore o uguale al ciclo della visita
grazie alla disuguaglianza triangolare



Ricerca Locale

Per alcuni problemi, anche trovare algoritmi approssimati (con una buona approssimazione) è difficile.

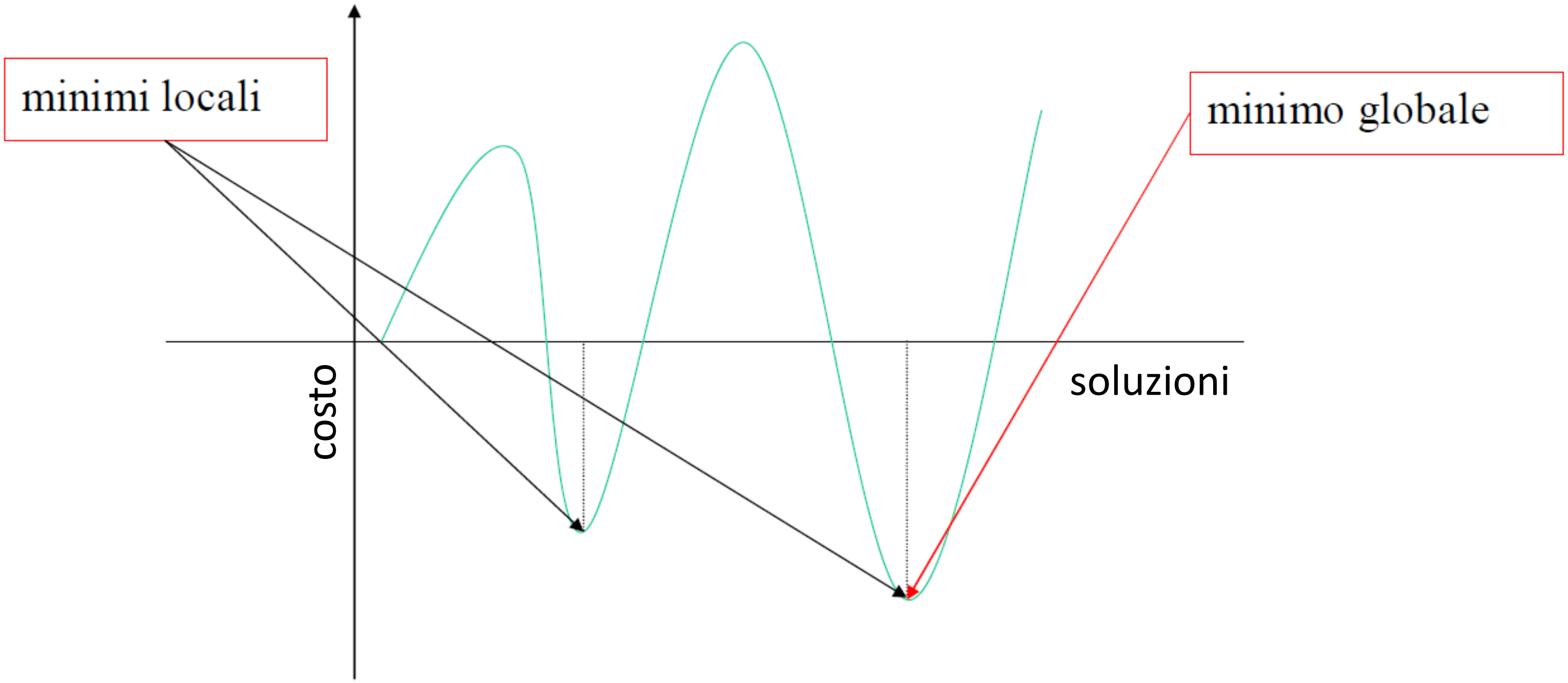
Una tecnica generale che trova delle soluzioni «buone» (ma non sappiamo quanto buone!) è quella della **ricerca locale**.

In un problema di ottimizzazione,

1. Si parte da una soluzione ammissibile x
2. Si calcolano tutte le soluzioni in un suo **vicinato N(x)** (cioè soluzioni ammissibili simili a x, in qualche modo)
3. Se esiste una soluzione y appartenente a N(x) migliore di x, allora x <- y e si riparte dal punto 2
4. Altrimenti, x è un **minimo (o massimo) locale** e ci possiamo fermare

Per utilizzare la ricerca locale dobbiamo essere in grado di **generare** almeno una **soluzione** e di definire un criterio per **generare** in maniera efficiente il **vicinato N(x)**.

Minimi Locali e Globali



Nella maggior parte dei casi **non possiamo fare assunzioni su** che minimo abbiamo raggiunto, né sulla **distanza tra il suo costo e quello della soluzione ottima**.

Se computazionalmente possibile, si applica la ricerca locale a **più soluzioni iniziali** calcolate casualmente, sperando di trovarne una sulla «cresta» del minimo globale, venendo attirati verso di esso (e trovandolo come soluzione).

Ricerca locale per TSP

Tecnica dei **k-scambi**:

1. Si genera un ciclo Hamiltoniano casuale
2. Se ne **cancellano k archi** non adiacenti e se ne **aggiungono altri k** in modo da ricreare il ciclo
3. Se così facendo si crea un ciclo di peso minore, si ricomincia l'iterazione dal punto 2
4. Altrimenti, si è trovata una **soluzione minima localmente**

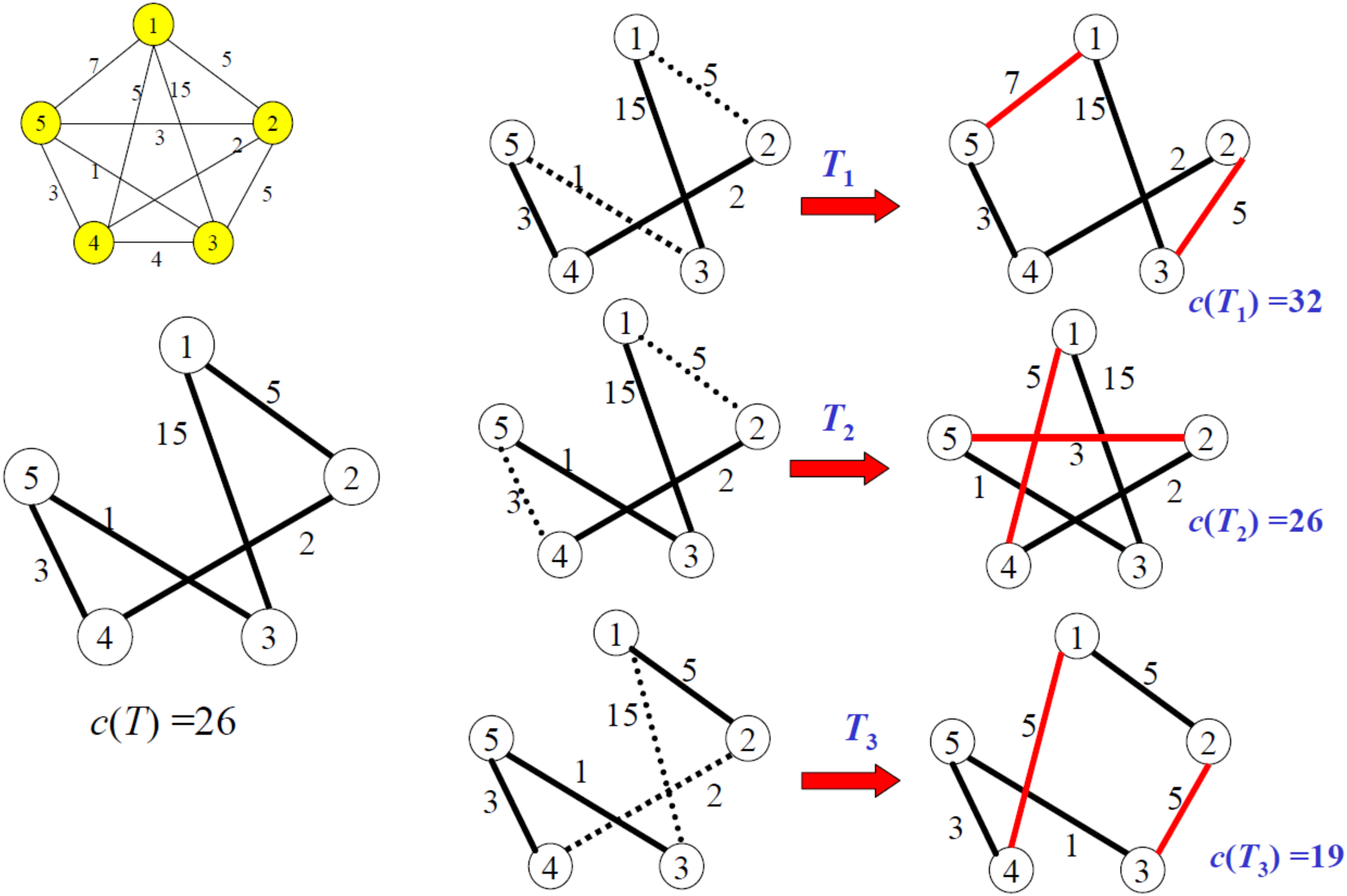
K-scambi con k = 2

1. Si genera un ciclo Hamiltoniano casuale
2. Per ogni coppia di archi (i,j) e (h,k), si cancellano (i,j) e (h,k) e si sostituiscono con (i,h) e (k,j)

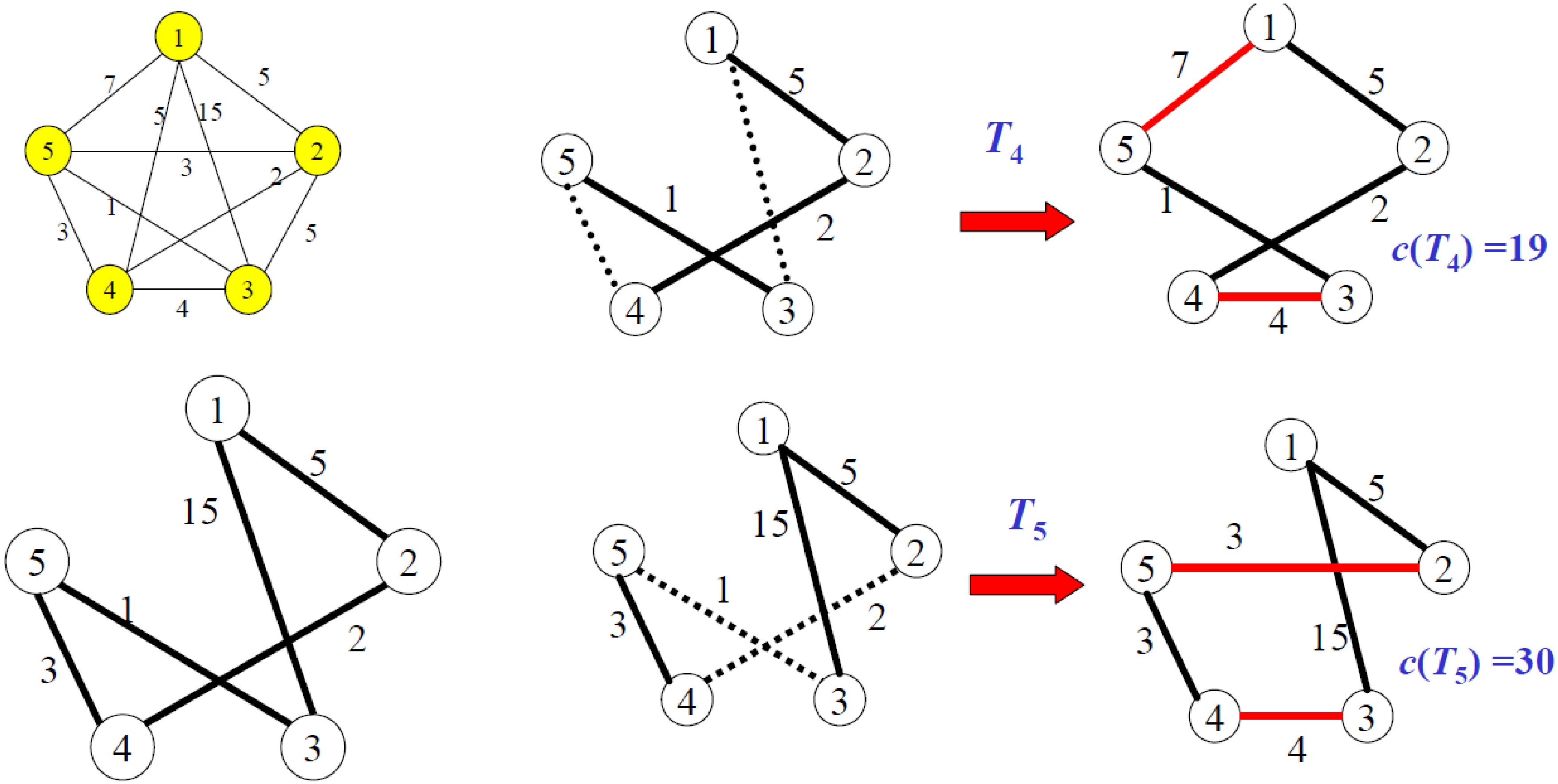
(dati n archi, il numero di sostituzioni fattibili sarà $O(n^2)$)

3. Se così facendo si crea un ciclo di peso minore, si ricomincia l’iterazione dal punto 2
4. Altrimenti, si è trovata una soluzione minima localmente

Esempio (da Bruni)
Generazione Vicinato



Esempio (da Bruni) Generazione Vicinato - II



$c(T) = 26$

Non ci sono altri modi di fare
2-scambio: ho generato tutto $N(T)$

- La miglior soluzione in $N(T)$ è T_4 (o anche T_3 , sono equivalenti)

Cosa devo aver capito fino ad ora

- Algoritmi di Approssimazione
- Problema della Copertura di Vertici e algoritmo per risolverlo 2-approssimato
- Problema del Commesso Viaggiatore
  - Algoritmo Greedy
  - Algoritmo approssimato
- Tecnica della Ricerca Locale
- Massimi e minimi locali e globali
- Algoritmo di Ricerca Locale per TSP basato sui k-scambi
  - Algoritmo dei k-scambi con k = 2

…se non ho capito qualcosa

- Alzo la mano e chiedo
- Ripasso sul libro
- Chiedo aiuto sul forum
- Chiedo o mando una mail al docente