# Algoritmi Euristici

## Corso di Laurea in Informatica e Corso di Laurea in Matematica

Roberto Cordone

DI - Università degli Studi di Milano


Lezioni: Lunedì 13.30 - 15.30 in Aula G30
Giovedì 13.30 - 15.30 in Aula G30
Ricevimento: su appuntamento
Tel.: 02 503 16235
E-mail: roberto.cordone@unimi.it
Web page: http://homes.di.unimi.it/~cordone/courses/2018-ae/2018-ae.html

# Efficacia di un algoritmo euristico

Un algoritmo euristico è utile se è

1. efficiente: "costa" molto meno di un algoritmo esatto
2. efficace: fornisce "spesso" una soluzione "vicina" a quella corretta

Ora parliamo dell'efficacia di un algoritmo euristico:

- vicinanza della soluzione ottenuta a quella ottima
- frequenza di ottenimento della soluzione ottima

che si possono ovviamente combinare in una

- distribuzione di frequenze di soluzioni più o meno vicine all'ottimo

L'efficacia di un algoritmo euristico può essere sottoposta ad

- analisi teorica (a priori): dimostrare che l'algoritmo fornisce soluzioni con una data garanzia di qualità, sempre o con una data frequenza
- analisi sperimentale (a posteriori): misurare le prestazioni dell'algoritmo su un campione di istanze di benchmark

# Efficacia di un algoritmo euristico

L'efficacia di un algoritmo euristico di ottimizzazione $A$ si misura con la differenza tra i valori della soluzione euristica $f_A(I)$ e ottima $f^*(I)$

- differenza assoluta:

$$\tilde{\delta}_A(I) = |f_A(I) - f^*(I)| \geq 0$$

  Si usa di rado, perché dipende dall'unità di misura dell'obiettivo

- differenza relativa:

$$\delta_A(I) = \frac{|f_A(I) - f^*(I)|}{f^*(I)} \geq 0$$

  È frequente nell'analisi sperimentale (*spesso espressa in percentuale*)

- rapporto di approssimazione:

$$\rho_A(I) = \max\left[\frac{f_A(I)}{f^*(I)}, \frac{f^*(I)}{f_A(I)}\right] \geq 1$$

  È frequente nell'analisi teorica: si usa la prima forma per i problemi di minimo, la seconda forma per i problemi di massimo

Come per la complessità, puntiamo a una misura sintetica, non legata a $I$

# Analisi teorica (nel caso pessimo)

Come per la complessità, cominciamo valutando il caso pessimo

La differenza tra soluzione euristica e ottima è in generale illimitata, ma per alcuni algoritmi è limitata:

- approssimazione assoluta:

$$\exists \tilde{\alpha}_A \in \mathbb{N} : \tilde{\delta}_A(I) \leq \tilde{\alpha}_A \text{ per ogni } I \in \mathcal{I}$$

  Un esempio (raro) è l'algoritmo di Vizing per l'*Edge Coloring* ($\tilde{\alpha} = 1$)

- approssimazione relativa:

$$\exists \alpha_A \in \mathbb{R}^+ : \rho_A(I) \leq \alpha_A \text{ per ogni } I \in \mathcal{I}$$

Il fattore $\alpha_A$ ($\tilde{\alpha}_A$) si dice garanzia di approssimazione relativa (assoluta)

In generale, la garanzia dipende dalla dimensione dell'istanza

$$\rho_A(I) \leq \alpha_A(n) \text{ per ogni } I \in \mathcal{I}_n, n \in \mathbb{N}$$

ma, contrariamente all'efficacia, per l'efficienza può anche non dipenderne

# Come ricavare una garanzia di approssimazione?

Per un problema di minimizzazione, si vuole dimostrare che

$$f_A(I) \leq \alpha f^*(I) \text{ per ogni } I \in \mathcal{I}$$

1. si trova un modo per costruire una stima per difetto $LB(I)$

$$LB(I) \leq f^*(I) \qquad I \in \mathcal{I}$$

2. si trova un modo per costruire una stima per eccesso $UB(I)$, che sia legata a $LB(I)$ da un coefficiente $\alpha$ (oppure $\alpha(n)$)

$$UB(I) = \alpha\, LB(I) \qquad I \in \mathcal{I}$$

3. si trova un algoritmo $A$ la cui soluzione non è peggiore di $UB(I)$

$$f_A(I) \leq UB(I) \qquad I \in \mathcal{I}$$

Quindi $f_A(I) \leq UB(I) = \alpha LB(I) \leq \alpha f^*(I)$, per ogni $I \in \mathcal{I}$

$$f_A(I) \leq \alpha f^*(I) \text{ per ogni } I \in \mathcal{I}$$

*Ovviamente $\alpha$ dipende in qualche modo dall'algoritmo $A$*

# Un algoritmo 2-approssimato per il *VCP*

Dato un grafo non orientato $G = (V, E)$ si cerca il sottoinsieme di vertici di cardinalità minima tale che ogni lato del grafo vi incida

Si dice matching un insieme di lati non adiacenti fra loro

Matching massimale è un matching tale che qualsiasi altro lato del grafo è adiacente a un lato del matching

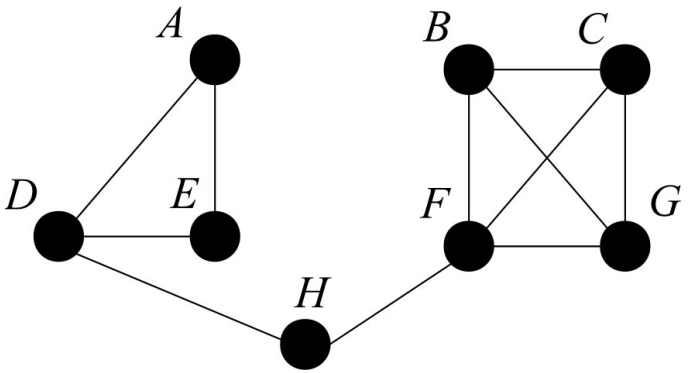
Algoritmo del matching:

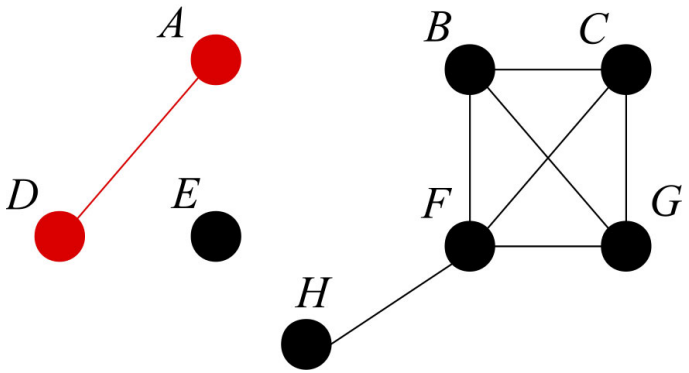
1. Si costruisce un matching massimale $M \subseteq E$,
   cioè tale che ogni altro lato di $E$ è adiacente a un lato di $M$
   *Basta scorrere i lati scartando quelli adiacenti ai lati già presi*
2. L'insieme dei vertici estremi dei lati del matching è una soluzione

$$x_A := \bigcup_{(u,v) \in M} \{u, v\}$$

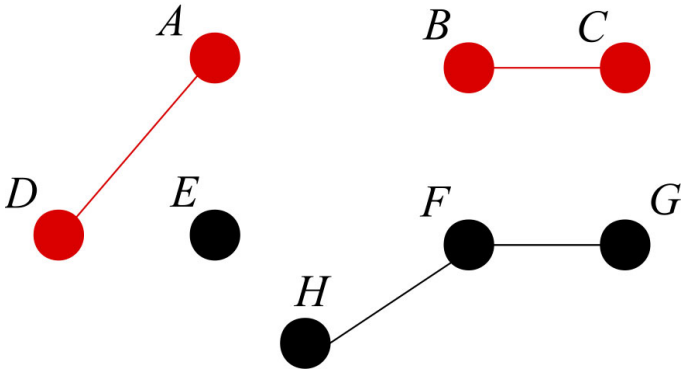
   e si può migliorare eliminando i vertici ridondanti

# Esempio

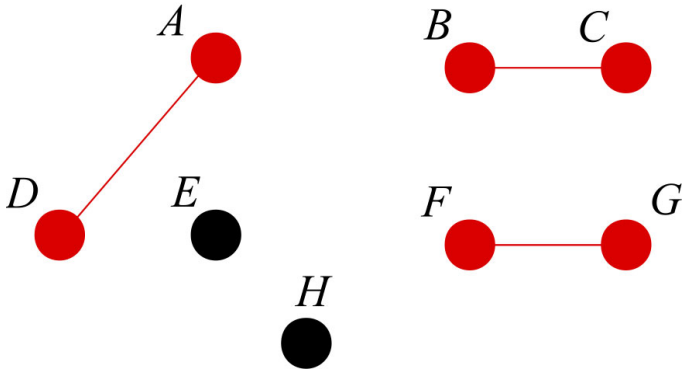



$|M| = 1$




$|M| = 2$




$f_A = 2 \cdot |M| = 6$ (e $G$ è ridondante)

La soluzione ottima è $f^* = 5$

# Dimostrazione

L'algoritmo del matching è 2-approssimato

1. La cardinalità del matching $M$ è una stima per difetto $LB(I)$
   - la cardinalità di una copertura ottima per qualsiasi sottoinsieme di lati $E' \subseteq E$ non supera quella di una copertura ottima per $E$

     $$|x^*_{E'}| \leq |x^*_E|$$

     (*coprire tutti i lati costa di più che coprire solo i lati del matching*)
   - la copertura ottima di un matching $M$ ha cardinalità $|M|$
     (*per ogni lato del matching basta e occorre un vertice diverso*)
2. Includendo entrambi i vertici di ogni lato del matching si ottiene
   - una stima per eccesso (*copre sia il matching sia i lati adiacenti*)
   - di valore $UB(I) = 2LB(I)$ (*due vertici diversi per ogni lato*)
3. L'algoritmo del matching dà soluzioni di valore $f_A(I) \leq UB(I)$
   (*scremando i vertici ridondanti, se ve ne sono*)

Ne deriva che $f_A(I) \leq 2f^*(I)$ per ogni $I \in \mathcal{I}$, cioè $\alpha_A = 2$

# . . . and the bound is tight!

Per costruzione, il fattore $\alpha_A$ non lega $f_A(I)$ e $f^*(I)$, ma $UB(I)$ e $LB(I)$

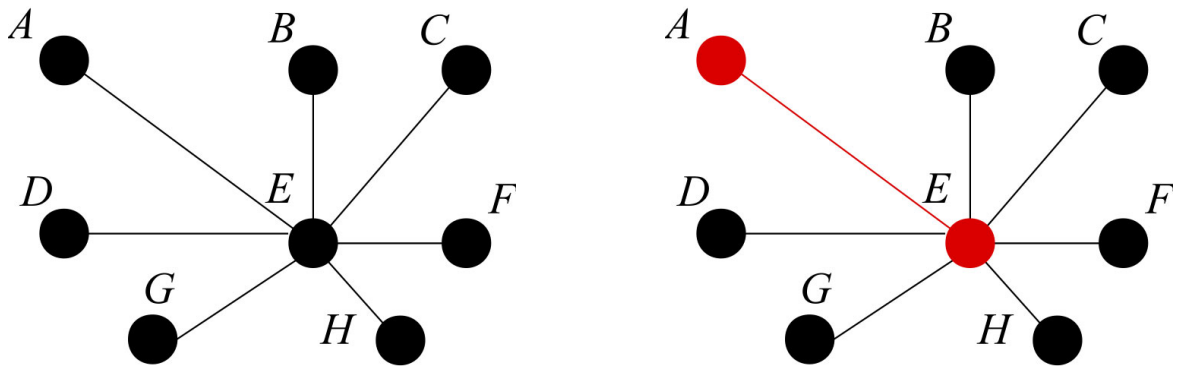
Il rapporto di approssimazione $\rho_A(I)$ spesso è molto migliore di $\alpha_A$

Esistono istanze $\bar{I}$ per cui $f_A(\bar{I}) = \alpha_A f^*(\bar{I})$? Che caratteristiche hanno?

Lo studio di queste istanze serve a

- valutare se sono rare o frequenti
- introdurre modifiche *ad hoc* per migliorare l'algoritmo

La tipica espressione *and the bound is tight* introduce in letteratura la presentazione di istanze per cui vale il caso pessimo




*Se si rimedia a tutti i casi pessimi, l'approssimazione migliora*

# Schemi di approssimazione

Per i problemi $\mathcal{NP}$-difficili

- un algoritmo $A$ esatto offre l'approssimazione ideale ($\tilde{\alpha}_A = 0$, $\alpha_A = 1$) ma con complessità $T_A$ esponenziale
- un algoritmo approssimato offre un'approssimazione peggiore ($\tilde{\alpha}_A > 0$, $\alpha_A > 1$), ma può avere complessità $T_A$ polinomiale

Talvolta è possibile trovare diversi compromessi fra efficienza ed efficacia

- fattori di approssimazione via via migliori: $\alpha_{A_1} < \ldots < \alpha_{A_r}$
- complessità computazionali via via crescenti: $T_{A_1} > \ldots > T_{A_r}$

Schema di approssimazione è un algoritmo $A$ parametrico con $\alpha$ a piacere

$$T_{A_\alpha} \in O\left(f\left(n, \alpha\right)\right) \text{ per ogni } \alpha \in [1; +\infty)$$

Uno schema di approssimazione può essere

- polinomiale se $f\left(n, \alpha\right)$ è un polinomio in $n$ per ogni $\alpha$ fissato
- pienamente polinomiale se $f\left(n, \alpha\right)$ è un polinomio in $n$ e in $1/\alpha$

# Inapprossimabilità

Esistono problemi non approssimabili, per i quali
l'unico algoritmo approssimato è un algoritmo esatto

Ad es., il $TSP$ ha istanze $\bar{I}$ non approssimabili:

- $G$ completo
- $c_{ij} = 0$ per $(i,j) \in A_0 \subset A$
- $c_{ij} = 1$ per $(i,j) \in A \setminus A_0$

Quanto vale l'ottimo?

$$\begin{cases} f^*\left(\bar{I}\right) = 0 \text{ se } A_0 \text{ contiene un ciclo orientato hamiltoniano} \\ f^*\left(\bar{I}\right) \leq 1 \text{ altrimenti} \end{cases}$$

Infatti, nel secondo caso la soluzione ottima contiene almeno un arco $\notin A$
Supponiamo che un algoritmo $A$ polinomiale offra una garanzia $\alpha$

$$f_A(I) \leq \alpha f^*(I) \ \forall I \in \mathcal{I}$$

Ma allora $f^*\left(\bar{I}\right) = 0 \Leftrightarrow f_A\left(\bar{I}\right) = 0$ ($A$ trova l'ottimo!)

Se c'è un ciclo hamiltoniano nel sottografo $G(N, A_0)$, $A$ lo trova,
risolvendo un problema $\mathcal{NP}$-completo: dunque $\mathcal{P} = \mathcal{NP}$!

# Il *TSP* con disuguaglianza triangolare

Un problema non approssimabile può contenere problemi approssimabili

Consideriamo il *TSP* con le ipotesi aggiuntive (frequenti in pratica) che

- che il grafo $G = (N, A)$ sia completo
- la funzione $c$ sia simmetrica e goda della disuguaglianza triangolare

$$c_{ij} = c_{ji} \quad \forall i, j \in N \qquad \text{e} \qquad c_{ij} + c_{jk} \geq c_{ik} \quad \forall i, j, k \in N$$

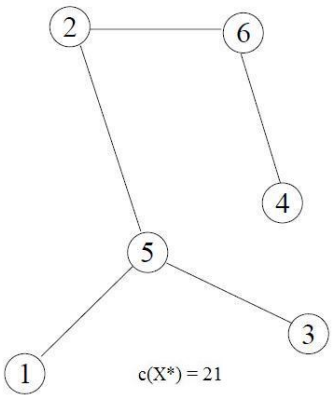
Algoritmo del doppio albero

1. Consideriamo il grafo completo non orientato corrispondente a $G$
2. Costruiamo un albero ricoprente di costo minimo $T^* = (N, X^*)$
3. L'insieme $x$ degli archi di $G$ corrispondenti ai lati di $X^*$ forma un ciclo orientato che passa per ogni nodo, in generale più volte
4. Finché ci sono nodi con più di un arco uscente ed entrante
   - si sceglie un nodo qualsiasi $j$ e una coppia di archi $(i, j)$ e $(j, k)$ in $x'$
   - si sostituisce la coppia di archi con l'arco diretto

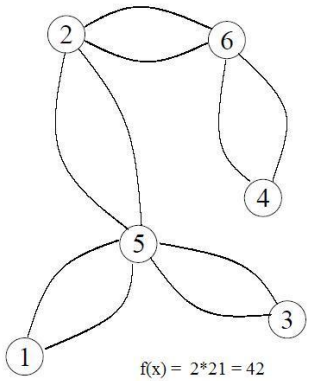
$$x := x \setminus \{(i, j), (j, k)\} \cup \{(i, k)\}$$

# Esempio

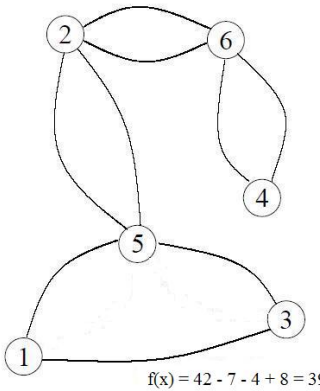
| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | - | 9 | 8 | 9 | 7 | 9 |
| 2 | 9 | - | 6 | 5 | 4 | 3 |
| 3 | 8 | 6 | - | 9 | 4 | 6 |
| 4 | 9 | 5 | 9 | - | 8 | 3 |
| 5 | 7 | 4 | 4 | 8 | - | 5 |
| 6 | 9 | 3 | 6 | 3 | 5 | - |



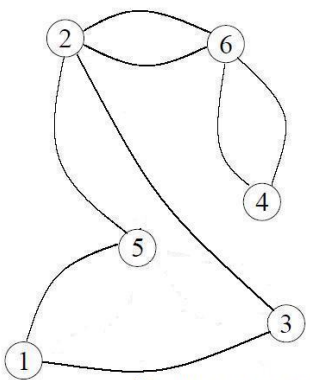
c(X*) = 21



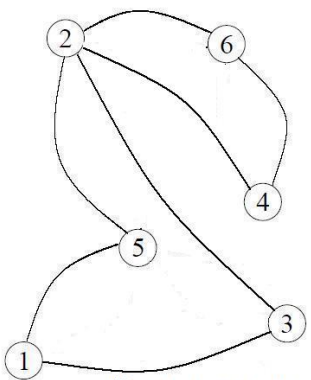
f(x) = 2*21 = 42



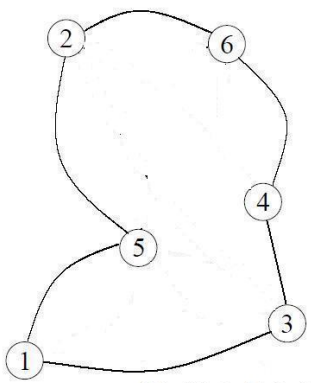
f(x) = 42 - 7 - 4 + 8 = 39



f(x) = 39 - 4 - 4 + 6 = 37



f(x) = 37 - 3 - 3 + 5 = 36



f(x) = 36 - 6 - 5 + 9 = 34

# Dimostrazione

L'algoritmo del doppio albero è 2-approssimato

1. il costo dell'albero ricoprente minimo è una stima per difetto $LB(I)$
   - togliendo un arco a un ciclo hamiltoniano si ottiene un cammino hamiltoniano meno costoso
   - un cammino hamiltoniano è un particolare albero ricoprente
2. sostituendo ogni lato dell'albero con due archi opposti si ottiene
   - una stima per eccesso (*è un cammino hamiltoniano*)
   - di valore $UB(I) = 2LB(I)$ (*due archi sostituiscono ogni lato*)
3. l'algoritmo del doppio albero dà soluzioni di valore $f_A(I) \leq UB(I)$
   (*sostituendo due archi consecutivi con uno diretto il costo cala*)

Ne deriva che $f_A(I) \leq 2f^*(I)$ per ogni $I \in \mathcal{I}$, cioè $\alpha_A = 2$

# Oltre il caso pessimo

Il criterio del caso pessimo è brutale, come nella misura della complessità: vi sono algoritmi con prestazioni spesso buone, ma talvolta cattive

Le vie alternative sono simili a quelle usate per studiare la complessità

- parametrizzazione: introdurre fattori di approssimazione che dipendono da parametri caratteristici dell'istanza
- studio del caso medio: introdurre una distribuzione di probabilità sulle istanze e valutare il valore atteso del fattore di approssimazione
  (*l'algoritmo potrebbe avere prestazioni cattive solo su istanze rare*)

ma ce n'è un'altra

- randomizzazione: l'algoritmo esegue operazioni che non dipendono solo dai dati, ma anche da numeri pseudocasuali
  - il risultato dell'algoritmo diventa una variabile casuale
  - la complessità potrebbe diventarlo, ma di solito non cambia
    (*però si può rieseguire l'algoritmo indefinitamente, alimentandolo con sequenze pseudocasuali diverse*)

# Algoritmi approssimati randomizzati

Per un algoritmo $A$ randomizzato, $f_A(I)$ e $\rho_A(i)$ sono variabili aleatorie

Un algoritmo randomizzato approssimato ha rapporto di approssimazione il cui valore atteso è limitato da una costante

$$E[\rho_A(I)] \leq \alpha_A \text{ per ogni } I \in \mathcal{I}$$

Prendiamo il *Max-SAT*: data una CNF, si trovi un assegnamento di verità alle variabili logiche che soddisfi un insieme di formule di peso massimo

Algoritmo puramente casuale:

Ad ogni variabile $x_j$ $(j = 1, \ldots, n)$ si assegna

- valore *Falso* con probabilità $1/2$
- valore *Vero* con probabilità $1/2$

*Qual è il valore atteso della soluzione?*

# Approssimazione randomizzata per il *Max-SAT*

Sia $\mathcal{C}_x \subseteq \{1, \ldots, m\}$ l'insieme delle formule soddisfatte dalla soluzione $x$

Il valore dell'obiettivo $f(x)$ è il peso totale delle formule in $\mathcal{C}_x$
Il valore atteso rispetto a tutte le soluzioni $x$ proposte dall'algoritmo $A$ è

$$E[f_A(I)] = E\left[\sum_{i \in \mathcal{C}_x} w_i\right] = \sum_{i \in \mathcal{C}} (w_i \cdot Pr[i \in \mathcal{C}_x])$$

Sia $k_i$ il numero di letterali della formula $i \in \mathcal{C}$ e $k_{\min} = \min_{i \in \mathcal{C}} k_i$

$$Pr[i \in \mathcal{C}_x] = 1 - \left(\frac{1}{2}\right)^{k_i} \geq 1 - \left(\frac{1}{2}\right)^{k_{\min}} \quad \text{per ogni } i \in \mathcal{C}$$

$$\Rightarrow E[f_A(I)] \geq \sum_{i \in \mathcal{C}} w_i \cdot \left[1 - \left(\frac{1}{2}\right)^{k_{\min}}\right] = \left[1 - \left(\frac{1}{2}\right)^{k_{\min}}\right] \sum_{i \in \mathcal{C}} w_i$$

e siccome $f^*(I) \leq \sum_{i \in \mathcal{C}} w_i$ per ogni $I \in \mathcal{I}$ ed $E[\rho_A(I)] = f^*(I) / E[f_A(I)]$

si ottiene

$$E[\rho_A(I)] \leq 1 / \left[1 - \left(\frac{1}{2}\right)^{k_{\min}}\right] \leq 2$$