Complessità ed Approssimazione

Paola Festa

Dip. di Matematica e Applicazioni “R. Caccioppoli”

Università degli Studi di Napoli FEDERICO II

Studio 137 – Tel. 081 675605

http://www.dma.unina.it/∼festa/

E-mail: paola.festa@unina.it

Sommario

- Introduzione agli Algoritmi di Approssimazione
  - algoritmi greedy e algoritmi sequenziali;
  - algoritmi di ricerca locale;
  - algoritmi basati sulla programmazione lineare;
  - algoritmi di programmazione dinamica;
  - algoritmi random.
- Classi di Approssimazione
  - $\mathbf{APX}$: approssimazione con rapporto di prestazione garantito;
  - $\mathbf{PTAS}$: schemi di approssimazione polinomiale;
  - $\mathbf{FPTAS}$: schemi di approssimazione completamente polinomiale e risultati negativi per la classe $\mathbf{FPTAS}$;
  - pseudo-polinomialità e problemi *fortemente* $\mathbf{NP}$-completi.
- Approssimazione attraverso Randomizzazione

Introduzione agli Algoritmi di Approssimazione

Problemi di Ottimizzazione, ①

Molti problemi di ottimizzazione sono *intrattabili* dal punto di vista computazionale ($\mathbf{NP}$-hard).

Un generico problema di ottimizzazione $\mathcal{P}$ può essere descritto dalla quadrupla $(\Upsilon_{\mathcal{P}}, SOL_{\mathcal{P}}, z_{I \in \Upsilon_{\mathcal{P}}}, obj_{\mathcal{P}})$, dove

- ✗ $\Upsilon_{\mathcal{P}}$ è l'insieme delle istanze di $\mathcal{P}$;
- ✗ $SOL_{\mathcal{P}}$ è una funzione che associa ad ogni istanza $I \in \Upsilon_{\mathcal{P}}$ l'insieme (eventualmente vuoto) delle *soluzioni ammissibili* per I;
- ✗ $z_I(\cdot)$ è la *funzione obiettivo* di $\mathcal{P}$: data un'istanza $I \in \Upsilon_{\mathcal{P}}$ e dato l'insieme $SOL_{\mathcal{P}}(I) = \{x \mid x \text{ soluz. ammissibile per l'istanza } I \text{ di } \mathcal{P}\}$, $z_I(x)$ è un valore numerico associato alla soluzione x;
- ✗ $obj_{\mathcal{P}} \in \{\min, \max\}$ indica se $\mathcal{P}$ è un problema in cui si vuole minimizzare la f.o. $z_I(\cdot)$ oppure massimizzarla.

Esempio ①

Problema: Vertex Cover Minimo

ISTANZA: Grafo non orientato $G = (V, E)$;

SOLUZIONE: $V' \subseteq V$ t.c. $\forall\, [v_i, v_j] \in E$, $v_i \in V'$ o $v_j \in V'$;

F.O. z: $|V'|$;

obj: min.

Esempio ②

Problema: Colorazione Minima di un Grafo

ISTANZA: Grafo non orientato $G = (V, E)$;

SOLUZIONE: Assegnamento $f : V \mapsto \mathcal{K} = \{1, \ldots, k\}$ di k colori ai nodi t.c. $\forall\, [u, v] \in E$, $f(u) \neq f(v)$;

F.O. z: $|\mathcal{K}|$;

obj: min.

Problemi di Ottimizzazione, ③

È interessante osservare che appartengono alla classe dei problemi computazionalmente intrattabili molti problemi “simili” a problemi di ottimizzazione trattabili.

Trattabile	**Intrattabile**
Massimo Flusso (MF)	Massimo Flusso Multi-commodity (MFM)
Minimum Spanning Tree (MST)	Commesso Viaggiatore (TSP)
Bipartite Matching (BM)	3D-Matching (3DM)
Zaino Frazionario	Zaino 0/1

Un problema è detto intrattabile da un punto di vista computazionale, in quanto è molto improbabile che esista un algoritmo efficiente (polinomiale) in grado di individuarne una soluzione ottima.

Algoritmi di Approssimazione, ①

Nell'affrontare un problema intrattabile, è conveniente progettare un algoritmo efficiente (polinomiale) in grado di individuarne una soluzione ammissibile subottima.

In molti casi è banale individuare un algoritmo polinomiale in grado di calcolare una soluzione ammissibile (spesso banale).

Esempi:

- ✗ `Vertex Cover Minimo`:
  un algoritmo che restituisce in output $V' = V$;
- ✗ `Colorazione Minima di un Grafo`:
  un algoritmo che assegna un colore differente ad ogni nodo e restituisce in output $\mathcal{K} = \{1, \ldots, |V|\}$.

Algoritmi di Approssimazione, ②

Chiaramente, non è altrettanto semplice individuare un algoritmo polinomiale in grado di calcolare una soluzione ammissibile subottima che soddisfi un qualche criterio di qualità, ossia che possa essere definita una soluzione approssimata e l'algoritmo che l'ha individuata algoritmo di approssimazione.

Dati un problema $\mathcal{P}$, una sua istanza $I \in \Upsilon_{\mathcal{P}}$ ed indicate con x_I^* e x_I rispettivamente una soluzione ottima ed una soluzione subottima per I, la qualità di x_I può essere definita in termini di distanza fra $z_I(x_I^*)$ e $z_I(x_I)$.

DEFINIZIONE: Sia $\mathcal{A}$ un algoritmo di approssimazione per un problema $\mathcal{P}$ e sia x_I una soluzione subottima individuata da $\mathcal{A}$ in corrispondenza di un'istanza $I \in \Upsilon_{\mathcal{P}}$.
Si definisce *errore assoluto di $\mathcal{A}$ nel caso peggiore* (o *rapporto di prestazione di $\mathcal{A}$ nel caso peggiore*) la più piccola quantità $r(\mathcal{A})$ tale che

$$R_{\mathcal{A}}(I, x_I) = \max\left(\frac{z_I(x_I^*)}{z_I(x_I)}, \frac{z_I(x_I)}{z_I(x_I^*)}\right) \leq r(\mathcal{A}), \qquad \text{per ogni } I \in \Upsilon_{\mathcal{P}}.$$

Si definisce *errore relativo di $\mathcal{A}$ nel caso peggiore* la più piccola quantità $\epsilon(\mathcal{A})$ tale che

$$E_{\mathcal{A}}(I, x_I) = \frac{|z_I(x_I^*) - z_I(x_I)|}{\max\{z_I(x_I^*), z_I(x_I)\}} \leq \epsilon(\mathcal{A}), \qquad \text{per ogni } I \in \Upsilon_{\mathcal{P}}.$$

<u>OSSERVAZIONI</u>:

- ✓ $E_{\mathcal{A}}(I, x_I) = 1 - \frac{1}{R_{\mathcal{A}}(I, x_I)}$;
- ✓ $\epsilon(\mathcal{A}) = 0 \Longrightarrow$ la soluzione individuata è ottima;
  $\epsilon(\mathcal{A}) \to 1 \Longrightarrow$ la soluzione individuata non è di buona qualità;
- ✓ $r(\mathcal{A}) = 1 \Longrightarrow$ la soluzione individuata è ottima;
  $r(\mathcal{A}) \gg 1 \Longrightarrow$ la soluzione individuata non è di buona qualità.

Algoritmi di Approssimazione, ④

Alcune tecniche comunemente usate per individuare in tempo polinomiale soluzioni ottime per problemi trattabili, risultano spesso essere tecniche di approssimazione per problemi intrattabili:

★ algoritmi greedy;

★ algoritmi sequenziali;

★ algoritmi di ricerca locale;

★ algoritmi basati sulla programmazione lineare;

★ algoritmi di programmazione dinamica;

★ algoritmi random.

Dato un generico problema $\mathcal{P}$ di ottimizzazione combinatoria[a], una qualunque tecnica di tipo greedy (greedy=goloso) procede iterativamente.

Inizialmente, l'insieme degli oggetti viene ordinato in accordo ad un prefissato criterio (criterio o funzione greedy) tipicamente legato alla f.o. e tutti gli oggetti sono tutti candidati a far parte della soluzione finale.

A partire da una soluzione vuota, ad ogni it. viene aggiunto alla soluzione parziale in costruzione un nuovo oggetto scelto fra i candidati, ossia quegli oggetti che, se scelti, non ledono l'ammissibilità.
Fra tutti i possibili candidati ad essere aggiunti, viene scelto l'oggetto più promettente, ossia quello cui corrisponde il miglior valore greedy.
L'algoritmo termina quando si è ottenuta una soluzione completa.

[a] Ogni istanza di $\mathcal{P}$ è caratterizzata da un insieme finito di n oggetti e dall'obiettivo di selezionare un sottoinsieme di essi al fine di max o min una certa f.o.

Algoritmi greedy, ②

Nella valutazione della complessità di tempo di un algoritmo greedy, occorre sicuramente tener conto della complessità computazionale necessaria per l'ordinamento iniziale degli oggetti ($O(n \log n)$) e della complessità computazionale necessaria per effettuare gli n test di ammissibilità (che dipende dal problema).

La qualità della soluzione ottenuta dipende dal criterio greedy scelto: differenti criteri greedy conducono a differenti soluzioni.

Se il problema $\mathcal{P}$ è risolvibile all'ottimo con una tecnica greedy, allora per ogni istanza di $\mathcal{P}$ sicuramente esiste un ordinamento che conduce ad una soluzione ottima. Chiaramente, se $\mathcal{P}$ è intrattabile, non ci si aspetta di poter individuare tale ordinamento in tempo polinomiale.

Tuttavia, come vedremo di seguito, per alcuni problemi esistono semplici ordinamenti che conducono a soluzione subottime di buona qualità.

Problemi dello Zaino, ①

Problema: Zaino 0/1

ISTANZA: $O = \{o_1, \ldots, o_n\}, \forall\, o_i \in O\ p_i, w_i \in \mathbb{Z}^+, W \in \mathbb{Z}^+$;

SOLUZIONE: $x \in \{0,1\}^n$ t.c. $x_i = 1$, se o_i viene scelto ($x_i = 0$, altrimenti) e $\sum_{i=1}^{n} w_i \cdot x_i \leq W$;

F.O. z: $\sum_{i=1}^{n} p_i \cdot x_i$;

obj: $\max$.

Problema: Zaino Frazionario

ISTANZA: $O = \{o_1, \ldots, o_n\}, \forall\, o_i \in O\ p_i, w_i \in \mathbb{Z}^+, W \in \mathbb{Z}^+$;

SOLUZIONE: $x \in [0,1]^n$ t.c. $\sum_{i=1}^{n} w_i \cdot x_i \leq W$;

F.O. z: $\sum_{i=1}^{n} p_i \cdot x_i$;

obj: $\max$.

- Zaino Frazionario$\in\mathbf{P}$ risolvibile all'ottimo da un algoritmo greedy polinomiale.
- Zaino 0/1$\in\mathbf{NP}$-completi, ma esiste un algoritmo greedy polinomiale che individua una soluzione $\frac{1}{2}$-approssimata;
- Zaino Frazionario è il rilassamento continuo di Zaino 0/1.

Problemi dello Zaino, ②

algoritmo `greedy-zaino-frazionario` (n, p, w, W)

1 **for** $i = 1$ to n **do** $x_i := 0$;

2 $z := 0$;

3 ordinare e rienumerare gli n oggetti t.c. $\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \cdots \geq \frac{p_n}{w_n}$;

4 $W_r := W$; /* W_r è la capacità residua dello zaino */

5 $i := 1$;

6 **while** $(W_r - w_i \geq 0)$ **do**

7 $\quad x_i := 1$;

8 $\quad z := z + p_i$;

9 $\quad W_r := W_r - w_i$;

10 $\quad i := i + 1$;

11 **endwhile**

12 $x_i := \frac{W_r}{w_i}$;

13 $z := z + p_i \cdot x_i$;

14 **return** (x, z);

end `greedy-zaino-frazionario` /* $O(n \log n)$ */

➠ `linea 12`: $i = s \in \{1, \ldots, n\}$ t.c. $\sum_{j=1}^{s-1} w_j < W \leq \sum_{j=1}^{s} w_j$;

➠ l'oggetto o_s è detto oggetto critico.

Paola Festa - DMA, Università degli Studi di Napoli FEDERICO II.
E-mail: `paola.festa@unina.it` - Web: `http://www.dma.unina.it/∼festa`

Problemi dello Zaino, ③

Data un'istanza I per `Zaino 0/1`, `greedy-zaino-frazionario` individua una soluzione x_I costituita da tutte componenti intere fatta eccezione della componente critica.

È immediato osservare che ponendo a `0` il valore della componente critica, si ottiene una soluzione ammissibile subottima per `Zaino 0/1` in corrispondenza della quale si ha il seguente valore di f.o.

$$z(x_I) = \sum_{j=1}^{s-1} p_j.$$

Ma di quanto x_I è peggiore della soluzione x_I^* ottima per ogni istanza I di `Zaino 0/1`?

Problemi dello Zaino, ④

Si osservi che

$$z(x_I) \leq z(x_I^*) \leq U < z(x_I) + p_s,$$

dove $U = \sum_{j=1}^{s-1} p_j + \left\lfloor W_r \frac{p_s}{w_s} \right\rfloor$ (Dantzig, 1957).

Dunque, in tal caso, il rapporto di prestazione nel caso peggiore è arbitrariamente cattivo.

Si consideri, infatti, la famiglia di istanze per cui

- ✓ $n = 2$;
- ✓ $p_1 = w_1 = 1$, $p_2 = w_2 = k$;
- ✓ $W = k$.

In tal caso, si ha

$$z(x_I) = 1,\ z(x_I^*) = k \Longrightarrow \frac{z(x_I)}{z(x_I^*)} = \frac{1}{k} \xrightarrow{k\to\infty} 0.$$

Problemi dello Zaino, ⑤

algoritmo `appr-greedy-zaino0/1` (n, p, w, W)

for $i = 1$ to n **do** $x_i := 0$;
$z := 0$;
ordinare e rienumerare gli n oggetti t.c. $\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \cdots \geq \frac{p_n}{w_n}$;
$W_r := W$; /* W_r è la capacità residua dello zaino */
$i := 1$;
while $(W_r - w_i \geq 0)$ **do**
  $x_i := 1$;
  $z := z + p_i$;
  $W_r := W_r - w_i$;
  $i = i + 1$;
endwhile
if $(p_i > z)$ **then**
  $x_i = 1$;
  $z = p_i$;
  for $j = 1$ to $i - 1$ **do** $x_j := 0$;
endif
return (x, z);

end `appr-greedy-zaino0/1` /* $O(n \log n)$ */

Paola Festa - DMA, Università degli Studi di Napoli FEDERICO II.
E-mail: paola.festa@unina.it - Web: http://www.dma.unina.it/∼festa

Anche `appr-greedy-zaino0/1` calcola x_I in maniera greedy, ma restituisce in output $z' = \max\{z(x_I), p_s\}$.

Teorema: `appr-greedy-zaino0/1` ha rapporto di prestazione nel caso peggiore pari a $\frac{1}{2}$.

Dimostrazione: È immediato osservare che

$$z(x_I^*) \leq z(x_I) + p_s \overset{z'=\max\{z(x_I),p_s\}}{\Longrightarrow} z(x_I^*) \leq 2z' \Longrightarrow \frac{z^*}{z'} \leq 2.$$

Per dimostrare che $\frac{1}{2}$ è stretto, si consideri la famiglia di istanze per cui

- ✓ $n = 3$;
- ✓ $p_1 = w_1 = 1$, $p_2 = w_2 = p_3 = w_3 = k$;
- ✓ $W = 2k$.

In tal caso, si ha

$$z' = k + 1,\ z(x_I^*) = 2k \Longrightarrow \frac{z'}{z^*} \xrightarrow{k\to\infty} \frac{1}{2}.$$

Max Insieme Indipendente, ①

Problema: Massimo Insieme Indipendente

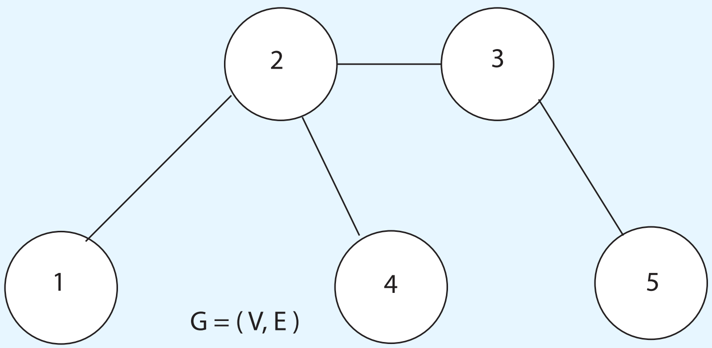
ISTANZA: Grafo non orientato $G = (V, E)$;

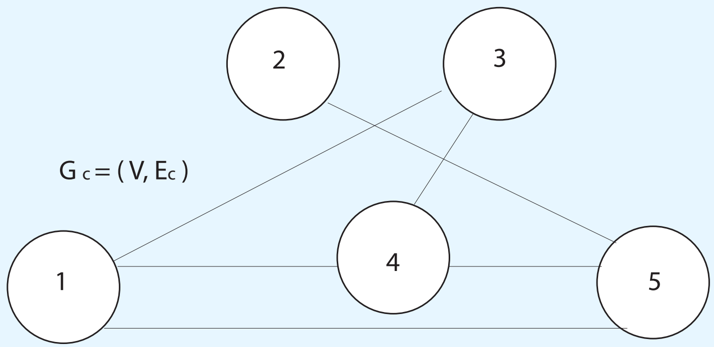
SOLUZIONE: $V' \subseteq V$ t.c. $\forall\, [u, v] \in E$, $u \notin V'$ oppure $v \notin V'$;

F.O. z: $|V'|$;

obj: $\max$.

NOTA: V' è una clique nel grafo complemento $G_c = (V, E_c)$, dove $E_c = V \times V \setminus \{\{(i, i) \mid i \in V\} \cup E\}$.





Come definire il criterio greedy?

Grado $deg(i)$, $i \in V$ (minimo).

Sommario

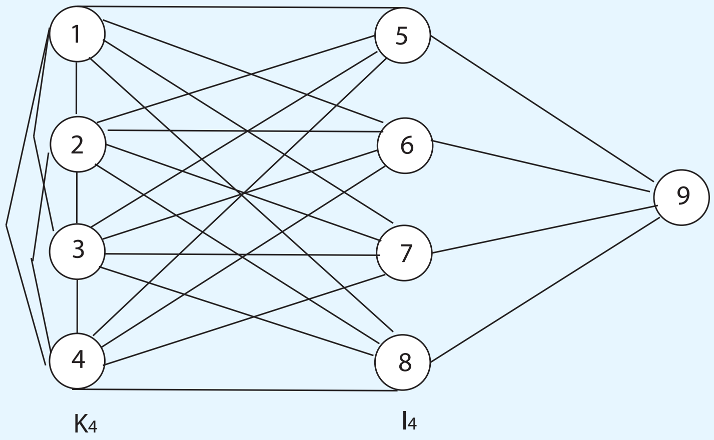
Max Insieme Indipendente, ②

```
algoritmo appr-max-indset (V,E)
    V' := ∅; U := V;
    while (U ≠ ∅) do
        v := argmin{deg(i) | i in G_U = (U, E_U)};    /* G_U sottografo indotto da U */
        V' := V' ∪ {v};
        U := U \ {{v} ∪ {u ∈ U | [v,u] ∈ E_U}};
    endwhile
    return (V');
end appr-max-indset
```

Ma di quanto V_I' è peggiore della soluzione V_I^* ottima per ogni istanza I di `Massimo Insieme Indipendente`?

RISPOSTA:
Il rapporto di prestazione nel caso peggiore è arbitrariamente cattivo.

Max Insieme Indipendente, ③



- $\mathsf{K}_4 = \{1, 2, 3, 4\}$ è una clique di 4 nodi;
- $\mathsf{I}_4 = \{5, 6, 7, 8\}$ è un insieme indipendente di 4 nodi;
- V_I^*? V_I'?

- $V_I^* = \mathsf{I}_4$;
- $V_I' = \{9\} \cup \{i\}$, $i \in \mathsf{K}_4$.

In generale, per $k \geq 2$, $\frac{|V_I'|}{|V_I^*|} = \frac{2}{k} \stackrel{k \to \infty}{\longrightarrow} 0$.

Max Insieme Indipendente, ④

È possibile dimostrare che,

a meno che $\mathbf{P}=\mathbf{NP}$,

NON ESISTE per `Massimo Insieme Indipendente` un algoritmo di approssimazione polinomiale il cui rapporto di prestazione sia costante.

È possibile dimostrare, invece, che `appr-max-indset` ha rapporto di prestazione limitato da una funzione della densità del grafo istanza del problema.

Max Insieme Indipendente, ⑤

Teorema: Sia $G = (V, E)$, $|V| = n$, $|E| = m$ una generica istanza I passata in input a `appr-max-indset`.
Per ogni istanza I, risulta che

$$|V_I'| \geq \frac{n}{2\delta + 1}, \qquad \delta = \frac{m}{n}.$$

Dimostrazione: Sia $v_i \in U$ il vertice selezionato all'i.ma it. del ciclo `while` (linee 2–6) e sia d_i il suo grado.

`appr-max-indset` rimuove da U il nodo v_i e tutti i suoi d_i nodi adiacenti. Dato che ciascuno dei d_i adiacenti ha grado almeno pari a $d_i + 1$, il nr. di archi eliminati è almeno pari a $\frac{d_i(d_i+1)}{2}$.
Sommando per tutte le it., si ha

$$\sum_{i=1}^{|V_I'|} \frac{d_i(d_i + 1)}{2} \leq m \overset{m=\delta n}{=} \delta n. \tag{1}$$

Max Insieme Indipendente, ⑥

Dal momento che `appr-max-indset` termina quando tutti i nodi sono stati eliminati, si ha

$$\sum_{i=1}^{|V_I'|}(d_i+1)=n. \tag{2}$$

Sommando (2) al doppio di (1), si ottiene:

$$\underbrace{\sum_{i=1}^{|V_I'|}(d_i+1)+2\sum_{i=1}^{|V_I'|}\frac{d_i(d_i+1)}{2}}_{\sum_{i=1}^{|V_I'|}(d_i+1)^2}\leq n+2\delta n=n(2\delta+1) \tag{3}$$

Utilizzando la disuguaglianza di Cauchy-Schwarz, per la quale

$$\left(\sum_{k=1}^{l}a_kb_k\right)^2\leq\left(\sum_{k=1}^{l}a_k^2\right)\left(\sum_{k=1}^{l}b_k^2\right),$$

Max Insieme Indipendente, ⑦

nel nostro caso, si ha

$$\left(\sum_{i=1}^{|V_I'|}(d_i+1)^2\right)\left(\sum_{i=1}^{|V_I'|}1^2\right) \geq \left(\sum_{i=1}^{|V_I'|}(d_i+1)\cdot 1\right)^2 \overset{(2)}{=} n^2.$$

Dunque, si ha che

(4)
$$\sum_{i=1}^{|V_I'|}(d_i+1)^2 \geq \frac{n^2}{\sum_{i=1}^{|V_I'|}1^2} = \frac{n^2}{|V_I'|}.$$

Dunque, dalla (4) e la (3), si ottiene:

$$\frac{n^2}{|V_I'|} \leq \sum_{i=1}^{|V_I'|}(d_i+1)^2 \leq n(2\delta+1),$$

da cui segue che

$$|V_I'| \geq \frac{n}{2\delta+1}.$$

Teorema: Sia $G = (V, E)$, $|V| = n$, $|E| = m$ una generica istanza I passata in input a `appr-max-indset` e sia V_I^* una soluzione ottima.
Per ogni istanza I, risulta che

$$\frac{|V_I^*|}{|V_I'|} \leq \delta + 1.$$

Dimostrazione: Sia $v_i \in U$ il vertice selezionato all'i.ma it. del ciclo `while` (linee 2–6) e sia d_i il suo grado.
`appr-max-indset` rimuove da U il nodo v_i e tutti i suoi d_i nodi adiacenti.
Sia k_i il nr. di vertici in V^* che tra i $d_i + 1$ vertici vengono elinimati all'i.ma it.
Chiaramente risulta che

$$\sum_{i=1}^{|V_I'|} k_i = |V_I^*|.$$

Si usano ragionamenti analoghi alla dim. del precedente teorema.

Problema: `Vertex Cover Minimo`

ISTANZA: Grafo non orientato $G = (V, E)$;

SOLUZIONE: $V' \subseteq V$ t.c. $\forall\, [u, v] \in E$, $u \in V'$ oppure $v \in V'$;

F.O. z: $|V'|$;

obj: $\min$.

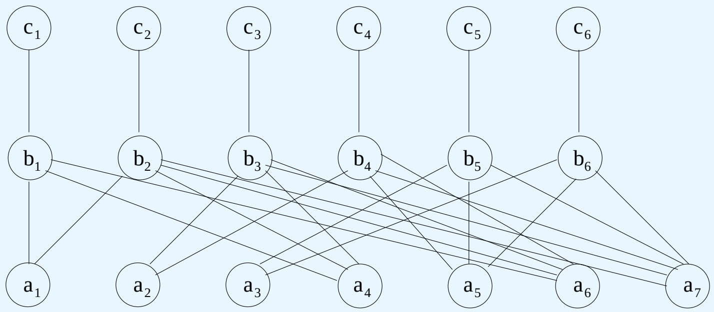
Come definire il criterio greedy?

Grado $deg(i)$, $i \in V$ (minimo).

Ma di quanto V_I' è peggiore della soluzione V_I^* ottima per ogni istanza I di `Vertex Cover Minimo`?

RISPOSTA:
Anche in questo caso, il rapporto di prestazione nel caso peggiore è arbitrariamente cattivo.

Vertex Cover Minimo, ②



- $V =$ $$\{c_1, \ldots, c_n\} \cup \{b_1, \ldots, b_n\} \cup \left\{a_i,\ i = 1, \ldots, L(n) = \sum_{j=2}^{n-1} \left\lfloor \frac{n}{j} \right\rfloor \right\};$$
- V_I^*? V_I'?
- $V_I^* = \{b_1, \ldots, b_n\}$;
- nel caso peggiore $V_I' = \{a_i,\ i = 1, \ldots, L(n)\} \cup \{b_1, \ldots, b_n\}$ oppure $V_I' = \{a_i,\ i = 1, \ldots, L(n)\} \cup \{c_1, \ldots, c_n\}$.

A differenza di `Massimo Insieme Indipendente`, per `Vertex Cover Minimo` è possibile dimostrare che esiste un algoritmo di approssimazione polinomiale con rapporto di prestazione costante.

algoritmo `random-approx-vc` (V, E)

$V' := \emptyset$; $E' := E$;

while $(E' \neq \emptyset)$ **do**

  $[i, j] :=$`select`(E'); /* seleziona da E' un arco a caso */

  $V' := V' \cup \{i, j\}$;

  $E' := E' \setminus \{[u, v] \mid v = i$ o $u = j\}$;

endwhile

return (V');

end `random-approx-vc`

Paola Festa - DMA, Università degli Studi di Napoli FEDERICO II.
E-mail: `paola.festa@unina.it` - Web: `http://www.dma.unina.it/∼festa`

Vertex Cover Minimo, ④

Teorema: Sia $G = (V, E)$, $|V| = n$, $|E| = m$ una generica istanza I passata in input a `random-approx-vc` e sia V_I^* una soluzione ottima.
Per ogni istanza I, risulta che

$$\frac{|V_I'|}{|V_I^*|} \leq 2.$$

Dimostrazione: Sia A l'insieme degli archi selezionati dall'algoritmo. Chiaramente, risulta che

$$|V_I'| = 2\,|A|. \tag{5}$$

Inoltre due archi in A non possono avere nessun vertice incidente in comune. Ne consegue che

$$|A| \leq |V_I^*|. \tag{6}$$

Unendo le relazioni (5) e (6), si ottiene la tesi.

□