Algoritmi di approssimazione per lo zaino (2/2)

Sapienza Università di Roma

Indice

Garanzie di prestazione
- Il rapporto di prestazione nel caso peggiore di GR
- Il rapporto di prestazione nel caso peggiore di GR^e
- Il rapporto di prestazione nel caso peggiore di H^2
- Il rapporto di prestazione nel caso peggiore di H^ℓ
- Schemi di approssimazione
- Il rapporto di prestazione nel caso peggiore dell'algoritmo
- Basic-FPTAS

Garanzia di prestazione assoluta

Errori tipici

Implementazione in C++

In sintesi

Dove eravamo

- Nella prima parte abbiamo descritto cinque algoritmi per il problema dello zaino 0/1 (KP), con gli oggetti ordinati per efficienza non crescente e l'oggetto critico s:
  - l'algoritmo greedy GR e l'algoritmo Ext-Greedy GR^e;
  - gli algoritmi H^2 e H^ℓ, che scommettono sugli oggetti di profitto maggiore di una soluzione ottima;
  - l'algoritmo Basic-FPTAS FP^ε, che arrotonda per difetto i profitti.

- In questa parte dimostriamo quanto ciascuno di essi può essere lontano dall'ottimo, nel caso peggiore.

Garanzie di prestazione

Introduzione: Garanzie di prestazione

- Determiniamo ora il rapporto di prestazione nel caso peggiore degli algoritmi delle sezioni precedenti
- Per ogni algoritmo: un limite superiore al rapporto su ogni istanza, e una famiglia di istanze che mostra che il limite è tight
- Alla fine otteniamo due schemi di approssimazione, un PTAS e un FPTAS

Il rapporto di prestazione

- Ricordiamo che, essendo il KP un problema di massimo, il rapporto di prestazione di un algoritmo $\mathscr{A}$ su un'istanza I è

$$\frac{z(I)}{\mathscr{A}(I)} \geq 1.$$

- $r(\mathscr{A})$ è il suo estremo superiore su tutte le istanze: il **rapporto di prestazione nel caso peggiore** di $\mathscr{A}$.

Il rapporto di prestazione nel caso peggiore di GR

Proposizione 1

Esistono istanze I del KP tali che:

$$\frac{z(I)}{GR(I)} \to +\infty \qquad \text{per } c \to +\infty.$$

La dimostrazione costruisce una famiglia di istanze con due oggetti.

Dimostrazione (1/2)

Fissiamo un intero $\vartheta > 2$ e consideriamo l'istanza I_ϑ del KP con capacità dello zaino $c = \vartheta$ e l'insieme di oggetti $\mathscr{I} = \{1, 2\}$, dove i profitti e i pesi degli oggetti sono

$$p_1 = 2, \quad w_1 = 1, \qquad \text{e} \qquad p_2 = \vartheta, \quad w_2 = \vartheta.$$

Una soluzione ottima inserisce l'oggetto 2, quindi

$$z(I_\vartheta) = \vartheta.$$

Dimostrazione (2/2)

Poiché

$$\frac{p_1}{w_1} = 2 > \frac{p_2}{w_2} = \frac{\vartheta}{\vartheta} = 1,$$

GR inserisce per primo l'oggetto 1, lasciando capacità disponibile $\vartheta - 1$, quindi l'oggetto 2 non si può aggiungere. Quindi

$$GR(I_\vartheta) = 2.$$

Pertanto,

$$\lim_{\vartheta \to +\infty} \frac{z(I_\vartheta)}{GR(I_\vartheta)} = \lim_{\vartheta \to +\infty} \frac{\vartheta}{2} = +\infty.$$

□

GR non ha una garanzia costante

- La proposizione mostra che non esiste una costante $\alpha \in [1, +\infty)$ tale che

$$\frac{z(I)}{GR(I)} \leq \alpha \qquad \text{per ogni istanza } I.$$

- Equivalentemente, il rapporto $\frac{z(I)}{GR(I)}$ può diventare grande a piacere con una scelta opportuna dell'istanza.
- In altre parole, non esiste una costante $\alpha \in [1, +\infty)$ tale che *GR* sia un algoritmo di α-approssimazione, e il rapporto di prestazione nel caso peggiore di *GR* è illimitato:

$$r(GR) = +\infty.$$

Il difetto fondamentale di *GR*

- Un difetto fondamentale di *GR* è che può non selezionare l'oggetto di profitto massimo (anche quando quel solo oggetto è ottimo).
- In particolare, dando la priorità all'efficienza p_i/w_i, *GR* può riempire lo zaino con oggetti di efficienza alta e lasciare una capacità disponibile insufficiente per l'oggetto di profitto massimo.

Il rapporto di prestazione nel caso peggiore di GR^e

Proposizione 2

Per ogni istanza I del KP, abbiamo:

$$\frac{z(I)}{GR^e(I)} \leq 2.$$

La dimostrazione confronta $z(I)$ con il rilassamento lineare, scritto con l'oggetto critico,
e $GR^e(I)$ con i suoi due bound.

Dimostrazione (1/2)

Supponiamo che gli oggetti siano ordinati per efficienza non crescente. Siano s l'oggetto critico e rc la capacità residua. Abbiamo

$$\underbrace{\sum_{i=1}^{s-1} p_i \; + \; p_s \frac{rc}{w_s}}_{=z\left(\mathrm{LP_{KP}}\right)} \; \leq \; \sum_{i=1}^{s-1} p_i \; + \; p_s \qquad \text{poiché} \qquad rc < w_s,$$

dove $z\left(\mathrm{LP_{KP}}\right)$ è il valore ottimo del rilassamento lineare della formulazione PLI naturale del KP.

Di conseguenza,

$$z(I) \leq z\left(\mathrm{LP_{KP}}\right) \leq \sum_{i=1}^{s-1} p_i + p_s \leq \sum_{i=1}^{s-1} p_i + p_{\max},$$

poiché per definizione di $p_{\max}$ abbiamo $p_{\max} \geq p_s$.

Dimostrazione (2/2)

Inoltre,

$$GR^e(I) \geq GR(I) \geq \sum_{i=1}^{s-1} p_i \quad \text{e} \quad GR^e(I) \geq p_{\max}.$$

Sommando queste due disuguaglianze otteniamo

$$2\, GR^e(I) = GR^e(I) + GR^e(I) \;\geq\; \sum_{i=1}^{s-1} p_i + p_{\max}.$$

Pertanto,

$$z(I) \leq \sum_{i=1}^{s-1} p_i + p_{\max} \;\leq\; 2\, GR^e(I).$$

Dividendo per $GR^e(I) > 0$ otteniamo

$$\frac{z(I)}{GR^e(I)} \leq 2.$$

La garanzia 2 è tight

Proposizione 3

Esistono istanze I del KP tali che:

$$\frac{z(I)}{GR^e(I)} \to 2 \quad \text{per} \quad c \to +\infty.$$

La dimostrazione costruisce una famiglia di istanze con tre oggetti.

Dimostrazione (1/2)

Fissiamo un intero $\vartheta > 2$ e consideriamo l'istanza I_ϑ del KP con capacità $c = 2\vartheta$ e 3 oggetti:

$$p_1 = 2,\ w_1 = 1, \qquad p_2 = \vartheta,\ w_2 = \vartheta, \qquad p_3 = \vartheta,\ w_3 = \vartheta.$$

Gli oggetti 2 e 3 entrano insieme nello zaino poiché $w_2 + w_3 = 2\vartheta = c$; quindi una soluzione ottima inserisce gli oggetti 2 e 3 e

$$z(I_\vartheta) = 2\vartheta.$$

Inoltre,

$$\frac{p_1}{w_1} = 2 > \frac{p_2}{w_2} = \frac{p_3}{w_3} = 1,$$

quindi *GR* inserisce per primo l'oggetto 1.

Dimostrazione (2/2)

Dopo l'inserimento dell'oggetto 1, la capacità disponibile è $2\vartheta - 1$, quindi si può aggiungere esattamente uno degli oggetti 2 e 3. Pertanto *GR* restituisce una soluzione ammissibile di valore $\vartheta + 2$.

Poiché $p_{\max} = \max\{p_1, p_2, p_3\} = \vartheta$, abbiamo $\vartheta + 2 > p_{\max}$, e quindi

$$GR^e(I_\vartheta) = \vartheta + 2.$$

Di conseguenza,

$$\lim_{\vartheta \to +\infty} \frac{z(I_\vartheta)}{GR^e(I_\vartheta)} = \lim_{\vartheta \to +\infty} \frac{2\vartheta}{\vartheta + 2} = \lim_{\vartheta \to +\infty} \frac{2}{1 + \frac{2}{\vartheta}} = 2.$$

□

GR^e è un algoritmo di 2-approssimazione

- Dalle due proposizioni precedenti segue che il rapporto di prestazione nel caso peggiore di GR^e è due, cioè abbiamo

$$r(GR^e) = 2$$

  e GR^e è un algoritmo di 2-approssimazione.

- Un punto di forza di GR^e è che attenua il difetto principale di GR: anche se l'inserimento greedy perde un oggetto molto redditizio a causa dell'ordine per efficienza, GR^e confronta esplicitamente la soluzione greedy con la migliore soluzione ammissibile con un solo oggetto e restituisce la migliore, garantendo così un fattore di approssimazione costante.

Il limite di GR^e

- Tuttavia, la garanzia $r(GR^e) = 2$ è tight: esistono istanze in cui GR^e è ingannato dall'efficienza e, dopo aver selezionato all'inizio un oggetto di efficienza alta, non ha più capacità sufficiente per inserire insieme i due oggetti di profitto maggiore (che formano la soluzione ottima), con un rapporto vicino a piacere a 2.

Verso algoritmi migliori

- Una soluzione che nel caso peggiore vale solo la metà della soluzione ottima non è un risultato molto soddisfacente.
- Conviene quindi cercare un algoritmo di approssimazione migliore, eventualmente al prezzo di un tempo di calcolo maggiore.
- Nell'esempio di caso peggiore precedente GR^e commette l'"errore" di escludere l'oggetto 3, che ha un profitto molto alto. Per evitare questo errore servono algoritmi euristici più avanzati.

Il rapporto di prestazione nel caso peggiore di H^2

Proposizione 4

Per ogni istanza I del KP, abbiamo:

$$\frac{z(I)}{H^2(I)} \leq \frac{4}{3}.$$

La dimostrazione guarda la coppia di oggetti di profitto maggiore di una soluzione ottima,
e si divide in due casi.

Dimostrazione (1/6)

Se $z(I)$ è dato da un solo oggetto, abbiamo $H^2(I) = z(I)$.

Altrimenti consideriamo una coppia ammissibile $(\tilde{\imath}, \tilde{k}) \in \mathscr{P}$ che contribuisce con il profitto più alto in una soluzione ottima.

- L'istanza residua è $I(\tilde{\imath}, \tilde{k})$ e l'insieme residuo di oggetti è $\mathscr{I}(\tilde{\imath}, \tilde{k})$.
- L'algoritmo H^2 considera questa coppia di oggetti durante la sua esecuzione e applica GR^e all'istanza residua $I(\tilde{\imath}, \tilde{k})$.

La coppia contiene i due oggetti di profitto maggiore di una soluzione ottima, quindi ogni altro oggetto di quella soluzione ha profitto al più $\min\{p_{\tilde{\imath}}, p_{\tilde{k}}\}$ ed entra nella capacità residua: appartiene a $\mathscr{I}(\tilde{\imath}, \tilde{k})$.

Dimostrazione (2/6)

Il valore ottimo si può quindi scrivere come:

$$z(I) = z\big(I(\tilde{i}, \tilde{k})\big) + p_{\tilde{i}} + p_{\tilde{k}} \qquad (*)$$

dove $z\big(I(\tilde{i}, \tilde{k})\big)$ è il valore ottimo dell'istanza residua $I(\tilde{i}, \tilde{k})$.

Dalle operazioni di H^2, abbiamo:

$$H^2(I) \quad \geq \quad p_{\tilde{i}} + p_{\tilde{k}} + GR^e\big(I(\tilde{i}, \tilde{k})\big) \qquad (**)$$

e poiché GR^e è un algoritmo di 2-approssimazione, cioè $GR^e(I) \geq \frac{1}{2}\, z(I)$, abbiamo:

$$GR^e\big(I(\tilde{i}, \tilde{k})\big) \quad \geq \quad \frac{1}{2}\, z\big(I(\tilde{i}, \tilde{k})\big)$$

Dimostrazione (3/6)

Di conseguenza:

$$H^2(I) \geq p_{\tilde{i}} + p_{\tilde{k}} + \frac{1}{2}\, z\big(I(\tilde{i}, \tilde{k})\big)$$

Inoltre, per la $(*)$, abbiamo:

$$H^2(I) \geq p_{\tilde{i}} + p_{\tilde{k}} + \frac{1}{2}\big(z(I) - p_{\tilde{i}} - p_{\tilde{k}}\big) = \frac{1}{2}\big(z(I) + p_{\tilde{i}} + p_{\tilde{k}}\big)$$

Consideriamo due casi possibili:

$$i)\quad p_{\tilde{i}} + p_{\tilde{k}} \geq \frac{1}{2}\, z(I) \qquad \text{e} \qquad ii)\quad p_{\tilde{i}} + p_{\tilde{k}} < \frac{1}{2}\, z(I)$$

Dimostrazione (4/6)

Caso i):

$$p_{\tilde{i}} + p_{\tilde{k}} \geq \frac{1}{2} z(I)$$

In questo caso abbiamo:

$$H^2(I) \geq \frac{1}{2}\left(z(I) + p_{\tilde{i}} + p_{\tilde{k}}\right)$$

$$\geq \frac{1}{2}\left(z(I) + \frac{1}{2} z(I)\right) = \frac{3}{4} z(I)$$

e di conseguenza

$$H^2(I) \geq \frac{3}{4} z(I)$$

Dimostrazione (5/6)

Caso *ii*):

$$p_{\tilde{i}} + p_{\tilde{k}} < \frac{1}{2}\, z(I)$$

Poiché i loro profitti sommano a meno di $\frac{1}{2}\, z(I)$, almeno uno dei due oggetti $\tilde{i}$, $\tilde{k}$ ha profitto minore di $\frac{1}{4}\, z(I)$.

Dal rilassamento $\mathrm{LP}_{\mathrm{KP}}$ dell'istanza residua $I(\tilde{i}, \tilde{k})$, abbiamo:

$$\begin{aligned} z\big(I(\tilde{i}, \tilde{k})\big) &\leq z\big(\mathrm{LP}_{\mathrm{KP}}\big) \\ &\leq GR^e\big(I(\tilde{i}, \tilde{k})\big) + p_s \\ &\leq GR^e\big(I(\tilde{i}, \tilde{k})\big) + \frac{1}{4}\, z(I) \end{aligned}$$

dove s è l'oggetto critico di $I(\tilde{i}, \tilde{k})$ e $p_s < \frac{1}{4} z(I)$ poiché tutti i suoi oggetti hanno profitto al più $\min\{p_{\tilde{i}}, p_{\tilde{k}}\} < \frac{1}{4} z(I)$ (per costruzione dell'istanza residua).

Dimostrazione (6/6)

Se tutti gli oggetti dell'istanza residua entrano nella capacità residua non c'è oggetto critico: allora GR^e li inserisce tutti, $z\big(I(\tilde{i},\tilde{k})\big) = GR^e\big(I(\tilde{i},\tilde{k})\big)$, e le disuguaglianze valgono con $p_s := 0$. Abbiamo:

$$z(I) = p_{\tilde{i}} + p_{\tilde{k}} + z\big(I(\tilde{i},\tilde{k})\big)$$

$$\leq \underbrace{p_{\tilde{i}} + p_{\tilde{k}} + GR^e\big(I(\tilde{i},\tilde{k})\big)}_{\leq H^2(I),\ \text{per la } (**)} + \frac{1}{4}\, z(I)$$

$$\leq H^2(I) + \frac{1}{4}\, z(I)$$

e di conseguenza $H^2(I) \geq \frac{3}{4}\, z(I)$.

In entrambi i casi $H^2(I) \geq \frac{3}{4}\, z(I)$, cioè, dividendo per $H^2(I) > 0$,

$$\frac{z(I)}{H^2(I)} \leq \frac{4}{3}.$$

La garanzia $\frac{4}{3}$ è tight

Proposizione 5

Esistono istanze I_M del KP tali che:

$$\frac{z(I_M)}{H^2(I_M)} \rightarrow \frac{4}{3} \qquad \textit{per} \qquad M \rightarrow +\infty.$$

La dimostrazione costruisce una famiglia di istanze con un oggetto piccolo e quattro oggetti grandi.

Dimostrazione (1/2)

Consideriamo l'istanza I_M del KP con capacità $c = 4\,M$ e 5 oggetti:

$$p_1 = 2, \quad w_1 = 1, \quad \text{e}$$

$$p_2 = p_3 = p_4 = p_5 = M, \quad w_2 = w_3 = w_4 = w_5 = M,$$

dove $M \in \mathbb{Z}_{\geq 1}, M > 2$ è un numero intero.

Una soluzione ottima inserisce i quattro oggetti grandi, poiché $w_2 + w_3 + w_4 + w_5 = 4\,M = c$, quindi

$$z(I_M) = 4\,M.$$

Le efficienze sono

$$\frac{p_1}{w_1} = 2 \quad > \quad \frac{p_2}{w_2} = \frac{p_3}{w_3} = \frac{p_4}{w_4} = \frac{p_5}{w_5} = \frac{M}{M} = 1$$

quindi l'oggetto 1 viene per primo nell'ordine per efficienza.

Dimostrazione (2/2)

Consideriamo i due tipi possibili di coppia esaminati da H^2:

1. una coppia di due oggetti grandi: la capacità residua è $2\,M$ e l'istanza residua mantiene l'oggetto 1 e i due oggetti grandi rimanenti, quindi GR^e inserisce l'oggetto 1 e un oggetto grande, per un valore totale di $3\,M + 2$;
2. una coppia formata dall'oggetto 1 e da un oggetto grande: l'istanza residua è vuota, perché nessun oggetto ha profitto al più $\min\{2, M\} = 2$, per un valore totale di $M + 2$.

Il primo tipo di coppia dà il valore maggiore, quindi $H^2(I_M) = 3\,M + 2$. Di conseguenza,

$$\lim_{M\to\infty} \frac{z(I_M)}{H^2(I_M)} = \lim_{M\to\infty} \frac{4\,M}{3\,M+2} = \lim_{M\to\infty} \frac{4}{3+\frac{2}{M}} = \frac{4}{3}.$$

□

H^2 è un algoritmo di $\frac{4}{3}$-approssimazione

- Dalle due proposizioni precedenti segue che il rapporto di prestazione nel caso peggiore di H^2 è quattro terzi, cioè abbiamo

$$r(H^2) = \frac{4}{3}$$

  e H^2 è un algoritmo di $\frac{4}{3}$-approssimazione.

- Scommettere su due oggetti invece che su uno fa scendere il rapporto dal 2 di GR^e a $\frac{4}{3}$.

Il rapporto di prestazione nel caso peggiore di H^ℓ

Proposizione 6

Per ogni istanza I del KP, abbiamo:

$$\frac{z(I)}{H^\ell(I)} \leq \frac{\ell+2}{\ell+1}.$$

La dimostrazione segue quella per H^2, con l'insieme degli ℓ oggetti di profitto maggiore di una soluzione ottima al posto della coppia.

Dimostrazione (1/4)

Se una soluzione ottima contiene al più $\ell - 1$ oggetti, l'algoritmo la trova fra gli insiemi di $\widehat{T}(\ell)$, e $H^{\ell}(I) = z(I)$.

Altrimenti, sia $\tilde{L} \in T(\ell)$ l'insieme degli ℓ oggetti di profitto maggiore di una soluzione ottima. L'algoritmo considera $\tilde{L}$ durante la sua esecuzione e applica GR^{e} all'istanza residua $I(\tilde{L})$. Tutti gli altri oggetti della soluzione ottima appartengono a $\mathscr{I}(\tilde{L})$, quindi

$$z(I) = \sum_{i \in \tilde{L}} p_i + z\big(I(\tilde{L})\big), \qquad (*)$$

dove $z\big(I(\tilde{L})\big)$ è il valore ottimo dell'istanza residua.

Dimostrazione (2/4)

Dalle operazioni di H^ℓ, e poiché GR^e è un algoritmo di 2-approssimazione,

$$H^\ell(I) \geq \sum_{i \in \tilde{L}} p_i + GR^e\big(I(\tilde{L})\big)$$

$$\geq \sum_{i \in \tilde{L}} p_i + \frac{1}{2} z\big(I(\tilde{L})\big) = \frac{1}{2}\Big(z(I) + \sum_{i \in \tilde{L}} p_i\Big),$$

dove l'ultima uguaglianza segue dalla $(*)$. Consideriamo due casi.

Caso i): $\sum_{i \in \tilde{L}} p_i \geq \frac{\ell}{\ell+2} z(I)$. Allora

$$H^\ell(I) \geq \frac{1}{2}\left(z(I) + \frac{\ell}{\ell+2} z(I)\right) = \frac{\ell+1}{\ell+2} z(I).$$

Dimostrazione (3/4)

Caso *ii*): $\sum_{i \in \tilde{L}} p_i < \frac{\ell}{\ell+2} z(I)$.

- Allora almeno un oggetto di $\tilde{L}$ ha profitto minore di $\frac{1}{\ell+2} z(I)$, e lo stesso vale per tutti gli oggetti dell'istanza residua, i cui profitti sono al più $\min\{p_i : i \in \tilde{L}\}$.
- Sia s l'oggetto critico di $I(\tilde{L})$ (se non esiste, GR^e inserisce tutti gli oggetti di $I(\tilde{L})$, e le disuguaglianze seguenti valgono con $p_s := 0$).

Il rilassamento lineare dell'istanza residua dà

$$\begin{aligned} z\big(I(\tilde{L})\big) \;\leq\; z\big(\mathrm{LP}_{\mathrm{KP}}\big) \;&\leq\; GR^e\big(I(\tilde{L})\big) + p_s \\ &<\; GR^e\big(I(\tilde{L})\big) + \frac{1}{\ell+2} z(I), \end{aligned}$$

Dimostrazione (4/4)

e quindi, per la $(*)$,

$$z(I) < \sum_{i \in \tilde{L}} p_i + GR^e\left(I(\tilde{L})\right) + \frac{1}{\ell+2} z(I)$$

$$\leq H^\ell(I) + \frac{1}{\ell+2} z(I),$$

cioè $H^\ell(I) > \frac{\ell+1}{\ell+2} z(I)$.

In entrambi i casi $H^\ell(I) \geq \frac{\ell+1}{\ell+2} z(I)$, e dividendo per $H^\ell(I) > 0$ si ottiene l'enunciato. □

La garanzia $\frac{\ell+2}{\ell+1}$ è tight

Proposizione 7

Esistono istanze I_M del KP tali che:

$$\frac{z(I_M)}{H^\ell(I_M)} \to \frac{\ell+2}{\ell+1} \qquad \textit{per} \qquad M \to +\infty.$$

La dimostrazione generalizza l'istanza usata per H^2.

Dimostrazione (1/2)

Fissiamo un intero $M > 2$ e consideriamo l'istanza I_M del KP con capacità $c = (\ell + 2)\, M$ e $n = \ell + 3$ oggetti, dove

$$p_1 = 2, \quad w_1 = 1, \quad \text{e} \quad p_j = w_j = M \quad \text{per} \quad j \in \{2, 3, \ldots, n\}.$$

- Una soluzione ottima inserisce gli $\ell + 2$ oggetti di peso M, che riempiono lo zaino, quindi $z(I_M) = (\ell + 2)\, M$.
- L'oggetto 1 viene per primo nell'ordine per efficienza, poiché $\frac{p_1}{w_1} = 2 > 1$.

Dimostrazione (2/2)

Gli insiemi L esaminati da H^ℓ sono di due tipi:

1. L formato da ℓ oggetti di profitto M: la capacità residua è $2M$, l'istanza residua mantiene l'oggetto 1 e i due oggetti rimanenti di profitto M, e GR^e inserisce l'oggetto 1 e uno di essi, per un valore totale di $\ell M + M + 2 = (\ell + 1)\, M + 2$;
2. L che contiene l'oggetto 1 e $\ell - 1$ oggetti di profitto M: l'istanza residua è vuota, poiché nessun oggetto ha profitto al più $\min\{2, M\} = 2$, per un valore totale di $(\ell - 1)\, M + 2$.

Anche gli insiemi di al più $\ell - 1$ oggetti danno al più $(\ell - 1)\, M + 2$.
Quindi $H^\ell(I_M) = (\ell + 1)\, M + 2$ e

$$\lim_{M\to\infty} \frac{z(I_M)}{H^\ell(I_M)} = \lim_{M\to\infty} \frac{(\ell + 2)\, M}{(\ell + 1)\, M + 2} = \lim_{M\to\infty} \frac{\ell + 2}{\ell + 1 + \frac{2}{M}} = \frac{\ell + 2}{\ell + 1}.$$

Per $\ell = 2$ è l'istanza usata per H^2. □

H^ℓ è un algoritmo di $\frac{\ell+2}{\ell+1}$-approssimazione

- Dalle due proposizioni precedenti segue che il rapporto di prestazione nel caso peggiore di H^ℓ è

$$r(H^\ell) = \frac{\ell + 2}{\ell + 1}$$

  e H^ℓ è un algoritmo di $\frac{\ell+2}{\ell+1}$-approssimazione.

- Per $\ell = 2$ ritroviamo $r(H^2) = \frac{4}{3}$, e il rapporto tende a 1 al crescere di ℓ.

Schemi di approssimazione: le definizioni

Richiamiamo le definizioni del capitolo «Principi degli algoritmi di approssimazione».

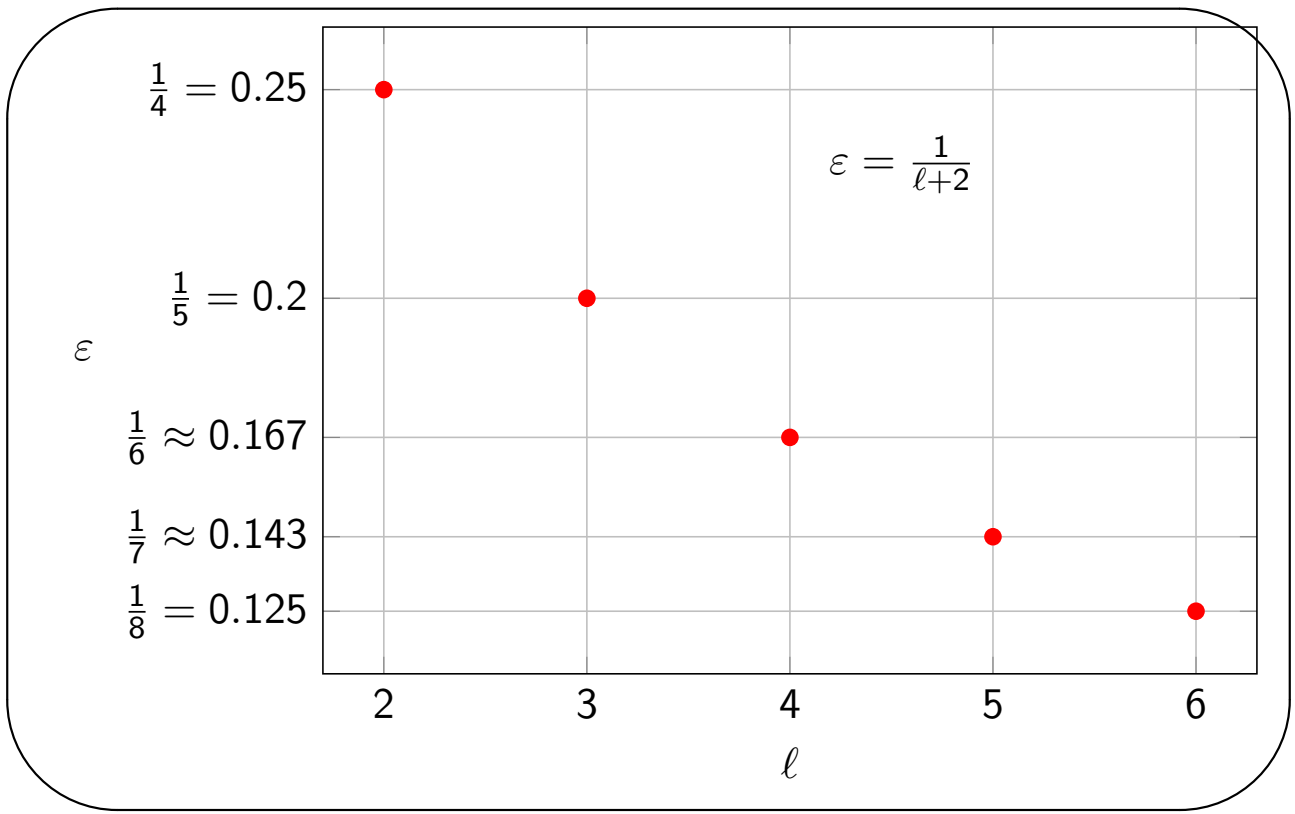
- L'**errore relativo** di un algoritmo $\mathscr{A}$ su un'istanza I del KP è $\dfrac{z(I) - \mathscr{A}(I)}{z(I)}$.
- Uno **schema di approssimazione** è una famiglia di algoritmi $\{\mathscr{A}_\varepsilon\}_{\varepsilon \in (0,1)}$ tale che, per ogni ε e ogni istanza I, l'errore relativo di $\mathscr{A}_\varepsilon$ è al più ε.
- Per un problema di massimo questo significa che $\mathscr{A}_\varepsilon$ è un algoritmo di $\frac{1}{1-\varepsilon}$-approssimazione.
- Lo schema è un **PTAS** se, per ogni ε fissato, il tempo di calcolo di $\mathscr{A}_\varepsilon$ è polinomiale nella dimensione dell'istanza.
- È un **FPTAS** se il tempo di calcolo è polinomiale sia nella dimensione dell'istanza sia nell'accuratezza $\frac{1}{\varepsilon}$.

L'errore relativo di H^ℓ

Per la garanzia di H^ℓ, $H^\ell(I) \geq \frac{\ell+1}{\ell+2}\, z(I)$, quindi l'errore relativo di H^ℓ è al più

$$1 - \frac{\ell+1}{\ell+2} = \frac{1}{\ell+2},$$

che tende a 0 al crescere di ℓ:



L'algoritmo H^ε

- La relazione si può leggere al contrario: dato l'errore relativo ε da raggiungere, basta scegliere ℓ con $\frac{1}{\ell+2} \leq \varepsilon$, cioè $\ell \geq \frac{1}{\varepsilon} - 2$.

- Scegliendo

$$\ell(\varepsilon) = \min\left\{\max\left\{\left\lceil\frac{1}{\varepsilon}\right\rceil - 2,\ 2\right\},\ n\right\}$$

  otteniamo l'algoritmo H^ε, cioè H^ℓ con $\ell = \ell(\varepsilon)$, il cui errore relativo è al più ε per ogni $\varepsilon \in (0, 1)$.

- Per $\varepsilon \geq \frac{1}{4}$ abbiamo $\ell(\varepsilon) = 2$, e H^ε è l'algoritmo H^2.

- La dimensione ℓ degli insiemi scommessi non può superare n, e con $\ell = n$ l'algoritmo enumera tutte le soluzioni ammissibili ed è esatto.

H^ε è un PTAS

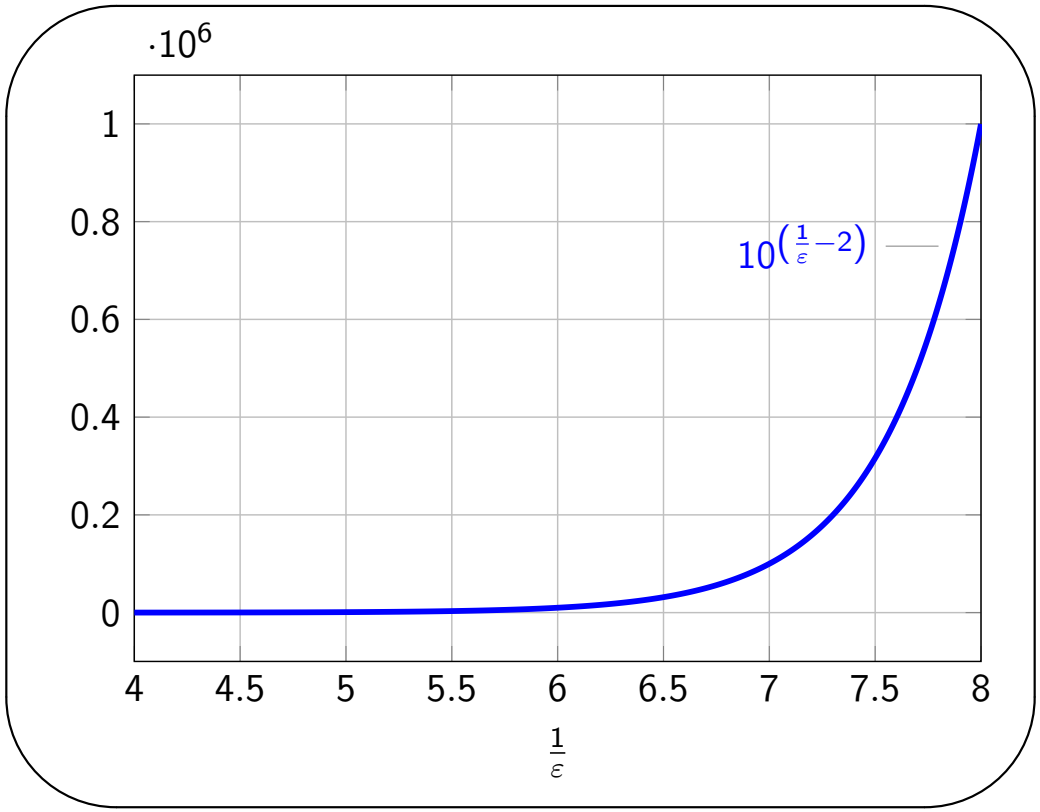
Osservazione 1

La famiglia $\{H^\varepsilon\}_{\varepsilon\in(0,1)}$ è un PTAS per il problema dello zaino 0/1.

- Per ogni ε fissato, $\ell(\varepsilon)$ è una costante, e per l'osservazione sul tempo di calcolo di H^ℓ il tempo di calcolo $O\left(n^{\ell(\varepsilon)+1}\right)$ è polinomiale nella dimensione n dell'istanza.
- Tuttavia, H^ε **non** è un FPTAS: l'esponente $\ell(\varepsilon) = \left\lceil \frac{1}{\varepsilon} \right\rceil - 2$ cresce con l'accuratezza $\frac{1}{\varepsilon}$, quindi il tempo di calcolo non è polinomiale in $\frac{1}{\varepsilon}$.

Il tempo di calcolo di H^{ε}

Il grafico mostra, per $n = 10$ oggetti e $4 \leq \frac{1}{\varepsilon} \leq 8$, il numero $n^{\frac{1}{\varepsilon}-2}$, un lower bound su $n^{\ell(\varepsilon)}$ (il limite $\ell(\varepsilon) \leq n$ non interviene, poiché $\frac{1}{\varepsilon} \leq 12$):



H^ε in pratica

- L'algoritmo H^ε è un metodo semplice e abbastanza facile da implementare.
- Tuttavia, il suo tempo di calcolo esplode al crescere dell'accuratezza richiesta.
- Anche per un errore relativo moderato $\varepsilon = \frac{1}{10}$, cioè uno scostamento del 10 per cento dal valore ottimo, abbiamo $\ell(\varepsilon) = 8$ e un tempo di calcolo $O(n^9)$, o comunque $O(n^8)$ con l'implementazione migliorata.

Il rapporto di prestazione nel caso peggiore dell'algoritmo Basic-FPTAS

Proposizione 8

Per ogni istanza I del KP e ogni $\varepsilon \in (0,1)$, abbiamo:

$$\frac{z(I)}{FP^{\varepsilon}(I)} \leq \frac{1}{1-\varepsilon}.$$

La dimostrazione confronta la soluzione restituita con una soluzione ottima,
attraverso i profitti scalati.

Dimostrazione (1/2)

Siano $\tilde{S}$ una soluzione ottima di I e S la soluzione restituita da FP^{ε}, cioè una soluzione ottima dell'istanza scalata I'. Poiché I e I' hanno le stesse soluzioni ammissibili, $\tilde{S}$ è ammissibile anche per I', e per l'ottimalità di S per I',

$$\sum_{j \in S} p'_j \geq \sum_{j \in \tilde{S}} p'_j.$$

Per definizione dei profitti scalati, $K\, p'_j \leq p_j$ e $K\, p'_j > p_j - K$ per ogni oggetto j. Pertanto

$$FP^{\varepsilon}(I) = \sum_{j \in S} p_j \geq K \sum_{j \in S} p'_j \geq K \sum_{j \in \tilde{S}} p'_j$$

$$> \sum_{j \in \tilde{S}} (p_j - K) = z(I) - |\tilde{S}|\, K.$$

Dimostrazione (2/2)

Poiché una soluzione ottima contiene al più n oggetti e $K = \varepsilon\, GR^e(I)/n$,

$$|\tilde{S}|\, K \;\leq\; n\, K \;=\; \varepsilon\, GR^e(I) \;\leq\; \varepsilon\, z(I),$$

dove l'ultima disuguaglianza vale perché $GR^e(I)$ è il valore di una soluzione ammissibile.

Combinando le due relazioni,

$$FP^{\varepsilon}(I) \;\geq\; z(I) - \varepsilon\, z(I) \;=\; (1-\varepsilon)\, z(I),$$

e dividendo per $FP^{\varepsilon}(I) > 0$ si ottiene l'enunciato.

Sull'istanza dell'esecuzione di FP^{ε} della prima parte, con $\varepsilon = \frac{1}{2}$, il rapporto è $\frac{47}{38} \approx 1.24$, ben al di sotto del bound $\frac{1}{1-\varepsilon} = 2$. □

Il tempo di calcolo di FP^{ε}

Osservazione 2

L'algoritmo FP^{ε} si può calcolare in tempo $O\left(n^2/\varepsilon\right)$.

La dimostrazione limita il valore ottimo dell'istanza scalata, che fissa la dimensione della tabella della programmazione dinamica.

Dimostrazione (1/2)

Sia $\tilde{q}$ il valore ottimo dell'istanza scalata I'. Ogni soluzione ammissibile di I' è ammissibile per I, quindi

$$\tilde{q} \leq \frac{z(I)}{K} \leq \frac{2\,GR^e(I)}{K} = \frac{2\,GR^e(I)\,n}{\varepsilon\,GR^e(I)} = \frac{2n}{\varepsilon},$$

dove $z(I) \leq 2\,GR^e(I)$ è esattamente la garanzia di prestazione relativa di GR^e.

Poiché $\tilde{q}$ è un intero, $\tilde{q} \leq Q = \lfloor 2n/\varepsilon \rfloor$, e troncare a Q la tabella della programmazione dinamica non cambia il risultato.

Dimostrazione (2/2)

- La tabella ha $n+1$ righe e $Q+1$ colonne, quindi la programmazione dinamica richiede tempo $O(n\,Q) = O(n^2/\varepsilon)$, mentre la scalatura e la chiamata a GR^e richiedono tempo $O(n)$ e $O(n \log n)$.
- Senza il troncamento, la tabella arriverebbe al profitto scalato totale $P' = \sum_j p'_j$.
- Poiché $p_j \leq GR^e(I)$ dà $p'_j \leq \frac{n}{\varepsilon}$, abbiamo solo $P' \leq \frac{n^2}{\varepsilon}$, valore che si può raggiungere, e il tempo di calcolo sarebbe $O(n\,P') = O(n^3/\varepsilon)$.

□

FP^{ε} è un FPTAS

- Dall'ultima proposizione e dall'ultima osservazione segue che la famiglia $\{FP^{\varepsilon}\}_{\varepsilon \in (0,1)}$ è uno **schema di approssimazione** il cui tempo di calcolo è polinomiale sia nella dimensione n dell'istanza sia nell'accuratezza $\frac{1}{\varepsilon}$, cioè un **FPTAS** per il problema dello zaino 0/1.

- La garanzia di GR^e si usa due volte, con due ruoli diversi:
  - $GR^e(I) \leq z(I)$ fissa il fattore di scala;
  - $z(I) \leq 2\, GR^e(I)$ limita la dimensione della tabella della programmazione dinamica.

- Un'euristica migliore al posto di GR^e accorcerebbe la tabella, non migliorerebbe la garanzia.

Riepilogo (1/2)

Rapporti di prestazione nel caso peggiore:

- $r(GR) = +\infty$: nessuna garanzia costante
- $r(GR^e) = 2$, dimostrato con il rilassamento lineare e l'oggetto critico, tight
- $r(H^2) = \frac{4}{3}$ e $r(H^\ell) = \frac{\ell+2}{\ell+1}$, tight, con due casi sul profitto degli oggetti scommessi

Tightness:

- un oggetto piccolo di efficienza 2 e oggetti grandi di efficienza 1 ingannano tutti gli algoritmi del capitolo

Riepilogo (2/2)

Schemi di approssimazione:

- $\{H^\varepsilon\}$, con $\ell(\varepsilon) = \min\{\max\{\lceil \frac{1}{\varepsilon} \rceil - 2, 2\}, n\}$, è un PTAS, ma non un FPTAS: il suo tempo di calcolo $O(n^{\ell(\varepsilon)+1})$ non è polinomiale in $\frac{1}{\varepsilon}$
- $\{FP^\varepsilon\}$ garantisce $\frac{1}{1-\varepsilon}$ in tempo $O(n^2/\varepsilon)$: è un FPTAS

Garanzia di prestazione assoluta

Introduzione: Garanzia di prestazione assoluta

- Finora le garanzie erano relative: un rapporto fra $z(I)$ e $\mathscr{A}(I)$
- Una garanzia assoluta limiterebbe con una costante la differenza $z(I) - \mathscr{A}(I)$
- Per il KP esiste solo un risultato negativo: un algoritmo così sarebbe un algoritmo esatto

Solo un risultato negativo

- Per le garanzie di prestazione assolute del problema dello zaino 0/1 esiste purtroppo solo un risultato negativo.
- Il teorema seguente afferma che, se un algoritmo di approssimazione avesse una garanzia di prestazione assoluta k per un certo k fissato, allora questo algoritmo si potrebbe usare per risolvere all'ottimo qualsiasi istanza del problema dello zaino 0/1.
- Sia $f(n, c)$ una funzione che descrive il tempo di calcolo di un algoritmo per il KP con n oggetti e capacità c.

Teorema 1

Sia $\mathscr{A}$ un algoritmo di approssimazione per il KP con garanzia di prestazione assoluta k e tempo di calcolo in $O\big(f(n,c)\big)$ per ogni istanza con n oggetti e capacità c. Allora $\mathscr{A}$ calcola anche la soluzione ottima di ogni istanza di questo tipo in tempo $O(f(n,c))$.

La dimostrazione moltiplica i profitti per $k+1$:
un errore al più k diventa allora un errore minore di 1.

Dimostrazione (1/3)

Consideriamo un algoritmo di approssimazione $\mathscr{A}$ per il KP con garanzia di prestazione assoluta k e sia I data da

$$p_1, \ldots, p_n, \quad w_1, \ldots, w_n, \quad c.$$

Costruiamo la nuova istanza I' con

$$p'_1, \ldots, p'_n, \quad w'_1, \ldots, w'_n, \quad c'$$

tale che

$$p'_j = (k+1)\, p_j, \quad w'_j = w_j, \quad \forall j \in \mathscr{I}, \quad \text{e} \quad c' = c.$$

- Le due istanze I e I' hanno esattamente lo stesso insieme di soluzioni ammissibili.
- Poiché il valore della funzione obiettivo di una soluzione di I' è esattamente $k+1$ volte il valore della funzione obiettivo della stessa soluzione per I, le due istanze hanno anche le stesse soluzioni ottime.

Dimostrazione (2/3)

Applicando l'algoritmo $\mathscr{A}$ all'istanza I' otteniamo una soluzione di valore $\mathscr{A}(I')$ con

$$z(I') - \mathscr{A}(I') \leq k$$

per la garanzia di prestazione assoluta di $\mathscr{A}$.

Interpretando l'output di $\mathscr{A}$ come soluzione di I otteniamo una soluzione ammissibile di valore $\mathscr{A}(I)$, e

$$\mathscr{A}(I') = (k+1)\,\mathscr{A}(I) \qquad \text{e} \qquad z(I') = (k+1)\,z(I)$$

Sostituendo nella disuguaglianza precedente otteniamo

$$(k+1)\,z(I) - (k+1)\,\mathscr{A}(I) \leq k \iff z(I) - \mathscr{A}(I) \leq \frac{k}{k+1}$$

Dimostrazione (3/3)

Ma poiché tutti i profitti sono interi, anche $z(I) - \mathscr{A}(I)$ deve essere intero, quindi questa disuguaglianza implica

$$z(I) - \mathscr{A}(I) \leq \left\lfloor \frac{k}{k+1} \right\rfloor \Longrightarrow z(I) - \mathscr{A}(I) \leq 0 \Longrightarrow z(I) = \mathscr{A}(I)$$

Poiché la soluzione approssimata di I' è stata calcolata in tempo $O(f(n, c))$, una soluzione ottima di I si può costruire nello stesso tempo. □

Che cosa dice il teorema

- Questo teorema indica che un algoritmo con garanzia di prestazione assoluta fissata per il KP avrebbe lo stesso tempo di calcolo sfavorevole di un algoritmo esatto.

È molto improbabile che esista un algoritmo esatto per il KP con tempo di calcolo polinomiale in n.

- Quindi anche gli algoritmi di approssimazione con garanzia di prestazione assoluta fissata sono destinati ad avere un tempo di calcolo pseudo-polinomiale, che dipende in modo sfavorevole dalla grandezza dei dati.

Riepilogo

Garanzia di prestazione assoluta:

- moltiplicare i profitti per $k+1$ non cambia le soluzioni ammissibili né quelle ottime
- un errore al più k sull'istanza scalata è un errore al più $\frac{k}{k+1} < 1$ su quella originale, quindi 0, poiché i profitti sono interi

Conseguenza:

- un algoritmo con una garanzia assoluta fissata per il KP lo risolve esattamente, nello stesso tempo di calcolo

Errori tipici

Introduzione: Errori tipici

- Raccogliamo gli errori che si commettono più spesso con gli algoritmi di questo capitolo
- Controllateli prima di eseguire a mano un algoritmo, o prima di enunciarne la garanzia

Osservazione 3: errori tipici (1/2)

- ***Fermare l'algoritmo greedy all'oggetto critico.*** *GR salta l'oggetto critico s ma continua a scorrere gli oggetti, e può selezionare oggetti che seguono s (come nella prima esecuzione di GR).*
- ***Confondere i_m e s.*** *In GR^e l'oggetto i_m è l'oggetto di profitto massimo $p_{\max}$, non l'oggetto critico s: i due coincidono solo per caso.*
- ***Dimenticare di ordinare per efficienza.*** *Tutti gli algoritmi di questo capitolo suppongono gli oggetti ordinati per $\frac{p_j}{w_j}$ non crescente; l'oggetto critico e la soluzione greedy dipendono dall'ordine.*

Osservazione: errori tipici (2/2)

- ***Chiamare H^ℓ un FPTAS.*** *Il suo tempo di calcolo $O(n^{\ell+1})$ ha l'esponente $\ell(\varepsilon) = \lceil \frac{1}{\varepsilon} \rceil - 2$ che cresce con $\frac{1}{\varepsilon}$: H^ε è un PTAS, non un FPTAS.*
- ***Scalare i pesi.*** *L'algoritmo Basic-FPTAS arrotonda per difetto solo i profitti; i pesi e la capacità restano invariati, così che le soluzioni ammissibili sono le stesse.*
- ***Contare l'intera tabella della programmazione dinamica.*** *Il bound $O(n^2/\varepsilon)$ vale perché la tabella è troncata a $\lfloor 2n/\varepsilon \rfloor$; il profitto scalato totale può arrivare a $\frac{n^2}{\varepsilon}$.*

Riepilogo

Quando si eseguono gli algoritmi:

- ordinare gli oggetti per efficienza; GR continua a scorrere dopo l'oggetto critico; i_m non è s

Quando si enunciano le garanzie:

- H^{ε} è un PTAS, non un FPTAS
- il Basic-FPTAS scala i profitti, mai i pesi, e tronca la tabella a $\lfloor 2n/\varepsilon \rfloor$

Implementazione in C++

Introduzione: Implementazione in C++

- Gli algoritmi del capitolo sono implementati in C++, nella cartella `KP_HEUR_CODE/src`
- Guardiamo brevi estratti dei codici, e seguiamo passo per passo il codice di GR e di GR^e su due istanze del capitolo
- In questa sezione gli oggetti sono indicizzati a partire da 0. La logica dell'algoritmo non cambia.

L'algoritmo greedy in C++

Un estratto di `GreedyKP`, l'implementazione in C++ dell'**algoritmo greedy**:

```
1  for (int i = 0; i < n; ++i)
2  {
3      x[i] = 0; // item i is initially not selected
4  }
5  int rc = c; // residual capacity
6
7  for (int i = 0; i < n; ++i)
8  {
9      if (w[i] <= rc)
10     {
11         x[i] = 1; // select item i
12         rc = rc - w[i]; // update residual capacity
13     }
14 }
```

Esecuzione di *GR* in C++ (1/3)

Questa è l'esecuzione passo per passo dell'implementazione in C++ di *GR* sull'istanza del KP dell'esempio dell'oggetto critico ($c = 9$, $n = 7$).

Inizializzazione

$x[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
	-	-	-	-	-	-	-

$rc = 9$

Iterazione 1 – L'oggetto 0 viene selezionato

$x[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
	1	-	-	-	-	-	-

$rc = 7$

Iterazione 2 – L'oggetto 1 viene selezionato

$x[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
	1	1	-	-	-	-	-

$rc = 4$

Esecuzione di *GR* in C++ (2/3)

Iterazione 3 – L'oggetto 2 non viene selezionato

$x[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
	1	1	0	-	-	-	-

$rc = 4$

Iterazione 4 – L'oggetto 3 non viene selezionato

$x[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
	1	1	0	0	-	-	-

$rc = 4$

Iterazione 5 – L'oggetto 4 non viene selezionato

$x[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
	1	1	0	0	0	-	-

$rc = 4$

Esecuzione di *GR* in C++ (3/3)

Iterazione 6 – L'oggetto 5 non viene selezionato

$x[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
	1	1	0	0	0	0	-

$rc = 4$

Iterazione 7 – L'oggetto 6 viene selezionato

$x[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
	1	1	0	0	0	0	1

$rc = 0$

Il valore calcolato è

$$z = 14$$

Esecuzione di *GR* in C++ (1/2)

Questa è l'esecuzione passo per passo dell'implementazione in C++ di *GR* sull'istanza del KP con $c = 10$ e 4 oggetti.

Inizializzazione

$x[i]$	$i = 0$	$i = 1$	$i = 2$	$i = 3$
	-	-	-	-

$rc = 10$

Iterazione 1 – L'oggetto 0 viene selezionato

$x[i]$	$i = 0$	$i = 1$	$i = 2$	$i = 3$
	1	-	-	-

$rc = 8$

Iterazione 2 – L'oggetto 1 viene selezionato

$x[i]$	$i = 0$	$i = 1$	$i = 2$	$i = 3$
	1	1	-	-

$rc = 5$

Esecuzione di *GR* in C++ (2/2)

Iterazione 3 – L'oggetto 2 non viene selezionato

$x[i]$	$i=0$	$i=1$	$i=2$	$i=3$
	1	1	0	-

$rc = 5$

Iterazione 4 – L'oggetto 3 viene selezionato

$x[i]$	$i=0$	$i=1$	$i=2$	$i=3$
	1	1	0	1

$rc = 0$

Il valore calcolato è

$$z = 18$$

L'algoritmo greedy esteso in C++

Un estratto di `ExtendedGreedyKP`, l'implementazione in C++ dell'**algoritmo greedy esteso**: dopo la chiamata a `GreedyKP`, il miglior oggetto singolo.

```
1  int im = 0; // best single item
2  int pm = p[0];
3  for (int i = 1; i < n; ++i)
4  {
5      if (p[i] > pm)
6      {
7          im = i; // update best single item
8          pm = p[i]; // update its profit
9      }
10 }
```

Esecuzione di GR^e in C++

Questa è l'esecuzione passo per passo dell'implementazione in C++ di GR^e sull'istanza del KP dell'esempio dell'oggetto critico. La soluzione calcolata da GR è

$x[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
	1	1	0	0	0	0	1

- Il valore calcolato è $z = 14$.
- Il profitto massimo è $p_{\max} = 9$, e l'oggetto di profitto massimo è (indici a partire da 0) $i_m = 3$.
- Poiché $14 \geq 9$, GR^e mantiene la soluzione greedy.

Esecuzione di GR^e in C++

Questa è l'esecuzione passo per passo dell'implementazione in C++ di GR^e sull'istanza del KP con $c = 10$ e 4 oggetti. La soluzione calcolata da GR è

$x[i]$	$i = 0$	$i = 1$	$i = 2$	$i = 3$
	1	1	0	1

- Il valore calcolato è $z = 18$; il profitto massimo è $p_{\max} = 19$, e l'oggetto di profitto massimo è (indici a partire da 0) $i_m = 2$.
- Poiché $18 < 19$, GR^e sostituisce la soluzione greedy con la migliore soluzione ammissibile con un solo oggetto:

$x[i]$	$i = 0$	$i = 1$	$i = 2$	$i = 3$
	0	0	1	0

Il valore aggiornato è $z = 19$.

L'algoritmo H^2 in C++

Un estratto di `H2KP`: chiama `ExtendedGreedyKP` sull'istanza residua di ogni coppia ammissibile, esattamente come lo pseudocodice, e tiene la migliore soluzione trovata.

```
1  if (w[i] + w[k] > c) continue; // (i,k) is not in P
2  int cbar = c - w[i] - w[k]; // residual capacity
3  int pmin = (p[i] < p[k] ? p[i] : p[k]);
4  int m = 0; // residual set I(i,k)
5  for (int j = 0; j < n; ++j)
6      if (j != i && j != k && p[j] <= pmin
7          && w[j] <= cbar)
8      { idx[m] = j; pr[m] = p[j]; wr[m] = w[j]; m++; }
9  int zres = (m > 0) ?
10     ExtendedGreedyKP(m, cbar, wr, pr, xr) : 0;
11 int lbik = p[i] + p[k] + zres;
```

L'algoritmo H^2 in C++: osservazioni

- Sull'istanza del KP dell'esempio di H^2, con $n = 7$ oggetti e capacità $c = 12$, la funzione restituisce $lb = 20$ con $x = (1, 1, 0, 1, 0, 0, 0)$, cioè la soluzione $S = \{1, 2, 4\}$ già trovata in quell'esempio, ottima per quell'istanza.

- I due cicli annidati su i e k eseguono $O(n^2)$ iterazioni, e ogni iterazione costruisce l'istanza residua ed esegue `ExtendedGreedyKP` in tempo $O(n)$: è il tempo di calcolo $O(n^3)$ della proposizione sul tempo di calcolo di H^2.

L'algoritmo H^ℓ e la PD sui profitti in C++

- `HlKP` implementa l'**algoritmo** H^ℓ. La funzione `NextSubset` enumera i sottoinsiemi di esattamente r oggetti in ordine lessicografico; la preelaborazione esamina i sottoinsiemi di al più $\ell - 1$ oggetti, e ogni sottoinsieme di esattamente ℓ oggetti viene completato da `ExtendedGreedyKP` sulla sua istanza residua, come nello pseudocodice.

- `DPProfits` implementa la **programmazione dinamica sui profitti**, il secondo algoritmo di PD del capitolo «PD per il problema dello zaino 0/1» del modulo *Programmazione dinamica*, usato dall'algoritmo Basic-FPTAS.
  - Con gli indici a partire da 0 del codice, la cella `u[i][q]` contiene il peso totale minimo di un sottoinsieme dei primi i oggetti di profitto totale esattamente q, e una soluzione ottima viene ricostruita all'indietro.
  - Il parametro `qmax` tronca la tabella: si calcolano solo i profitti $q \leq \min\{P, \texttt{qmax}\}$, dove P è il profitto totale.

L'algoritmo Basic-FPTAS in C++

Un estratto di `BasicFPTAS`. L'accuratezza ε è passata come frazione $\frac{\texttt{eps_num}}{\texttt{eps_den}}$, così che i profitti scalati $\left\lfloor \frac{p_j}{K} \right\rfloor = \left\lfloor \frac{p_j \, n \, \texttt{eps_den}}{\texttt{eps_num} \, GR^e(I)} \right\rfloor$ sono calcolati in aritmetica intera esatta, senza errori di arrotondamento.

```
int lb = ExtendedGreedyKP(n, c, w, p, x);
int* ps = new int[n]; // scaled profits p'
for (int j = 0; j < n; ++j)
    ps[j] = (int)(((long long)p[j] * n * eps_den)
                  / ((long long)eps_num * lb));
int qmax = (int)((2LL * n * eps_den) / eps_num);
DPProfits(n, c, w, ps, qmax, x); // scaled instance
```

La tabella di `DPProfits` è troncata a $\lfloor 2n/\varepsilon \rfloor$.

Riepilogo

I codici:

- `GreedyKP`, `ExtendedGreedyKP`, `H2KP`, `H1KP`, `DPProfits` e `BasicFPTAS` seguono gli pseudocodici, con gli indici a partire da 0
- le esecuzioni passo per passo di GR e GR^e danno i valori 14 e 18, e GR^e dà 19 sulla seconda istanza

Test:

- tutte le funzioni sono nella cartella `KP_HEUR_CODE/src`; il comando `make test` le compila ed esegue ogni esempio di questo capitolo e ogni istanza degli esercizi, confrontando il valore trovato con quello atteso

In sintesi

Introduzione: In sintesi

- Raccogliamo le idee chiave del capitolo sugli algoritmi di approssimazione per il KP
- Il quadro riassuntivo raccoglie i bound trovati finora sull'istanza dell'esempio dell'oggetto critico

Riepilogo (1/2)

- GR non ha una garanzia costante, GR^e (il migliore fra GR e il miglior oggetto singolo) ha $r(GR^e) = 2$, H^2 ha $r(H^2) = \frac{4}{3}$ e H^ℓ ha $r(H^\ell) = \frac{\ell+2}{\ell+1}$.

- $\{H^\varepsilon\}$ è un PTAS con tempo di calcolo $O(n^{\ell+1})$; l'algoritmo Basic-FPTAS arrotonda per difetto i profitti, risolve esattamente l'istanza scalata con la programmazione dinamica ed è un FPTAS: garantisce $\frac{1}{1-\varepsilon}$ in tempo $O(n^2/\varepsilon)$.

- Non esiste un algoritmo con garanzia di prestazione assoluta fissata, a meno che non risolva esattamente il KP.

Riepilogo (2/2)

Quadro riassuntivo dell'istanza del KP dell'esempio dell'oggetto critico ($c = 9$, $p = (6, 5, 8, 9, 6, 7, 3)$, $w = (2, 3, 6, 7, 5, 9, 4)$). I bound su $z(I)$ trovati finora:

Fonte	Lower bound	Upper bound	Da
Bound di Dantzig	–	16	$\lfloor \frac{49}{3} \rfloor$, con $s = 3$ e $rc = 4$
Questo capitolo, GR e GR^e	14	–	$S = \{1, 2, 7\}$
Questo capitolo, H^2 e H^3	15	–	$S = \{1, 4\}$
Questo capitolo, FP^ε con $\varepsilon = \frac{1}{2}$	15	–	$K = 1$, l'istanza scalata è quella originale
Bound migliori	15	16	$\mathrm{UB} - \mathrm{LB} = 1$; il valore ottimo è 15