

Approssimazione per $P \parallel C_{\max}$

Indice

Perché garanzie per lo scheduling?

Notazione

Gli algoritmi *LIST* e *LPT*

List scheduling

Longest processing time first

Una proprietà del list scheduling

Garanzie di prestazione

Il rapporto di prestazione nel caso peggiore di LIST

Il rapporto di prestazione nel caso peggiore di LPT

Fonti

Errori tipici

In sintesi

Obiettivi

Alla fine di questo capitolo saremo in grado di:

- ▶ eseguire a mano *LIST* e *LPT* e disegnare lo schedule che ne risulta;
- ▶ dimostrare le garanzie $2 - \frac{1}{m}$ di *LIST* e $\frac{4}{3} - \frac{1}{3m}$ di *LPT*;
- ▶ costruire le famiglie di istanze che rendono tight queste garanzie;
- ▶ spiegare perché ordinare i lavori per tempo di esecuzione non crescente migliora la garanzia.

Prerequisiti

- ▶ Da questo modulo, il capitolo «[Principi degli algoritmi di approssimazione](#)»: rapporto di prestazione, garanzia di prestazione relativa, garanzia tight e rapporto di prestazione nel caso peggiore.
- ▶ Da questo modulo, il capitolo «[Problemi di scheduling](#)»: la definizione e l'istanza di riferimento di $P \parallel C_{\max}$, e il suo lower bound elementare.
- ▶ Da questo modulo, il capitolo «[Euristiche costruttive](#)»: gli algoritmi *LIST* e *LPT* come regole costruttive.

Perché garanzie per lo scheduling?

Introduzione: Perché garanzie per lo scheduling?

- ▶ Una semplice regola greedy assegna i lavori uno alla volta alla macchina meno carica
- ▶ Il makespan che trova dipende dall'**ordine** in cui si esaminano i lavori
- ▶ Vogliamo sapere quanto può andare male, non su un'istanza ma su *ogni* istanza

Una sfida

Sette lavori con tempi di esecuzione $(5, 5, 4, 4, 3, 3, 3)$ vanno assegnati a **tre macchine** in modo da minimizzare il makespan.

Assegnate i lavori uno alla volta a una macchina di carico minimo.

Provate l'ordine $7, 6, 5, 4, 3, 2, 1$ e l'ordine $1, 2, \dots, 7$.

La risposta

- ▶ Il primo ordine dà makespan 12, il secondo 11, e il valore ottimo è 9.
- ▶ Quanto può essere cattivo il primo ordine su altre istanze?
- ▶ I risultati positivi di questo capitolo limitano lo scarto su *ogni* istanza.

La sfida:

- ▶ la stessa regola greedy, due ordini dei lavori: makespan 12 e 11, contro l'ottimo 9

L'obiettivo del capitolo:

- ▶ limitare lo scarto fra il makespan greedy e l'ottimo su ogni istanza
- ▶ mostrare che i limiti non si possono migliorare

Notazione

Introduzione: Notazione

- ▶ Fissiamo la notazione del problema di scheduling: macchine, lavori, carichi e makespan
- ▶ Richiamiamo il rapporto di prestazione di un algoritmo euristico
- ▶ Richiamiamo il lower bound elementare sul valore ottimo, lo strumento principale di tutte le dimostrazioni

Macchine, lavori e carichi

- ▶ Il problema di questo capitolo è il **problema di scheduling su macchine parallele identiche** ($P \parallel C_{\max}$): la sua definizione e le sue ipotesi sono nella sua pagina.
- ▶ Ci sono m macchine identiche e un insieme $\mathcal{J} = \{1, 2, \dots, n\}$ di lavori con tempi di esecuzione t_j .
- ▶ Una soluzione assegna ogni lavoro $j \in \mathcal{J}$ a una macchina $mach_j = h$.
- ▶ L'insieme dei lavori della macchina h è J_h , e il suo **carico** è

$$L_h = \sum_{j \in J_h} t_j.$$

- ▶ Il **makespan** è $C_{\max} = \max_h L_h$.

Richiamo: il rapporto di prestazione

- ▶ Come nel capitolo «[Principi degli algoritmi di approssimazione](#)», $z(I)$ è il valore ottimo dell'istanza I e $\mathcal{A}(I)$ il valore della soluzione restituita dall'algoritmo $\mathcal{A}$.
- ▶ $P \parallel C_{\max}$ è un problema di minimo, quindi per ogni istanza I

$$\mathcal{A}(I) \geq z(I) \quad \text{e} \quad \frac{\mathcal{A}(I)}{z(I)} \geq 1.$$

- ▶ Il rapporto di prestazione nel caso peggiore è

$$r(\mathcal{A}) = \sup_I \frac{\mathcal{A}(I)}{z(I)}.$$

Il lower bound elementare

- ▶ Usiamo il lower bound elementare delle note sui problemi di ottimizzazione combinatoria:

$$z(I) \geq \max \left\{ \max_{j \in \mathcal{J}} t_j, \frac{\sum_{j \in \mathcal{J}} t_j}{m} \right\}. \quad (1)$$

- ▶ Il lavoro più lungo deve essere eseguito da qualche macchina.
- ▶ Il carico totale è ripartito fra m macchine, quindi qualche macchina ha almeno il carico medio.

L'istanza di riferimento

- L'istanza di riferimento ha

$$m = 3 \quad \text{e} \quad t = (5, 5, 4, 4, 3, 3, 3).$$

- Il bound (1) dà

$$\max \left\{ 5, \frac{27}{3} \right\} = 9.$$

- Il valore ottimo è $z(I) = 9$, con

$$J_1 = \{1, 3\}, \quad J_2 = \{2, 4\}, \quad J_3 = \{5, 6, 7\}.$$

Notazione:

- ▶ m macchine, lavori $\mathcal{J} = \{1, 2, \dots, n\}$, carichi L_h , makespan $C_{\max} = \max_h L_h$
- ▶ $\mathcal{A}(I) \geq z(I)$ e $r(\mathcal{A}) = \sup_I \mathcal{A}(I)/z(I)$

Lower bound:

- ▶ $z(I)$ è almeno il lavoro più lungo e almeno il carico medio
- ▶ sull'istanza di riferimento il bound dà $9 = z(I)$

Gli algoritmi *LIST* e *LPT*

Introduzione: Gli algoritmi *LIST* e *LPT*

- ▶ Presentiamo le due euristiche costruttive del capitolo: *LIST* e *LPT*
- ▶ *LIST* segue una lista qualsiasi dei lavori; *LPT* prima ordina i lavori
- ▶ Eseguiamo entrambi sull'istanza di riferimento e disegniamo i loro schedule

List scheduling

- ▶ Il *List Scheduling Algorithm* (indicato con *LIST*) è un'euristica costruttiva per $P \parallel C_{\max}$.
- ▶ Esamina i lavori nell'ordine di una lista λ data.
- ▶ Assegna ogni lavoro a una macchina di carico corrente minimo (in caso di pareggio, alla macchina di indice minimo).

Algorithm: List Scheduling (1/2)

Input : istanza di $P \parallel C_{\max}$ con n lavori, tempi $t[1..n]$,
 m macchine; lista $\lambda[1..n]$

Output: assegnamento lavori-macchine $mach[1..n]$ e
makespan $C_{\max}$

```
1 function ListScheduling( $n, m, t, \lambda$ )
2    $mach :=$  nuovo vettore[1.. $n$ ]      // assegnamento
   lavori-macchine
3   for  $h := 1$  to  $m$  do
4      $L[h] \leftarrow 0$                 // carichi delle macchine
5   end
```

Algorithm: List Scheduling (2/2)

function *ListScheduling*(n, m, t, λ) (continua)

```
6   for  $k := 1$  to  $n$  do
7        $j := \lambda[k]$     //  $k$ -esimo lavoro della lista
8        $s :=$  una macchina con  $L[s]$  minimo (indice
           minimo)          // macchina meno carica
9        $mach[j] \leftarrow s$  // assegnare il lavoro  $j$  alla
           macchina  $s$ 
10       $L[s] \leftarrow L[s] + t[j]$ 
11  end
12  return  $mach, C_{\max} = \max_h L[h]$ 
13 end
```

Come funziona *LIST*

- ▶ *LIST* è lo stesso algoritmo delle note sulle euristiche costruttive, dove è introdotto come regola greedy.
- ▶ Qui ci interessa il suo **caso peggiore**: quanto può essere cattiva una lista?
- ▶ Ogni macchina esegue i suoi lavori senza tempi morti, quindi il lavoro j assegnato all'iterazione k **inizia** all'istante S_j , pari al carico della sua macchina subito prima dell'assegnamento, e termina in $S_j + t_j$.

Eseguiamo *LIST* sull'istanza di riferimento
con **prima i lavori più brevi.**

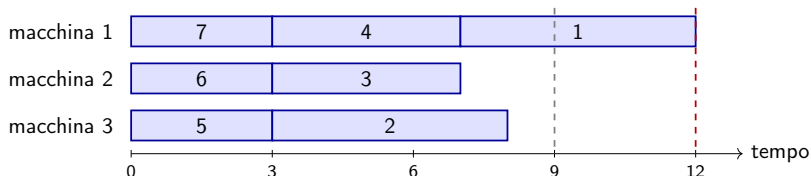
Esempio (1/2)

Consideriamo la lista $\lambda = (7, 6, 5, 4, 3, 2, 1)$. I carichi prima di ogni assegnamento e la macchina scelta sono:

lavoro j	t_j	carichi (L_1, L_2, L_3) prima	macchina
7	3	(0, 0, 0)	1
6	3	(3, 0, 0)	2
5	3	(3, 3, 0)	3
4	4	(3, 3, 3)	1
3	4	(7, 3, 3)	2
2	5	(7, 7, 3)	3
1	5	(7, 7, 8)	1

Esempio (2/2)

I carichi finali sono $(12, 7, 8)$, con $J_1 = \{7, 4, 1\}$, $J_2 = \{6, 3\}$, $J_3 = \{5, 2\}$, e $LIST(I) = 12$:



L'ultimo lavoro, il lavoro 1, inizia in $S_1 = 7$, il carico minimo in quel momento: tutte le macchine sono occupate almeno fino all'istante 7. È l'osservazione su cui si costruisce la garanzia di $LIST$.

Longest processing time first

- ▶ Il *Longest Processing Time first Algorithm* (indicato con *LPT*) è un'euristica costruttiva per $P \parallel C_{\max}$.
- ▶ Ordina i lavori per tempo di esecuzione non crescente e applica *LIST* a questa lista.
- ▶ Ogni lavoro, dal più lungo al più breve, viene assegnato a una macchina meno carica.

Algorithm: Longest Processing Time first

Input : istanza di $P || C_{\max}$ con n lavori, tempi $t[1..n]$,
 m macchine

Output: assegnamento lavori-macchine $mach[1..n]$ e
makespan $C_{\max}$

```
1 function  $LPT(n, m, t)$   
2    $\lambda :=$  i lavori ordinati per  $t[j]$  non crescente (pareggi  
   per indice) // prima i lavori più lunghi  
3   return  $ListScheduling(n, m, t, \lambda)$   
4 end
```

- Nelle dimostrazioni conviene rinumerare i lavori nell'ordine di λ , così che $t_1 \geq t_2 \geq \dots \geq t_n$ e LPT assegna i lavori nell'ordine $1, 2, \dots, n$.

Eseguiamo *LPT* sulla stessa istanza di riferimento:
ora **prima i lavori più lunghi**.

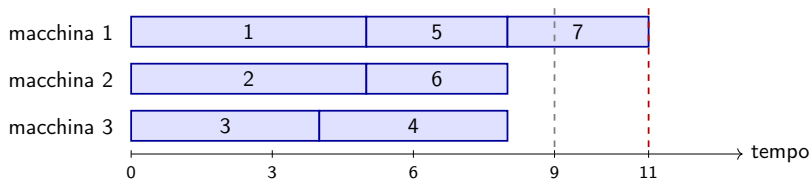
Esempio (1/2)

I lavori sono già ordinati, $\lambda = (1, 2, 3, 4, 5, 6, 7)$.

- ▶ I lavori 1, 2, 3 vanno sulle tre macchine vuote, con carichi $(5, 5, 4)$.
- ▶ Il lavoro 4 va sulla macchina 3, la meno carica: $(5, 5, 8)$.
- ▶ Il lavoro 5 va sulla macchina 1: $(8, 5, 8)$.
- ▶ Il lavoro 6 va sulla macchina 2: $(8, 8, 8)$.
- ▶ Il lavoro 7 trova tre macchine di carico 8 e va sulla macchina 1: $(11, 8, 8)$.

Esempio (2/2)

Quindi $J_1 = \{1, 5, 7\}$, $J_2 = \{2, 6\}$, $J_3 = \{3, 4\}$ e $LPT(I) = 11$:



Con $z(I) = 9$, il rapporto è $\frac{11}{9}$: l'ordinamento migliora il makespan da 12 a 11, ma l'ultimo lavoro, per quanto breve, sbilancia tre macchine perfettamente bilanciate.

Riepilogo

LIST:

- ▶ ogni lavoro della lista va su una macchina meno carica (di indice minimo in caso di pareggio)
- ▶ il lavoro j inizia in S_j , il carico della sua macchina subito prima dell'assegnamento
- ▶ istanza di riferimento, prima i lavori più brevi: $LIST(I) = 12$

LPT:

- ▶ *LIST* applicato ai lavori ordinati per tempo di esecuzione non crescente
- ▶ istanza di riferimento: $LPT(I) = 11$, contro $z(I) = 9$

Una proprietà del list scheduling

Introduzione: Una proprietà del list scheduling

- ▶ Le garanzie della prossima sezione confrontano il valore di una soluzione euristica con il lower bound (1) su $z(I)$.
- ▶ Oltre al bound, usano una proprietà degli schedule costruiti da *LIST*: il lavoro che termina per ultimo non può iniziare troppo tardi.

L'inizio dell'ultimo lavoro

Osservazione 1

Sia ℓ un lavoro che termina per ultimo in uno schedule costruito da LIST (con una lista qualsiasi). Allora il suo istante di inizio soddisfa

$$S_{\ell} \leq \frac{1}{m} \sum_{j \in \mathcal{J} \setminus \{\ell\}} t_j.$$

Dimostrazione

- ▶ Quando ℓ viene assegnato, S_ℓ è il minimo degli m carichi, quindi non supera la loro media.
- ▶ I carichi in quel momento contengono solo lavori assegnati prima di ℓ , quindi la loro somma è al più $\sum_{j \neq \ell} t_j$.
- ▶ Quindi

$$S_\ell \leq \frac{1}{m} \sum_{j \neq \ell} t_j.$$



L'inizio dell'ultimo lavoro:

- ▶ con una lista qualsiasi, il lavoro ℓ che termina per ultimo inizia non più tardi del carico medio degli altri lavori
- ▶ motivo: S_ℓ è il carico minimo, quindi al più il carico medio

Che cosa viene dopo:

- ▶ questa proprietà e il lower bound (1) danno le garanzie di *LIST* e *LPT*

Garanzie di prestazione

Introduzione: Garanzie di prestazione

- ▶ Dimostriamo la garanzia $2 - \frac{1}{m}$ di *LIST*, per ogni lista
- ▶ Dimostriamo la garanzia migliore $\frac{4}{3} - \frac{1}{3m}$ di *LPT*
- ▶ Per ogni garanzia costruiamo una famiglia di istanze su cui viene raggiunta: entrambe le garanzie sono **tight**

Proposizione 1

Per $P \parallel C_{\max}$, per ogni istanza I e ogni lista λ ,

$$\frac{LIST(I)}{z(I)} \leq 2 - \frac{1}{m}.$$

- Idea: limitare l'inizio dell'ultimo lavoro con l'osservazione della sezione precedente, e la sua durata con il lower bound (1).

Dimostrazione

Sia ℓ un lavoro che termina per ultimo, così che $LIST(I) = S_\ell + t_\ell$.
Per l'osservazione sull'inizio dell'ultimo lavoro,

$$\begin{aligned} LIST(I) &\leq \frac{1}{m} \sum_{j \neq \ell} t_j + t_\ell \\ &= \underbrace{\frac{1}{m} \sum_{j \in \mathcal{J}} t_j}_{\leq z(I)} + \left(1 - \frac{1}{m}\right) \underbrace{t_\ell}_{\leq z(I)} \\ &\leq \left(2 - \frac{1}{m}\right) z(I), \end{aligned}$$

dove entrambi i bound vengono dalla (1). Dividendo per $z(I)$ si ottiene la tesi.



Una lista può essere così cattiva
da raggiungere il rapporto $2 - \frac{1}{m}$?

La garanzia di *LIST* è tight

Proposizione 2

Per ogni $m \geq 2$ esistono, per $P \parallel C_{\max}$, un'istanza I_m con m macchine e una lista λ tali che

$$\frac{LIST(I_m)}{z(I_m)} = 2 - \frac{1}{m}.$$

Dimostrazione (1/3)

- ▶ I_m ha $n = m(m - 1) + 1$ lavori:
 - ▶ i lavori $1, \dots, m(m - 1)$ con $t_j = 1$;
 - ▶ il lavoro n con $t_n = m$.
- ▶ La lista è $\lambda = (1, 2, \dots, n)$.
- ▶ Abbiamo $n > m$ per $m \geq 2$.

Dimostrazione (2/3)

- ▶ I lavori unitari vengono assegnati a turni di m : in ogni turno tutte le macchine hanno lo stesso carico, i lavori vanno sulle macchine $1, 2, \dots, m$ in quest'ordine e ogni carico cresce di 1.
- ▶ Dopo gli $m - 1$ turni tutti i carichi valgono $m - 1$, e il lavoro n inizia in $m - 1$ sulla macchina 1:

$$LIST(I_m) = (m - 1) + m = 2m - 1.$$

Dimostrazione (3/3)

- ▶ Mettendo il lavoro n da solo sulla macchina 1 e gli $m(m-1)$ lavori unitari, m per macchina, sulle macchine $2, \dots, m$ si ottiene makespan m .
- ▶ La (1) dà $z(I_m) \geq t_n = m$.

- ▶ Quindi $z(I_m) = m$ e

$$\frac{LIST(I_m)}{z(I_m)} = \frac{2m-1}{m} = 2 - \frac{1}{m}.$$



Il rapporto di prestazione nel caso peggiore di *LIST*

- ▶ Dalle due proposizioni precedenti segue che *LIST* è un algoritmo di $(2 - \frac{1}{m})$ -approssimazione, qualunque sia la lista.
- ▶ Il rapporto di prestazione nel caso peggiore di *LIST* su m macchine è

$$r(LIST) = 2 - \frac{1}{m}$$

Osservazioni su *LIST*

- ▶ Per $m = 3$ la famiglia ha $t = (1, 1, 1, 1, 1, 1, 3)$, $LIST(I_3) = 5$ e $z(I_3) = 3$.
- ▶ Il lavoro critico è quello lungo, messo per ultimo: *LPT* lo metterebbe per primo e troverebbe l'ottimo.
- ▶ Nell'esempio di *LIST* con prima i lavori più brevi,

$$LIST(I) = 12 \leq \frac{5}{3} \cdot 9 = 15.$$

- ▶ Il bound e la famiglia sono dovuti a Graham (1966).

Ordinare i lavori mette per primi i lavori lunghi.
Quanto migliora la garanzia?

Proposizione 3

Per $P \parallel C_{\max}$, per ogni istanza I ,

$$\frac{LPT(I)}{z(I)} \leq \frac{4}{3} - \frac{1}{3m}.$$

- Idea: tenere solo i lavori fino all'ultimo, poi distinguere un ultimo lavoro breve (l'argomento di *LIST*) da un ultimo lavoro lungo (al più due lavori per macchina).

Dimostrazione (1/5)

- ▶ Numeriamo i lavori in modo che $t_1 \geq t_2 \geq \dots \geq t_n$, l'ordine in cui *LPT* li assegna.
- ▶ Sia ℓ un lavoro che termina per ultimo, così che $LPT(I) = S_\ell + t_\ell$.
- ▶ I lavori dopo ℓ non hanno alcun ruolo: sia I' l'istanza con i soli lavori $1, \dots, \ell$.
- ▶ *LPT* li assegna su I' esattamente come su I , quindi $LPT(I') = LPT(I)$.
- ▶ $z(I') \leq z(I)$, poiché uno schedule di I ristretto ai lavori di I' è uno schedule di I' .

Dimostrazione (2/5)

- ▶ Basta quindi dimostrare $LPT(I') \leq \left(\frac{4}{3} - \frac{1}{3m}\right) z(I')$.
- ▶ In I' il lavoro ℓ è il più breve. Distinguiamo due casi.

Caso 1: $t_\ell \leq \frac{z(I')}{3}$. Come nella dimostrazione per *LIST*, con l'osservazione sull'inizio dell'ultimo lavoro applicata a I' ,

$$\begin{aligned} LPT(I') &\leq \frac{1}{m} \sum_{j=1}^{\ell} t_j + \left(1 - \frac{1}{m}\right) t_\ell \\ &\leq z(I') + \left(1 - \frac{1}{m}\right) \frac{z(I')}{3} \\ &= \left(\frac{4}{3} - \frac{1}{3m}\right) z(I'). \end{aligned}$$

Dimostrazione (3/5)

Caso 2: $t_\ell > \frac{z(I')}{3}$.

- ▶ Ogni lavoro di I' è più lungo di $\frac{z(I')}{3}$, quindi in uno schedule ottimo di I' nessuna macchina ha tre lavori, e $\ell \leq 2m$.
- ▶ Se $\ell \leq m$, il lavoro ℓ inizia su una macchina vuota e $LPT(I') = t_\ell \leq z(I')$.
- ▶ Altrimenti $\ell = m + r$ con $1 \leq r \leq m$; poniamo $k := m - r + 1$, così che $1 \leq k \leq m$.
- ▶ Mostriamo che

$$z(I') \geq t_k + t_\ell \geq LPT(I').$$

Dimostrazione (4/5)

Lower bound. Consideriamo uno schedule ottimo di I' .

- ▶ Se due dei lavori $1, \dots, k$ condividono una macchina, il suo carico è almeno $2t_k \geq t_k + t_\ell$.
- ▶ Altrimenti stanno su k macchine diverse; le altre $m - k = r - 1$ macchine ricevono al più $2(r - 1)$ lavori, mentre i lavori $k + 1, \dots, \ell$ sono $\ell - k = 2r - 1$.
- ▶ Quindi uno di essi condivide una macchina con uno dei lavori $1, \dots, k$, e quel carico è almeno $t_k + t_\ell$.
- ▶ In entrambi i casi $z(I') \geq t_k + t_\ell$.

Dimostrazione (5/5)

Upper bound.

- ▶ I primi m lavori vanno su m macchine diverse, poiché una macchina vuota ha carico 0 e ogni lavoro ha durata positiva.
- ▶ Prima del lavoro ℓ , le m macchine contengono $\ell - 1 = m + r - 1$ lavori, almeno uno per macchina, quindi al più $r - 1$ macchine contengono due o più lavori e almeno $m - r + 1 = k$ macchine contengono esattamente un lavoro, uno dei lavori $1, \dots, m$.
- ▶ Sono k lavori diversi fra $1, \dots, m$, quindi il massimo indice fra di essi è almeno k , e la sua durata è al più t_k .
- ▶ Quindi $S_\ell \leq t_k$, e $LPT(I') = S_\ell + t_\ell \leq t_k + t_\ell$.

Pertanto nel Caso 2 vale $LPT(I') \leq z(I')$, risultato ancora migliore della tesi. □

La garanzia di LPT è tight?
Costruiamo una famiglia di istanze che la raggiunge.

La garanzia di *LPT* è tight

Proposizione 4

Per ogni $m \geq 2$ esiste, per $P \parallel C_{\max}$, un'istanza I_m con m macchine tale che

$$\frac{LPT(I_m)}{z(I_m)} = \frac{4}{3} - \frac{1}{3m}.$$

Dimostrazione (1/5)

- ▶ I_m ha $n = 2m + 1$ lavori, già ordinati: due lavori per ciascuna durata $2m - 1, 2m - 2, \dots, m + 1$, e tre lavori di durata m ,
 $t = (2m - 1, 2m - 1, 2m - 2, 2m - 2, \dots, m + 1, m + 1, m, m, m)$.
- ▶ Per $j \leq 2m$ la durata è

$$t_j = 2m - \lceil j/2 \rceil,$$

$$\text{e } t_{2m+1} = m.$$

Dimostrazione (2/5)

Ottimo.

- ▶ La durata totale è

$$2 \sum_{v=m+1}^{2m-1} v + 3m = 3m^2,$$

quindi la (1) dà $z(I_m) \geq 3m$.

- ▶ Lo schedule che accoppia un lavoro di durata v con un lavoro di durata $3m - v$, per $v = m + 1, \dots, 2m - 1$ (entrambe le durate stanno in $\{m + 1, \dots, 2m - 1\}$, e ogni durata si usa due volte), riempie $m - 1$ macchine con carico $3m$.
- ▶ Mette i tre lavori di durata m sull'ultima macchina, di nuovo con carico $3m$.
- ▶ Quindi $z(I_m) = 3m$.

Dimostrazione (3/5)

LPT.

- ▶ I lavori $1, \dots, m$ vanno sulle macchine $1, \dots, m$.
- ▶ Poi il lavoro $m + 1$ va su una macchina con il lavoro singolo più breve, t_m , e il suo carico diventa $t_m + t_{m+1}$.
- ▶ In generale, affermiamo che il lavoro $m + q$, per $q = 1, \dots, m$, va su una macchina il cui unico lavoro ha durata t_{m+1-q} , e il carico diventa

$$t_{m+1-q} + t_{m+q} = 3m - 1.$$

Dimostrazione (4/5)

L'identità. Sia $j = m + 1 - q$, così che $2m + 1 - j = m + q$.

- ▶ Se $j = 2i - 1$ è dispari, allora

$$\lceil j/2 \rceil = i \quad \text{e} \quad \lceil (2m + 1 - j)/2 \rceil = m - i + 1.$$

- ▶ Se $j = 2i$ è pari, le due parti intere superiori sono ancora i e $m - i + 1$.
- ▶ In entrambi i casi

$$t_j + t_{2m+1-j} = (2m - i) + (m + i - 1) = 3m - 1.$$

Dimostrazione (5/5)

L'affermazione. Segue per induzione.

- ▶ Una macchina con due lavori ha carico $3m - 1 > 2m - 1 \geq$ ogni lavoro singolo, quindi non è mai la meno carica finché restano macchine con un solo lavoro.
- ▶ Fra le macchine con un solo lavoro, la meno carica contiene il lavoro rimanente più breve, di durata t_{m+1-q} (a parità di durata i carichi sono uguali).
- ▶ Dopo $2m$ lavori tutti i carichi valgono $3m - 1$, e l'ultimo lavoro, di durata m , dà

$$LPT(I_m) = 3m - 1 + m = 4m - 1,$$

$$\frac{LPT(I_m)}{z(I_m)} = \frac{4m - 1}{3m} = \frac{4}{3} - \frac{1}{3m}.$$



Il rapporto di prestazione nel caso peggiore di *LPT*

- ▶ Dalle due proposizioni su *LPT* segue che *LPT* è un algoritmo di $(\frac{4}{3} - \frac{1}{3m})$ -approssimazione.
- ▶ Il suo rapporto di prestazione nel caso peggiore su m macchine è

$$r(LPT) = \frac{4}{3} - \frac{1}{3m}$$

- ▶ Ordinare la lista abbassa la garanzia di *LIST* da $2 - \frac{1}{m}$ a $\frac{4}{3} - \frac{1}{3m}$.

Osservazioni su LPT

- ▶ Per $m = 3$ la famiglia è $t = (5, 5, 4, 4, 3, 3, 3)$, l'istanza di riferimento: i carichi dopo sei lavori sono $(8, 8, 8) = 3m - 1$, il settimo lavoro dà $LPT(I) = 11 = 4m - 1$, e $z(I) = 9 = 3m$ (l'esempio di LPT visto sopra).
- ▶ L'istanza di riferimento è un caso peggiore per LPT :
$$\frac{11}{9} = \frac{4}{3} - \frac{1}{9}.$$
- ▶ Sulle istanze tight l'ultimo lavoro ha $t_\ell = m = \frac{z(I_m)}{3}$: stanno esattamente sul confine fra i due casi della dimostrazione, e ogni disuguaglianza del Caso 1 vale all'uguaglianza.
- ▶ Il bound e la famiglia sono dovuti a Graham (1969).

Riepilogo (1/2)

LIST:

- ▶ $LIST(I) \leq (2 - \frac{1}{m}) z(I)$ per ogni lista: inizio dell'ultimo lavoro più la sua durata, entrambi limitati da $z(I)$
- ▶ tight: $m(m-1)$ lavori unitari seguiti da un lavoro di durata m

LPT:

- ▶ $LPT(I) \leq (\frac{4}{3} - \frac{1}{3m}) z(I)$: Caso 1 ultimo lavoro breve, Caso 2 al più due lavori per macchina
- ▶ tight: la famiglia di Graham, l'istanza di riferimento per $m=3$

Riepilogo (2/2)

Rapporti di prestazione nel caso peggiore:

- ▶ $r(LIST) = 2 - \frac{1}{m}$
- ▶ $r(LPT) = \frac{4}{3} - \frac{1}{3m}$

Il ruolo dell'ordinamento:

- ▶ ordinare la lista per tempo di esecuzione non crescente abbassa la garanzia da $2 - \frac{1}{m}$ a $\frac{4}{3} - \frac{1}{3m}$

Fonti

Introduzione: Fonti

- ▶ Elenchiamo i due articoli di Graham in cui sono state dimostrate le garanzie di questo capitolo
- ▶ Il primo riguarda *LIST*, il secondo *LPT*

- ▶ R.L. Graham, Bounds for certain multiprocessing anomalies, *Bell System Technical Journal* 45(9):1563–1581, 1966.
- ▶ R.L. Graham, Bounds on multiprocessing timing anomalies, *SIAM Journal on Applied Mathematics* 17(2):416–429, 1969.

Fonti:

- ▶ Graham (1966): la garanzia $2 - \frac{1}{m}$ di *LIST* e la sua famiglia tight
- ▶ Graham (1969): la garanzia $\frac{4}{3} - \frac{1}{3m}$ di *LPT* e la sua famiglia tight

Errori tipici

Introduzione: Errori tipici

- ▶ Raccogliamo gli errori più frequenti con *LIST*, *LPT* e le loro garanzie
- ▶ Controllateli prima di risolvere gli esercizi

Errori tipici (1/2)

Osservazione 2: Usare la garanzia di *LPT* per una lista non ordinata

LIST ha la garanzia $2 - \frac{1}{m}$ per ogni lista; la garanzia migliore $\frac{4}{3} - \frac{1}{3m}$ richiede i lavori ordinati per tempo di esecuzione non crescente.

Osservazione 3: Calcolare il rapporto con il lower bound

Il bound (1) può essere minore di $z(I)$: il rapporto $\mathcal{A}(I)/LB$ è solo un upper bound sul rapporto di prestazione $\mathcal{A}(I)/z(I)$, e i due coincidono solo quando il bound è tight.

Osservazione 4: Risolvere i pareggi con un'altra regola

LIST assegna ogni lavoro a una macchina meno carica di indice minimo; un'altra regola per i pareggi può dare uno schedule diverso, e le soluzioni degli esercizi seguono la regola dell'algoritmo List Scheduling.

Garanzie:

- ▶ $\frac{4}{3} - \frac{1}{3m}$ solo per la lista ordinata; $2 - \frac{1}{m}$ per ogni lista
- ▶ $\mathcal{A}(I)/LB$ limita solo dall'alto il rapporto di prestazione

Pareggi:

- ▶ una macchina meno carica di indice minimo, come nell'algoritmo List Scheduling

In sintesi

Introduzione: In sintesi

- ▶ Raccogliamo le idee chiave del capitolo sull'approssimazione di $P \parallel C_{\max}$
- ▶ Il quadro riassuntivo raccoglie i bound su $z(I)$ trovati finora per l'istanza di riferimento

Riepilogo (1/2)

- ▶ Le garanzie di *LIST* e *LPT* si dimostrano con il bound sui carichi (1) su $z(I)$ e con l'osservazione sull'inizio dell'ultimo lavoro: il lavoro che termina per ultimo inizia non più tardi del carico medio degli altri lavori.
- ▶ Entrambi i rapporti sono tight, con le famiglie di istanze di Graham; per $m = 3$ l'istanza di riferimento è l'istanza tight di *LPT*.
- ▶ Ordinare la lista abbassa la garanzia da $2 - \frac{1}{m}$ a $\frac{4}{3} - \frac{1}{3m}$.

Riepilogo (2/2)

problema	algoritmo	rapporto nel caso peggiore
$P \parallel C_{\max}$	<i>LIST</i>	$2 - \frac{1}{m}$ (tight)
$P \parallel C_{\max}$	<i>LPT</i>	$\frac{4}{3} - \frac{1}{3m}$ (tight)

Quadro riassuntivo dell'istanza di riferimento. I bound su $z(I)$ trovati finora:

Istanza	$P \parallel C_{\max}, m = 3, t = (5, 5, 4, 4, 3, 3, 3)$
Lower bound	9, bound sui carichi
Valori euristici	<i>LIST</i> 12 (lista 7, 6, ..., 1), <i>LPT</i> 11
Valore ottimo	9