Modelli di shop scheduling

1. Considerazioni generali

- In molti ambienti produttivi l'esecuzione di un *job* richiede l'esecuzione non simultanea di un certo numero di operazioni su macchine dedicate.

- Ogni job in ogni istante può essere eseguito su una sola macchina, cioè le operazioni di uno stesso job non possono essere eseguite simultaneamente.

- Ogni macchina può eseguire un job alla volta.

- L'ambiente di processamento è pertanto quello dei modelli di (*general*) *shop scheduling*, così definito:

  - Sono dati n *job* $j = 1, \ldots, n$.
  - Sono date m *macchine dedicate* $M_1, \ldots, M_m$.
  - Il job j è formato da n_j *operazioni* (*task*) $T_j^1, \ldots, T_j^{n_j}$.
  - L'operazione h-esima del job j, T_j^h, deve essere eseguita sulla macchina dedicata $\mu_j^h \in \{M_1, \ldots, M_m\}$ per un *tempo di processamento* p_j^h.

- In generale, le *operazioni* di uno stesso job devono essere eseguite rispettando delle *relazioni di precedenza*.

Modelli di shop scheduling

1. Considerazioni generali (continua)

- L'obiettivo è determinare una schedula ammissibile che minimizza una funzione (regolare) dei tempi di completamento dei job.

- In base al *numero di operazioni di ciascun job* e alle *relazioni di precedenza tra le operazioni di uno stesso job* si definiscono *tre modelli* particolari.

  1) *Open-Shop*:
     - Il job j è costituito da $n_j = m$ operazioni, $\forall\, j$, e l'operazione T_j^h richiede la macchina $\mu_j^h = M_h$, $\forall\, h$.
     - *Non sono date relazioni di precedenza* fra le operazioni di uno stesso job.

  2) *Flow-Shop*:
     - Il job j è costituito da $n_j = m$ operazioni, $\forall\, j$, e l'operazione T_j^i richiede la macchina $\mu_j^i = M_h$, $\forall\, h$.
     - L'esecuzione di T_j^h deve *precedere* quella di T_j^{h+1}, $\forall\, j$, $\forall\, h = \{1, \ldots, m-1\}$.

  3) *Job-Shop*:
     - il *numero* n_j di *operazioni* del job j è *arbitrario*, $\forall\, j$, e l'operazione T_j^h richiede la macchina $\mu_j^h \in \{M_1, \ldots, M_m\}$, $\forall\, h$.
     - L'esecuzione di T_j^h deve *precedere* quella di T_j^{h+1}, $\forall\, j$, $\forall\, h = \{1, \ldots, n_j - 1\}$.

Modelli di shop scheduling

1. Considerazioni generali (continua)

- Una volta che l'operazione T_j^h del job j è stata eseguita dalla macchina μ_j^h, l'operazione viene immagazzinata nel *buffer d'uscita* della macchina μ_j^h se non è disponibile la macchina per processare l'operazione successiva del job j.
- La *capacità di immagazzinamento* può essere *illimitata o limitata*.
- Nel caso di *capacità* di *immagazzinamento illimitata* non appena un job è completato su una macchina, la stessa diventa libera anche se il job appena completato non trova subito disponibile la macchina successiva.
- Nel caso di *capacità* di *immagazzinamento limitata* può invece capitare che, a causa dell'esaurimento di questa capacità, un job pur avendo completato il suo processamento su una macchina non può renderla disponibile per il successivo provocando così un *blocco*.
  In corrispondenza a queste due possibilità occorre considerare modelli ed algoritmi di soluzione diversi.
- Nel seguito esamineremo alcuni modelli di flow-shop, open-shop e job-shop, ipotizzando *capacità* di *immagazzinamento illimitata*.
- Per semplicità di notazione, indicheremo nel seguito con T_{ij} il task T_j^h che richiede la macchina $\mu_j^h = M_i$ e con p_{ij} il suo tempo di processamento.

Modelli di shop scheduling

2. Il modello $O2||C_{\max}$

- In questo modello vi sono 2 macchine ed n job.
- Il generico job j può essere processato prima sulla macchina 1 e poi sulla macchina 2 o viceversa. Il decisore deve *determinare* le *rotte* dei *job* nello shop.
- È ovvio che per il "*makespan*" vale:

$$C_{\max} \geq \gamma = \max\left(\sum_{j=1}^{n} p_{1j}, \sum_{j=1}^{n} p_{2j}\right),$$

  in quanto il *makespan non può essere minore del maggiore carico di lavoro delle due macchine*.
- Poiché nella maggior parte dei casi il makespan è proprio pari al maggiore dei carichi di lavoro, è opportuno esaminare quando è strettamente più grande.
- È possibile mostrare che esiste almeno una schedula ottima senza ritardo.
- Pertanto, consideriamo solo le schedule *senza ritardo* ossia senza idle time non necessari.
  In tali schedule, se c'è un job in attesa di essere eseguito su una macchina e questa si libera non è consentito che essa resti inoperosa.

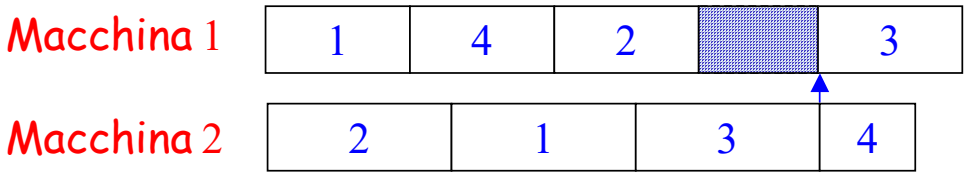
Modelli di shop scheduling

2. Il modello $O2||C_{\max}$ (continua)

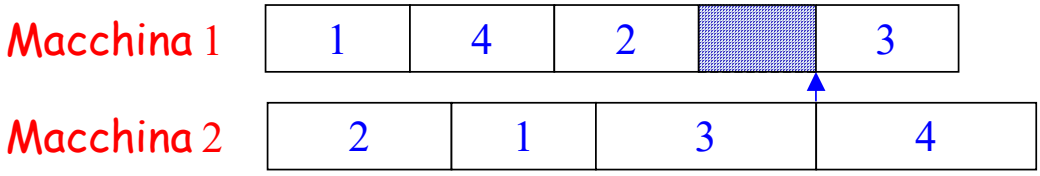
- Pertanto, nelle *schedule senza ritardo* un *idle time si può verificare se e solo se un solo job deve essere ancora processato su una macchina e quando questa si rende libera quest'ultimo job si trova ancora in fase di esecuzione sull'altra macchina*.
- Inoltre, *al più un solo periodo di idle time* si può verificare e al più su una sola macchina.

Esempio

caso a)



caso b)



- Caso a). L'*idle time* provoca un aumento non-necessario del makespan che risulta quindi strettamente maggiore di γ perché il job 3 è l'ultimo job ad essere completato.
- Caso b). L'*idle time* non provoca ciò perché l'ultimo job sulla macchina 1 non è l'ultimo ad essere completato (job 4). Il makespan è pari a γ, cioè al maggiore carico di lavoro delle due macchine.

Modelli di shop scheduling

2. Il modello $O2||C_{\max}$ (continua)

- *Regola di processamento*: quando una macchina è libera, scegli il job in attesa con il *più grande tempo di processamento sull'altra macchina*. Questa regola è detta *Longest Alternate Processing Time* (*LAPT*).
  Al tempo 0 quando entrambe le macchine sono libere può capitare che lo stesso job può essere processato per primo su entrambe le macchine e la scelta può essere fatta indifferentemente su una delle due macchine.
- *Conseguenza*: quando una macchina si è liberata i *job che hanno già completato il loro processamento sull'altra macchina hanno la priorità più bassa (ossia zero) sulla macchina appena liberata*. Non c'è quindi differenza di priorità tra job che sono stati già processati su una certa macchina e aspettano per il processamento sull'altra.

Teorema

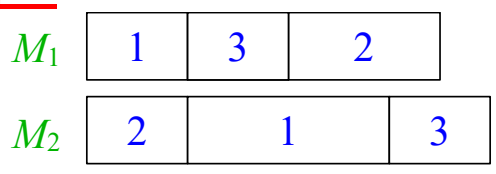
La regola *LAPT* dà una *schedula ottima* per $O2||C_{\max}$ con

$$C^*_{\max} = \max\left(\max_{j\in\{1,\ldots,n\}} (p_{1j} + p_{2j}), \sum_{j=1}^{n} p_{1j}, \sum_{j=1}^{n} p_{2j} \right)$$

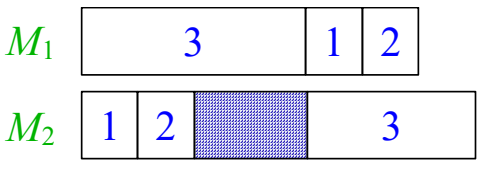
Il teorema assicura il non verificarsi di situazioni tipo il caso a) precedente. In particolare, $C^*_{\max} > \gamma$ solo se esiste un job il cui tempo totale è maggiore del massimo carico di lavoro delle due macchine.

Esempio

a)



b)



Modelli di shop scheduling

3. Il modello $O3||C_{\max}$

Teorema

Il problema $O3||C_{\max}$ è *NP-hard*

Dimostrazione

Attraverso riduzione dal problema *PARTITION*.
Poiché il problema *PARTITION* è *NP-completo in senso ordinario* (esiste algoritmo esatto pseudopolinomiale) il problema di scheduling $O3||C_{\max}$ è NP-hard in senso ordinario.
N.B.: Si veda l'Appendice per i dettagli della riduzione.

- Definizione del problema *PARTITION*

  Dato un insieme S di z interi positivi $a_1, a_2, \ldots, a_z$ e un intero B tali che $\sum_{i=1}^{z} a_i = 2B$, esiste una partizione di S in due sottoinsiemi (disgiunti e ricoprenti) S_1, S_2 tali che $\sum_{a_i \in S_j} a_i = B$, per $j = 1, 2$?

Dal teorema segue che:

Corollario

Il problema $Om||C_{\max}$, con $m \geq 3$ è NP-hard.

Modelli di shop scheduling

4. Il modello $F2||C_{\max}$

- In questo caso si considera un *flow-shop* con $m = 2$ *macchine in serie*. Ciascun job sarà eseguito prima sulla macchina 1 e poi sulla macchina 2.
- Siano n i job da schedulare essendo p_{1j} il tempo di processamento del generico job j sulla macchina 1 e p_{2j} quello sulla macchina 2.
- Questo problema è stato uno dei primi ad essere studiato ed è noto nella letteratura come "*Johnson's problem*" dal nome nella persona che trovò la regola di minimizzazione del makespan. (*Johnson's rule*)
- La *regola di Johnson* è la seguente:

  (i) Partiziona l'insieme dei i job in 2 sottoinsiemi:
  - S_1 contenente tutti i job con $p_{1j} < p_{2j}$
  - S_2 contenente tutti i job per i quali $p_{1j} > p_{2j}$.

  I job con $p_{1j} = p_{2j}$ possono essere inseriti nell'uno o nell'altro insieme.

  (ii) Schedula, seguendo lo stesso ordine sulle due macchine, *prima* l'insieme S_1 secondo la regola *Shortest Processing Time* (*SPT*) sui tempi p_{1j} e subito *dopo* S_2 secondo la regola *Longest Processing Time* (*LPT*) sui tempi p_{2j}.

  - Una schedula siffatta è denotata come *SPT*(1) – *LPT*(2) ed è *ottima*.

Modelli di shop scheduling

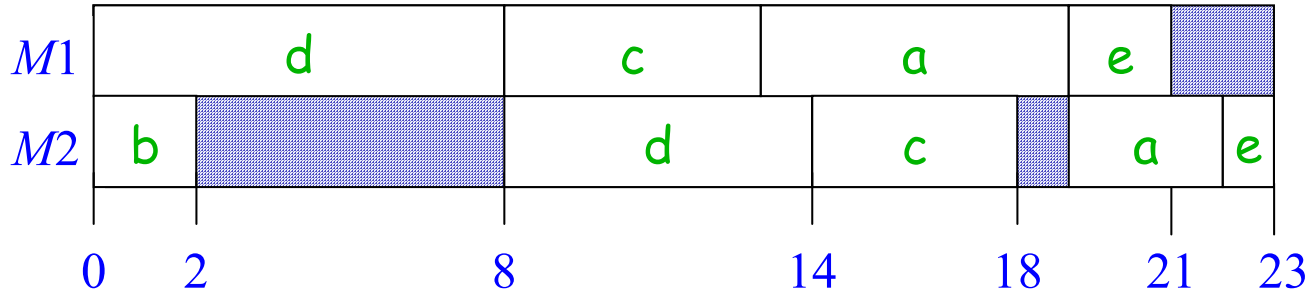
4. Il modello $F2||C_{\max}$ (continua)

Esempio 1

job	a	b	c	d	e
p_{1j}	6	0	5	8	2
p_{2j}	3	2	4	6	1

$S_1 = \{b\}$
$S_2 = \{a, c, d, e\}$

La schedula SPT(1) – LPT(2) è la seguente:



- Le schedule SPT(1) – LPT(2) *non sono le uniche ottime*. La classe di tutte le schedule ottime per questo problema è difficile da caratterizzare e dipende dai dati.
- Ad esempio, se esiste un job k con un tempo p_{1k} *molto piccolo* e con un tempo $p_{2k} \geq \sum_{j \neq k} p_{1j}$ allora il job k è il primo in una sequenza ottima ed i restanti job non influenzano il makespan.

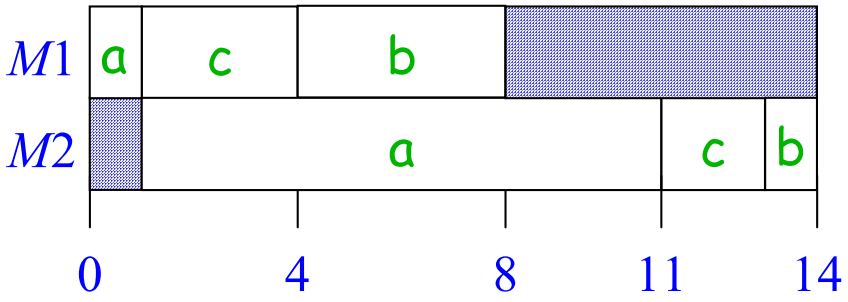
Modelli di shop scheduling

4. Il modello $F2||C_{\max}$ (continua)

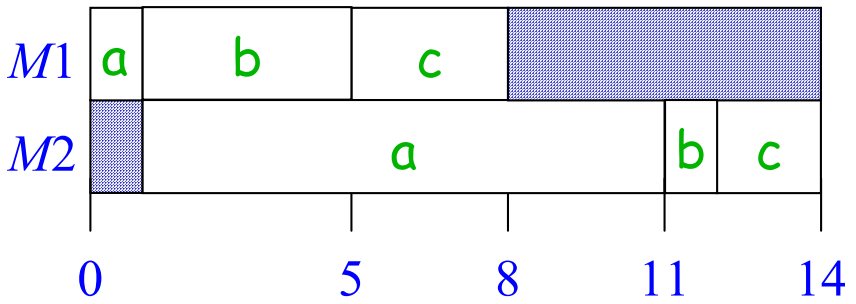
Esempio 2

job	a	b	c
p_{1j}	1	4	3
p_{2j}	10	1	2

$S_1 = \{a\}$
$S_2 = \{b, c\}$



a)



b)

Poiché p_{1a} è il più piccolo allora il job a si pone in prima posizione; inoltre visto che $p_{2a} > \sum_{j \neq a} p_{1j}$ gli altri job possono essere sequenziati in qualsiasi ordine, perché non incidono sul makespan.

Modelli di shop scheduling

5. Il modello $F3||C_{\max}$

Teorema

Il problema $F3||C_{\max}$ è *NP-hard in senso forte*

Dimostrazione

Attraverso riduzione dal problema *3-PARTITION*.

Poiché il problema *3-PARTITION* è *NP-completo in senso forte* il correlato problema di scheduling è NP-hard in senso forte.

N.B.: Si veda l'Appendice per i dettagli della riduzione.

- Definizione del problema *3-PARTITION*

  Dato un insieme S di $3z$ numeri interi positivi $a_1, \ldots, a_{3z}$, e un intero positivo B tali che

  $$\frac{B}{4} < a_i < \frac{B}{2}, \quad i = 1,...,3z \qquad \text{e} \qquad \sum_{i=1}^{3z} a_i = z\,B,$$

  esiste una partizione di S in z *sottoinsiemi* (disgiunti e ricoprenti) $S_1, \ldots, S_z$ ciascuno costituito da 3*interi*, tali che $\Sigma_{a_i \in S_j} a_i = B$, $\forall\, j \in \{1, \ldots, z\}$?

Corollario

Il problema $Fm||C_{\max}$, con $m \geq 3$ è NP-hard in senso forte.

Esempio

Dato $S = \{21, 24, 28, 26, 30, 31\}$ e $B = 80$. In questo caso $20 < a_i < 40$, per $i = 1, 2, \ldots, 6$, e $\Sigma_i a_i = 160$, pertanto $z = 2$.

Esistono $z = 2$ insiemi disgiunti e ricoprenti $S_1 = \{21, 28, 31\}$ $S_2 = \{24, 26, 30\}$, per i quali $\Sigma_{a_i \in S_j} a_i = 80$, $j = 1, 2$.

Modelli di shop scheduling

6. Il modello $Fm||C_{\max}$

- Senza perdita di generalità, possiamo circoscrivere la ricerca delle soluzioni ottime tra le schedule semiattive.

- Una *schedula semiattiva* per il problema $Fm||C_{\max}$ è definita a partire da m sequenze σ_1, σ_2, …, σ_m di visita degli n job sulle m macchine. σ_i è la sequenza dei job sulla macchina M_i.

- La schedula semiattiva si ottiene schedulando al più presto i job nel rispetto dei:
  - vincoli di precedenza tra le operazioni di uno stesso job
  - e dagli ulteriori vincoli di precedenza tra le operazioni che richiedono le stessa macchina, una volta assegnate le m sequenze.

- In generale la sequenza σ_{i+1} sulla macchina $i+1$ è diversa dalla σ_i sulla macchina i, quindi le possibili schedule semiattive sono $(n!)^m$.

Modelli di shop scheduling

6. Il modello $Fm||C_{\max}$

Teorema

Nel problema $Fm||C_{\max}$, esiste sempre una soluzione ottima in cui $\sigma_1 = \sigma_2$ e $\sigma_{m-1} = \sigma_m$.

Dimostrazione

Si consideri una schedula ottima per cui $\sigma_1 \neq \sigma_2$ e nella quale quindi ci sia un job j che precede direttamente k in σ_1 e che viceversa k preceda (eventualmente non direttamente) j in σ_2. Scambiando j e k in σ_1 non si modifica la schedula dei job su M_2. Ripetendo questo procedimento si ottiene una schedula ottima in cui le sequenze su M_1 e su M_2 sono uguali. Il discorso è analogo considerando le ultime due sequenze σ_{m-1} e σ_m, e modificando il sequenziamento su M_m fino a renderlo uguale a quello su M_{m-1}.

Corollario

Per $Fm||C_{\max}$, con $m \leq 3$, esiste una schedula ottima tra le $n!$ schedule di permutazione. Per $m \geq 4$ la schedula ottima va cercata tra $(n!)^{m-2}$ schedule semiattive.

- Tra le possibili soluzioni è interessante esaminare il sottoinsieme delle *schedule di permutazione*, cioè quelle associate ad uno *stesso sequenziamento dei job su tutte le macchine* ($\sigma_{i+1} = \sigma_i$).
- Le schedule di permutazione sono $n!$

Modelli di shop scheduling

Analizziamo nel seguito alcuni modelli di job shop

Considereremo il caso particolare in cui non esiste nessuna coppia di operazioni di uno stesso job che richiedono la stessa macchina: $\forall\ T_j^h$ e $T_j^k \rightarrow \mu_j^h \neq \mu_j^k$.

Per semplicità di notazione, possiamo quindi indicare nel seguito con T_{ij} *il task* T_j^h *che richiede la macchina* $\mu_j^h = M_i$ *e con* p_{ij} *il suo tempo di processamento.*

7. Il modello $J2||C_{\max}$

- In questo modello vi sono 2 macchine ed n job.
- *Alcuni job* devono essere processati *prima sulla macchina* 1 *e poi sulla macchina* 2. I *restanti job* ovviamente devono essere processati *prima sulla macchina* 2 *e poi sulla macchina* 1.
- I tempi di processamento del generico job j sono p_{1j} e p_{2j} rispettivamente per la macchina 1 e per la macchina 2. L'obiettivo è minimizzare il makespan.
- Il problema è trattabile.

Modelli di shop scheduling

7. Il modello $J2||C_{\max}$ (continua)

Algoritmo di soluzione

Questo problema può essere ricondotto al problema $F2||C_{\max}$ già visto come segue. Siano

$J_{1,2}$ Insieme di job che devono essere processati prima sulla macchina 1

$J_{2,1}$ Insieme di job che devono essere processati prima sulla macchina 2

I) Conseguenza:

Quando un job di $J_{1,2}$ ha completato il processamento sulla macchina 1 il posticipare il suo processamento sulla macchina 2 non ha alcun effetto sul makespan fintanto che questa resta occupata. Lo stesso vale dualmente per un job dell'insieme $J_{2,1}$.

II) Conseguenza:

I job di $J_{1,2}$ hanno una più alta priorità sulla macchina 1 di qualsiasi job di $J_{2,1}$ mentre i job di $J_{2,1}$ hanno una più alta priorità sulla macchina 2 di qualsiasi job di $J_{1,2}$.

Modelli di shop scheduling

7. Il modello $J2||C_{\max}$ (continua)

Con riferimento alle precedenti considerazioni la schedula ottima è simile a quanto ottenuto con la schedula *LAPT* per il problema $O2||C_{\max}$: *i job già processati su una macchina hanno la più bassa priorità sull'altra macchina*.

Resta quindi da *definire* la *sequenza* σ_1 *dei job* di $J_{1,2}$ *sulla macchina* 1 e la *sequenza* σ_2 *dei job* $J_{2,1}$ *sulla macchina* 2.

Queste due *sequenze* possono essere *determinate* considerando il *sequenziamento* di $J_{1,2}$ e $J_{2,1}$, ognuno come un *problema* $F2||C_{\max}$ in cui rispettivamente viene utilizzata prima la macchina 1 e poi la macchina 2 ovvero prima la macchina 2 e poi la macchina 1.

Questo porta a sequenze *SPT*(1) - *LPT*(2) per ognuno dei due insiemi di job con priorità tra gli insiemi definite in precedenza.

Modelli di shop scheduling

7. Il modello $J2||C_{max}$ (continua)

Esempio

job	a	b	c	d	e
p_{1j}	6	1	5	8	2
p_{2j}	3	2	6	6	1

Siano gli insiemi $J_{1,2} = \{a, c\}$; $J_{2,1} = \{b, d, e\}$
I due problemi $F2||C_{max}$ sono rispettivamente definiti sugli insiemi $J_{1,2}$ e $J_{2,1}$; per il primo (i) la *prima macchina* è la 1, e per il secondo (ii) la *prima macchina* è la 2.

Pertanto definiti

(i) $S_1(J_{1,2}) = \{c\}$; $S_2(J_{1,2}) = \{a\}$
(ii) $S_1(J_{2,1}) = \{d, e\}$; $S_2(J_{2,1}) = \{b\}$

le relative sequenze *SPT*(1)-*LPT*(2) sono:

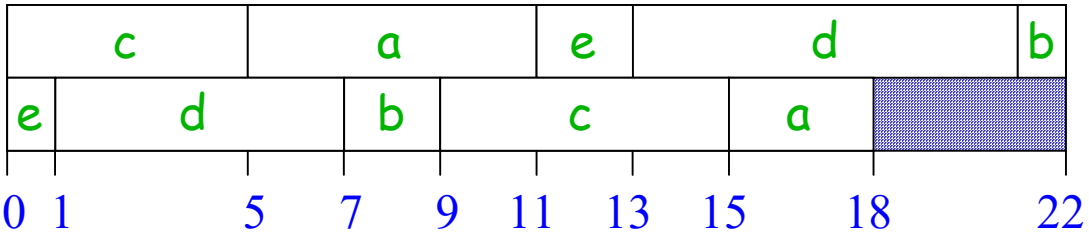
(i) $\sigma_1 = (c, a)$; (ii) $\sigma_2 = (e, d, b)$

Le sequenze complessive sulle due macchine sono quindi

$\overline{\sigma}_1 = (c, a, \sigma(J_{2,1}))$; $\overline{\sigma}_2 = (e, d, b, \sigma(J_{1,2}))$,

con $\sigma(J_{1,2}))$ e $\sigma(J_{2,1}))$ sequenze qualsiasi di $J_{1,2}$ e $J_{2,1}$ (rispettiv.), che non inducano idle time non necessari.

Sia, ad esempio, $\sigma(J_{1,2})) = \sigma_1$ e $\sigma(J_{2,1})) = \sigma_2$, e quindi $\overline{\sigma}_1 = (c, a, e, d, b)$ $\overline{\sigma}_2 = (e, d, b, c, a)$, la schedula complessiva è:



Modelli di shop scheduling

8. Il modello $J3||C_{\max}$

Teorema

Il problema $J3||C_{\max}$ è *NP-hard in senso forte*

Dimostrazione

Il teorema si dimostra semplicemente notando che $J3||C_{\max}$ è una generalizzazione di $F3||C_{\max}$.

Sapendo quindi che $F3||C_{\max}$ è NP-hard in senso forte, ciò implica che anche $J3||C_{\max}$ è NP-hard in senso forte.

Dal teorema precedente segue che:

Corollario

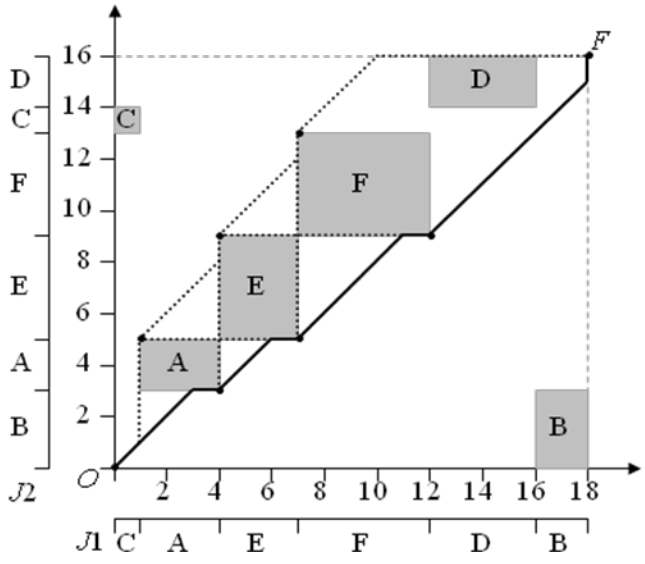
Il problema $Jm||C_{\max}$, con $m \geq 3$ è NP-hard in senso forte.

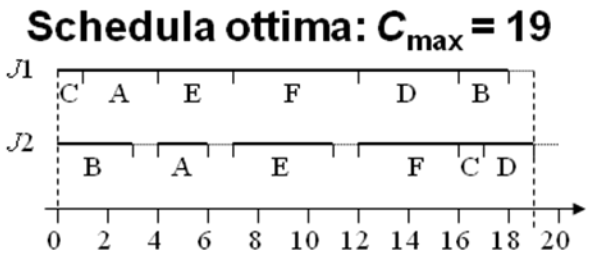
Modelli di shop scheduling

9. Il modello $J|n = 2|C_{\max}$

- Un caso facilmente risolvibile del job-shop con $m > 2$ macchine è quello in cui il numero dei job $n = 2$.
- Il problema può essere formulato come problema di cammino minimo nel piano con ostacoli rappresentati da oggetti rettangolari:
  - Gli assi x e y del piano rappresentano due assi temporali.
  - Le durate delle operazioni del job 1 (job 2) sono rappresentate da intervalli sull'asse x (asse y) che sono arrangiate secondo l'ordine di processamento delle operazioni del job
  - Gli intervalli sono etichettati con la macchina richiesta dalla operazione relativa all'intervallo temporale
  - F è il punto sul piano di coordinate pari alle durate dei job

Esempio: 6 macchine (A, B, C, D, E, F); job 1 ($J1$) è composto da 6 operazioni con rotta sulle macchine (C, A, E, F, D, B) e durate (1, 3, 3, 5, 4, 2) e job 2 ($J2$) è composto da 6 operazioni con rotta sulle macchine (B, A, E, F, C, D) e durate (3, 2, 4, 4, 1, 2).





Modelli di shop scheduling

9. Il modello $J|n=2|C_{\max}$ *(continua)*

- Una schedula ammissibile corrisponde ad un cammino dall'origine O al punto F. Questo cammino ha le seguenti proprietà:
  1. Il cammino consiste di segmenti che sono o paralleli ad uno dei due assi cartesiani (corrispondenti alla esecuzione di un solo job) o sono obliqui a 45° (corrispondenti alle esecuzioni simultanee dei due job);
  2. Il cammino deve evitare gli ostacoli rettangolari, corrispondenti all'impiego simultaneo (non ammissibile) della stessa macchina, in quanto due operazioni non possono simultaneamente essere eseguite dalla stessa macchina e l'interruzione dell'esecuzione di una operazione non è ammessa;
  3. La lunghezza del cammino, cioè la lunghezza della schedula (tempo richiesto per eseguire i due job), è pari alla somma delle lunghezze dei segmenti orizzontali, dei segmenti verticali, e dei segmenti obliqui diviso per radice di 2.

- Il problema consiste nell'identificare il cammino da O a F che massimizza la lunghezza totale dei segmenti obliqui.

Modelli di shop scheduling

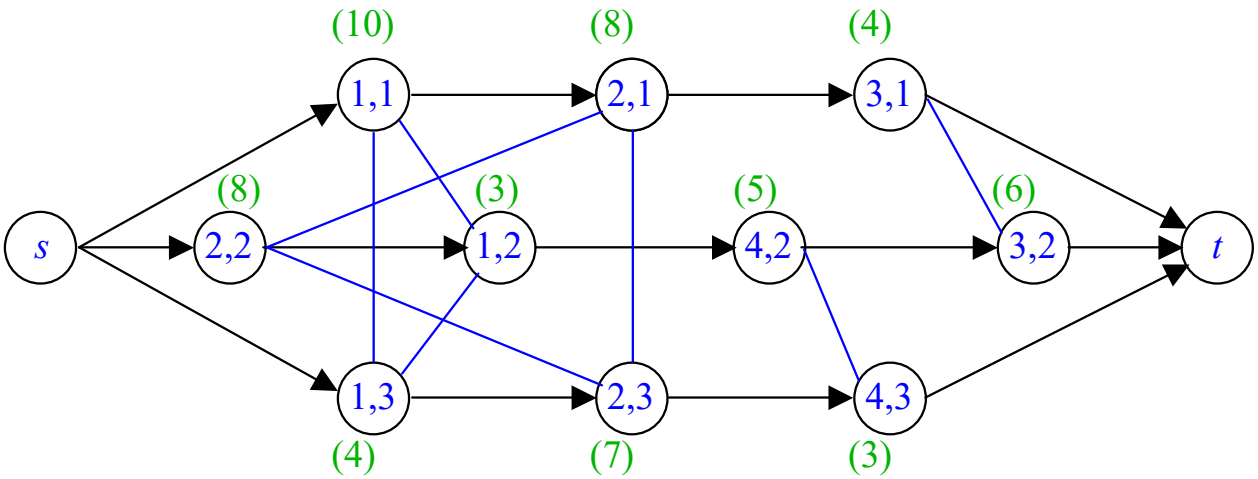
10. Il modello $Jm||C_{\max}$

10. 1 Rappresentazione su grafo disgiuntivo

- Il problema $Jm||C_{\max}$ può essere rappresentato tramite *grafo disgiuntivo* (pesato) $G = (V, A \cup E)$, con:
  - V, corrispondente all'*insieme dei task* unione $\{s, t\}$
  - A, insieme di *archi (orientati) congiuntivi* (rappresentanti relazioni di precedenza diretta tra i task)
  - E, insieme di *archi (non-orientati) disgiuntivi* (tra coppie di task che utilizzano la stessa macchina)
  - Ad ogni vertice è associato un peso pari al tempo di processamento p_{ij} del task T_{ij} rappresentato. I vertici s e t hanno peso 0.

Esempio

Job	*sequenza sulle macchine*	*tempi di processamento*
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{22}=8, p_{12}=3, p_{42}=5, p_{32}=6
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



Modelli di shop scheduling

10. Il modello $Jm||C_{\max}$

10.1 Rappresentazione su grafo disgiuntivo (continua)

- Una *soluzione ammissibile* corrisponde ad un *orientamento aciclico* di $G = (V, A \cup E)$, ottenuto orientando gli archi in E.
- Il *makespan* $C_{\max}$ della schedula associata all'orientamento aciclico di G è *pari* alla *lunghezza* l_π del *cammino (critico) di lunghezza massima* π da s a t.
- Il *problema si riduce* quindi a *determinare un orientamento aciclico che minimizza la lunghezza del cammino di lunghezza massima*.

10.2 Lower Bound

- 1° Ipotesi: *Rilassare completamente i vincoli sulle risorse*, supponendo che *tutte le macchine possono eseguire più di un task alla volta*.
  LB1: lunghezza del cammino critico del sottografo G' del grafo G, depurato degli archi non orientati.
- 2° Ipotesi: *Rilassare quasi completamente i vincoli sulle risorse*, supponendo che *tutte le macchine tranne una possono eseguire più di un task alla volta*.
  $LB2 = LB1 + \max_i(\delta_i)$, dove δ_i è il minimo tempo addizionale necessario per schedulare (in sequenza) i task sulla macchina i.

Modelli di shop scheduling

10. Il modello $Jm||C_{\max}$
10.2 Lower Bound (continua)

- Calcolo di δ_i:
  Per ogni task T_{ij}, siano r_{ij} il suo *earliest start time* e d_{ij} il suo *latest finish time* nell'ipotesi 1°.

  - *Calcolo di r_{ij} e d_{ij}:*

    Supponiamo di aver rinumerato i q task (nodi) $T_{(k)}$ secondo ordinamento *topologico*: $k < l$ solo se $T_{(l)} \not< T_{(k)}$
    Sia $T_{(0)}$ il task fittizio iniziale, corrispondente al nodo sorgente $s \equiv T_{(0)}$, e sia $T_{(q+1)}$ il task fittizio finale, corrispondente al nodo pozzo $t \equiv T_{(q+1)}$.
    Siano:

    $P_{(k)}$: insieme *predecessori diretti* di $T_{(k)}$
    $S_{(k)}$: insieme *successori diretti* di $T_{(k)}$

  - Gli istanti di tempo $r_{(k)}$ e $d_{(k)}$ di $T_{(k)}$ si calcolano con le seguenti procedure *forward* e *backward* che *visitano in larghezza* G' rispettivamente *in avanti* da s a t e *all'indietro* da t a s.

procedura forward	*procedura backward*
$r_{(0)} := 0;$ for $k := 1$ to $q + 1$ do $r_{(k)} := \max[r_{(j)}+p_{(j)} \mid T_{(j)} \in P_{(k)}]$	$d_{(q+1)} := r_{(q+1)};$ for $k := q$ downto 0 do $d_{(k)} := \min[d_{(l)}-p_{(l)} \mid T_{(l)} \in S_{(k)}]$

Modelli di shop scheduling

10. Il modello $Jm||C_{\max}$

10.2 Lower Bound (continua)

Il tempo addizionale richiesto dal task T_{ij} è pari alla sua *tardiness* rispetto a d_{ij}, mentre è ovvio che il task non può iniziare prima di r_{ij}.

Quindi il *tempo addizionale* necessario per sequenziare i task T_{ij} richiedenti la macchina i, è pari alla *massima tardiness* tra tali task.

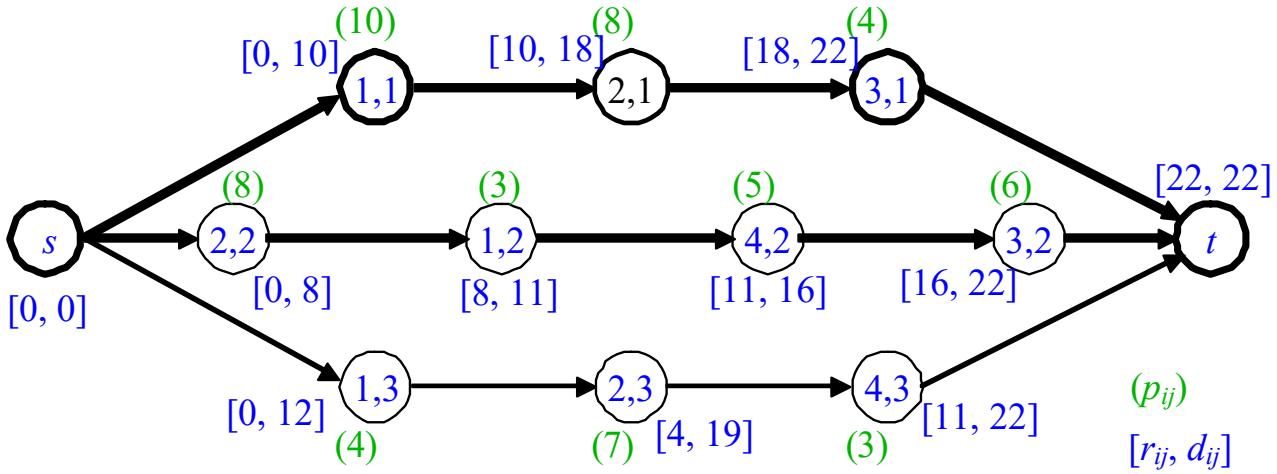
Determinare il *minimo tempo addizionale* δ_i equivale a risolvere il problema $1|r_j|T_{\max}$ (ovvero $1|r_j|L_{\max}$) sulla macchina i e sui task T_{ij} che la richiedono, con *release date* r_{ij} e *due date* d_{ij}. Quindi, $\delta_i = T^*_{\max} = \max(0, L^*_{\max})$.

N.B.: $1|r_j|L_{\max}$ è NP-hard in s. f., ma si può risolvere in tempi ragionevoli con B&B.

Esempio

Riconsideriamo l'esempio precedente.

LB1 = 22 è pari alla lunghezza del cammino critico del grafo disgiuntivo depurato degli archi non orientati



10. Il modello $Jm||C_{\max}$

10.2 Lower Bound (continua)

Calcoliamo $LB2 = LB1 + \max_i(\delta_i)$.

- Macchina 1 $\Rightarrow$ δ_1

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

Job	1	2	3
p_j	10	3	4
r_j	0	8	0
d_j	10	11	12

La sequenza ottima è $(1, 2, 3)$ con $L^*_{\max} = 5$. $\Rightarrow \delta_1 = 5$

- Macchina 2 $\Rightarrow$ δ_2

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

Job	1	2	3
p_j	8	8	7
r_j	10	0	4
d_j	18	8	19

La sequenza ottima è $(2, 3, 1)$ con $L^*_{\max} = 5$. $\Rightarrow \delta_2 = 5$

- Macchina 3 $\Rightarrow$ δ_3

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

Job	1	2
p_j	4	6
r_j	18	16
d_j	22	22

La sequenza ottima è $(2, 1)$ con $L^*_{\max} = 4$. $\Rightarrow$ $\delta_3 = 4$

10. Il modello $Jm||C_{\max}$

10.2 Lower Bound (continua)

- Macchina 4 $\Rightarrow$ δ_4

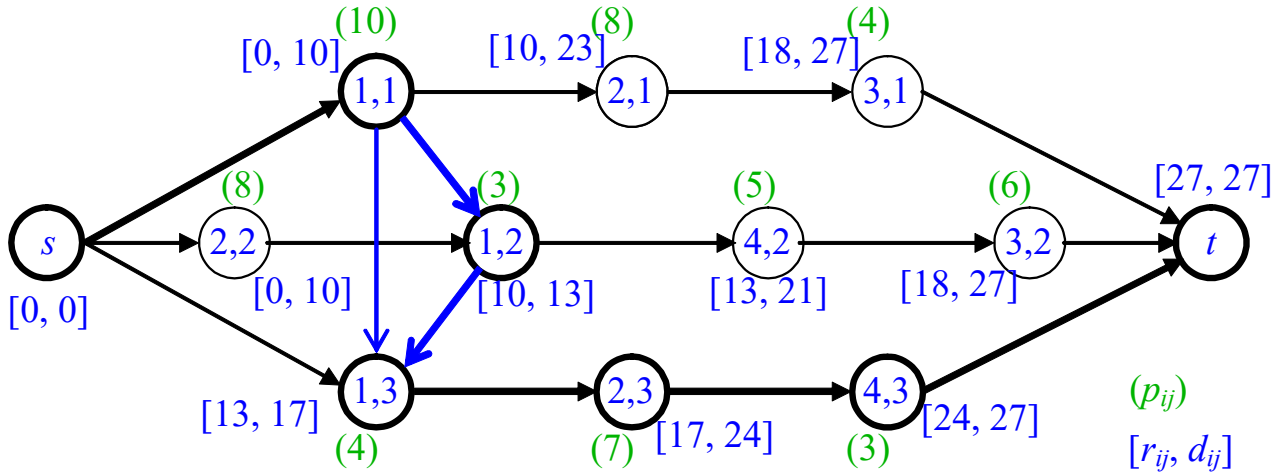
Risolviamo la seguente istanza di $1|r_j|L_{\max}$

Job	2	3
p_j	5	3
r_j	11	11
d_j	16	22

La sequenza ottima è $(2, 3)$ con $L^*_{\max} = 0$. $\Rightarrow$ $\delta_4 = 0$

Pertanto, $\max_i(\delta_i) = 5$ e quindi $\text{LB2} = 22 + 5 = 27$.

LB2 è pari alla lunghezza del cammino critico del grafo orientato aciclico ottenuto da quello disgiuntivo G depurato degli archi disgiuntivi (non orientati), ad eccezione di quelli definiti tra coppie di nodi relativi a task che richiedono la *macchina bottleneck* $i^* = \text{argmax}_i\{\delta_i\}$ e che sono orientati secondo la sequenza ottima del problema di $1|r_j|L_{\max}$ per il calcolo di δ_{i^*}. Nel nostro caso $i^* = 1$, con sequenza $(1, 2, 3)$ su di essa.



Modelli di shop scheduling

10. Il modello $Jm||C_{\max}$

10.3 Euristica Shifting Bottleneck

- È una delle euristiche di maggior successo.
- È una *procedura iterativa* che consta di m iterazioni.
- *Ad ogni iterazione* viene *individuata la macchina che costituisce il bottleneck del sistema*, e *determinata la miglior sequenza dei task da processare su di essa*. Questo comporta la definizione dell'orientamento degli archi disgiuntivi tra coppie di nodi che rappresentano tali task.
- Indichiamo con M l'insieme delle m macchine dello shop.
- Ad una generica iterazione, indichiamo con M_0 il sottoinsieme delle macchine per le quali è stata specificata la sequenza dei job.
- Pertanto nella generica iterazione il primo passo da fare è *individuare la macchina bottleneck* in $M \backslash M_0$.
- Per determinare le *macchina bottleneck*, si individua quale macchina in $M \backslash M_0$ è *causa del maggior accumulo di ritardo* a causa del sequenziamento dei task su di essa.

Modelli di shop scheduling

10. Il modello $Jm||C_{\max}$

10.3 Euristica Shifting Bottleneck (continua)

- Sia $G(M_0)$ il grafo disgiuntivo orientato (aciclico) con gli archi disgiuntivi orientati per le macchine schedulate e depurato di quelli non ancora orientati.

- La *lunghezza del cammino critico* di $G(M_0)$ è pertanto il *makespan* $C_{\max}(M_0)$ della schedula ottima supponendo di rilassare i vincoli sulle macchine $M \backslash M_0$ e avendo sequenziato i task sulle macchine di M_0 secondo quanto fatto nelle iterazioni precedenti

- la *macchina bottleneck* è quella k per cui $C_{\max}(M_0 \cup \{k\}) = C_{\max}(M_0) + \delta_k$ è *massimo*, dove δ_i è il minimo tempo addizionale necessario per sequenziare i task sulla macchina $i \in M \backslash M_0$.

- Calcolo di δ_i:
  Per ogni task T_{ij}, siano r_{ij} il suo *earliest start time* e d_{ij} il suo *latest finish time* calcolato su $G(M_0)$ noto $C_{\max}(M_0)$.
  Determinare δ_i equivale a risolvere un'istanza del problema $1|r_j|L_{\max}$ sui task T_{ij} che richiedono la macchina i, con *release date* r_{ij} e *due date* d_{ij}. Quindi, $\delta_i = T^*_{\max} = \max(0, L^*_{\max})$.
  N.B.: $1|r_j|L_{\max}$ è NP-hard in senso forte, ma può essere risolto in tempi ragionevoli con B&B.

Modelli di shop scheduling

10. Il modello $Jm||C_{\max}$

10.3 Euristica Shifting Bottleneck (continua)

- Determinata la *macchina bottleneck* k, i task vengono schedulati su di essa secondo la sequenza ottima ottenuta risolvendo il problema ausiliario $1|r_j|L_{\max}$.

- Prima di terminare la generica iterazione si riottimizza localmente la soluzione parziale ottenuta sequenziando le macchine $M_0 \cup \{k\}$.
  Per ogni h di M_0 si risequenziano (se conveniente) i task sulla macchina h in base alla sequenza ottima per il problema $1|r_j|L_{\max}$ definito sui task da eseguire su h in base a $G(\{k\} \cup M_0 \setminus \{h\})$ e noto $C_{\max}(\{k\} \cup M_0 \setminus \{h\})$.

- La struttura di questa euristica mostra la *relazione tra il concetto di bottleneck e i concetti di cammino critico e tardiness massima*.

- Un *cammino critico indica dove e quando si verifica una situazione di bottleneck*, la *tardiness massima* indica quale sarebbe l'*aumento del makespan dovuto all'inserimento della macchina bottleneck* tra le macchine schedulate.

Modelli di shop scheduling

10. Il modello $Jm||C_{\max}$:
10.3 Euristica Shifting Bottleneck (continua)

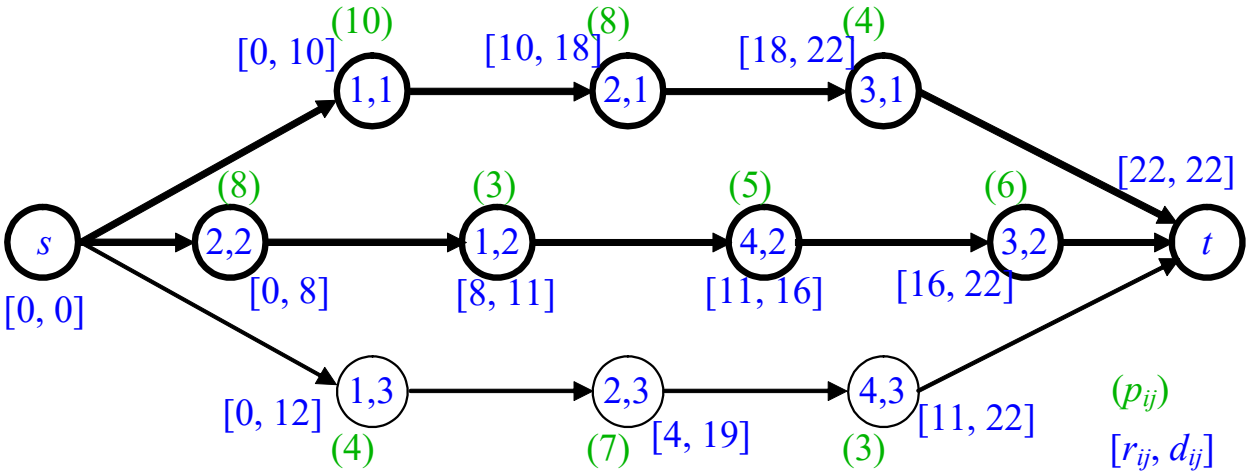
Esempio

Riconsideriamo l'esempio precedente.

Applichiamo l'euristica *shifting bottleneck*

Inizializzazione: M_0 è vuoto.

Iter. 1: $C_{\max}(M_0) = 22$ pari alla lunghezza del cammino critico di $G(M_0)$.



Determiniamo la macchina da schedulare

- Macchina 1 $\Rightarrow$ δ_1

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	2	3
p_j	10	3	4
r_j	0	8	0
d_j	10	11	12

La sequenza ottima è $(1, 2, 3)$ con $L^*_{\max} = 5 \Rightarrow \delta_1 = 5$

10. Il modello $Jm||C_{\max}$

10.3 Euristica Shifting Bottleneck (continua)

- Macchina 2 $\Rightarrow$ δ_2

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	2	3
p_j	8	8	7
r_j	10	0	4
d_j	18	8	19

La sequenza ottima è $(2, 3, 1)$ con $L^*_{\max} = 5$. $\Rightarrow \delta_2 = 5$

- Macchina 3 $\Rightarrow$ δ_3

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	2
p_j	4	6
r_j	18	16
d_j	22	22

La sequenza ottima è $(2, 1)$ con $L^*_{\max} = 4$. $\Rightarrow \delta_3 = 4$

- Macchina 4 $\Rightarrow$ δ_4

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	2	3
p_j	5	3
r_j	11	11
d_j	16	22

La sequenza ottima è $(2, 3)$ con $L^*_{\max} = 0$. $\Rightarrow \delta_4 = 0$

Modelli di shop scheduling

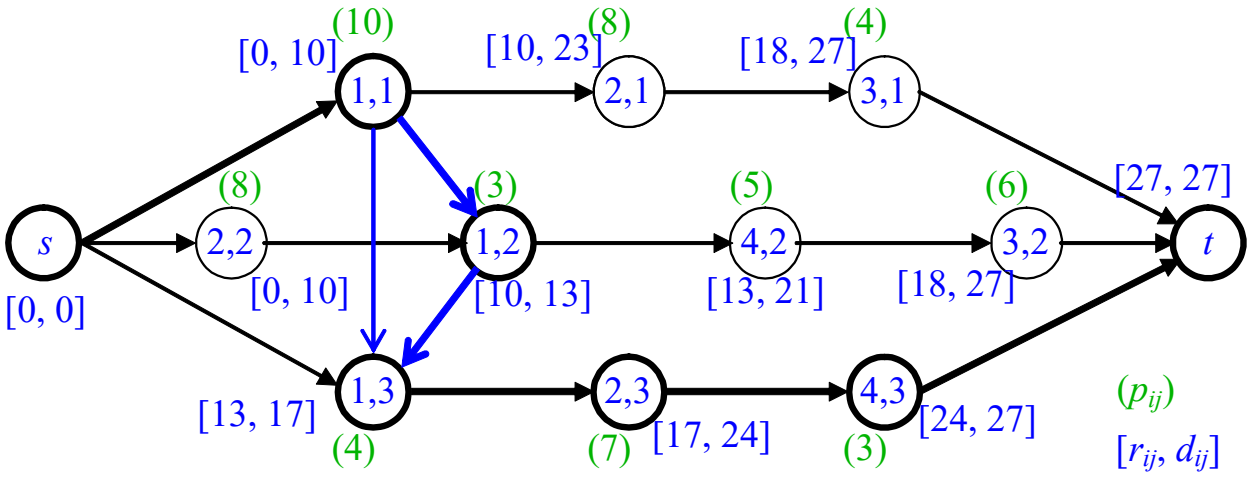
10. Il modello $Jm||C_{\max}$

10.3 Euristica Shifting Bottleneck (continua)

Le macchine candidate sono 1 e 2 ($L_{\max} = 5$)
Scegliamo la 1. $M_0 = \{1\}$.

Sequenziamo i job sulla 1 secondo $(1, 2, 3)$

Iter. 2: $C_{\max}(M_0) = 27$ pari alla lunghezza del cammino critico di $G(M_0)$.



Determiniamo la macchina da schedulare

- Macchina 2 $\Rightarrow$ δ_2
  Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	2	3
p_j	8	8	7
r_j	10	0	17
d_j	23	10	24

La sequenza ottima è $(2, 1, 3)$ con $L^*_{\max} = 1 \Rightarrow \delta_2 = 1$

10. Il modello $Jm||C_{\max}$

10.3 Euristica Shifting Bottleneck (continua)

- Macchina 3 $\Rightarrow$ δ_3

  Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	2
p_j	4	6
r_j	18	18
d_j	27	27

  La sequenza ottima è $(1, 2)$ con $L^*_{\max} = 1$. $\Rightarrow \delta_3 = 1$

- Macchina 4 $\Rightarrow$ δ_4

  Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	2	3
p_j	5	3
r_j	13	24
d_j	21	27

  La sequenza ottima è $(2, 3)$ con $L^*_{\max} = 0$. $\Rightarrow \delta_4 = 0$

Le macchine candidate sono 2 e 3 ($\delta_{\max} = \delta_2 = \delta_3$).
Scegliamo la 2. $M_0 = \{1, 2\}$.

Sequenziamo i job sulla 2 secondo $(2, 1, 3)$

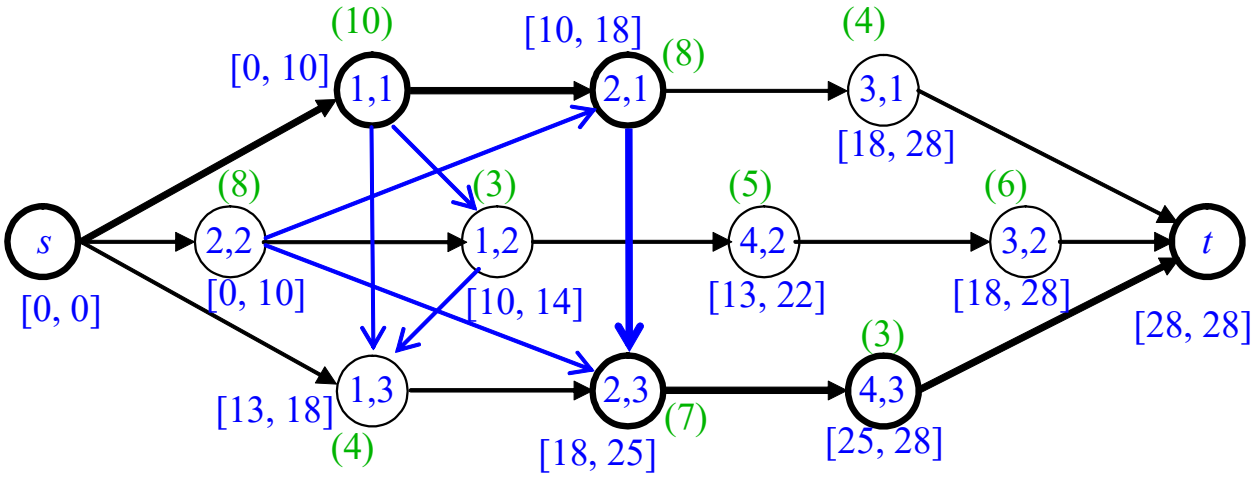
Cercando di risequenziare la macchina 1 non si ottiene alcun miglioramento.

Modelli di shop scheduling

10. Il modello $Jm||C_{\max}$

10.3 Euristica Shifting Bottleneck (continua)

Iter. 3: $C_{\max}(M_0) = 28$ pari alla lunghezza del cammino critico di $G(M_0)$.



Determiniamo la macchina da schedulare

Sia per la macchina 3 che per la 4 risolvendo $1|r_j|L_{\max}$ si ha $L^*_{\max} = 0$, con le sequenze di job $(2, 1)$ sulla macchina 3 e $(2, 3)$ sulla macchina 4.

Scegliamo la macchina 3. $M_0 = \{1, 2, 3\}$.

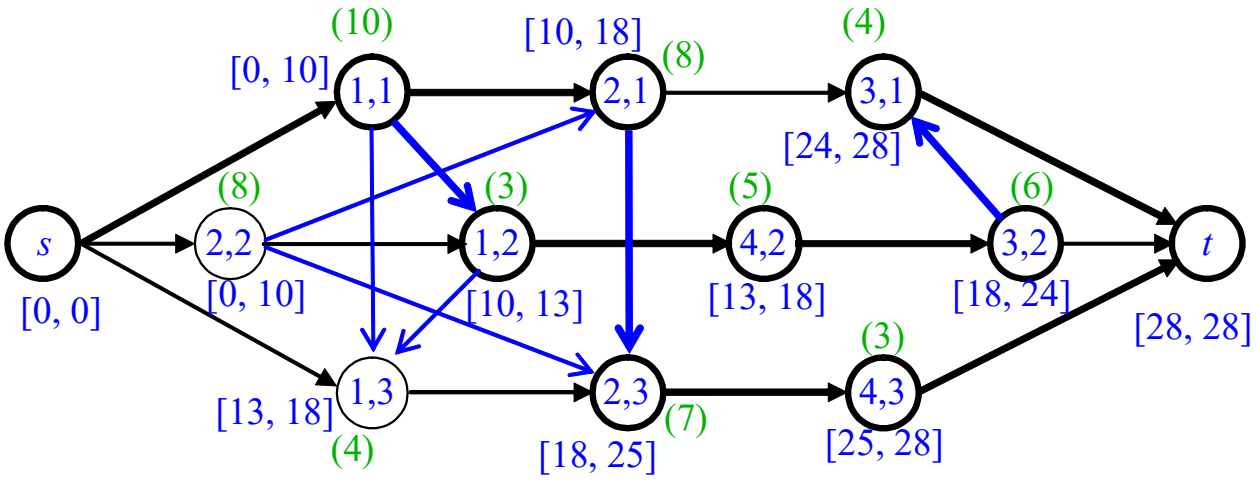
Sequenziamo i job sulla 3 secondo $(2, 1)$

Cercando di risequenziare le macchine 1 e 2 non si ottiene alcun miglioramento.

10. Il modello $Jm||C_{\max}$

10.3 Euristica Shifting Bottleneck (continua)

Iter. 4: $C_{\max}(M_0) = 28$ pari alla lunghezza del cammino critico di $G(M_0)$.



È rimasta da schedulare solo la macchina 4.

- Macchina 4 $\Rightarrow$ δ_4

  Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	2	3
p_j	5	3
r_j	13	25
d_j	18	28

La sequenza ottima è $(2, 3)$ con $L^*_{\max} = 0$. $\Rightarrow \delta_4 = 0$

Modelli di shop scheduling

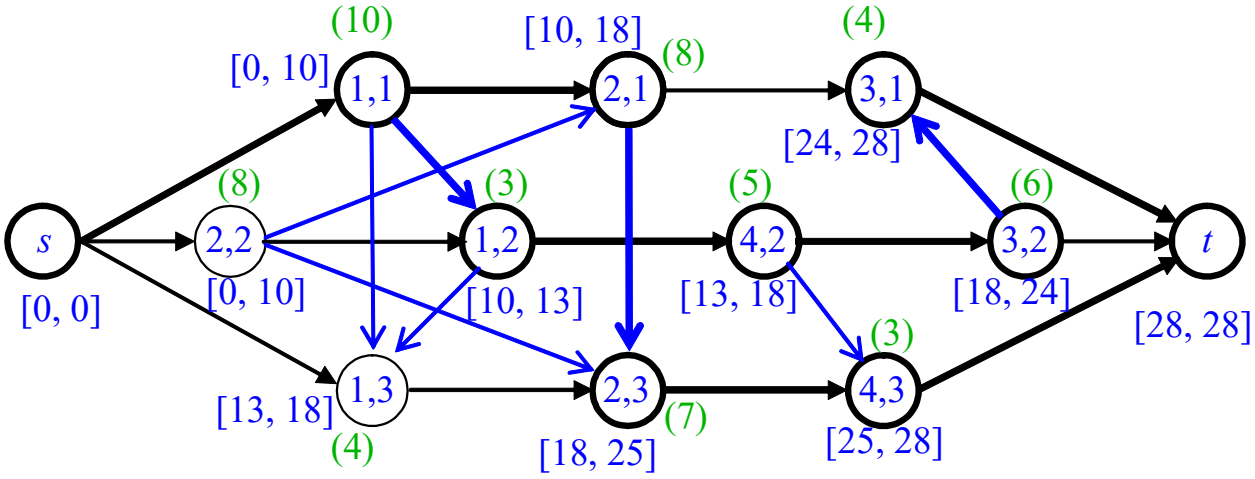
10. Il modello $Jm||C_{\max}$

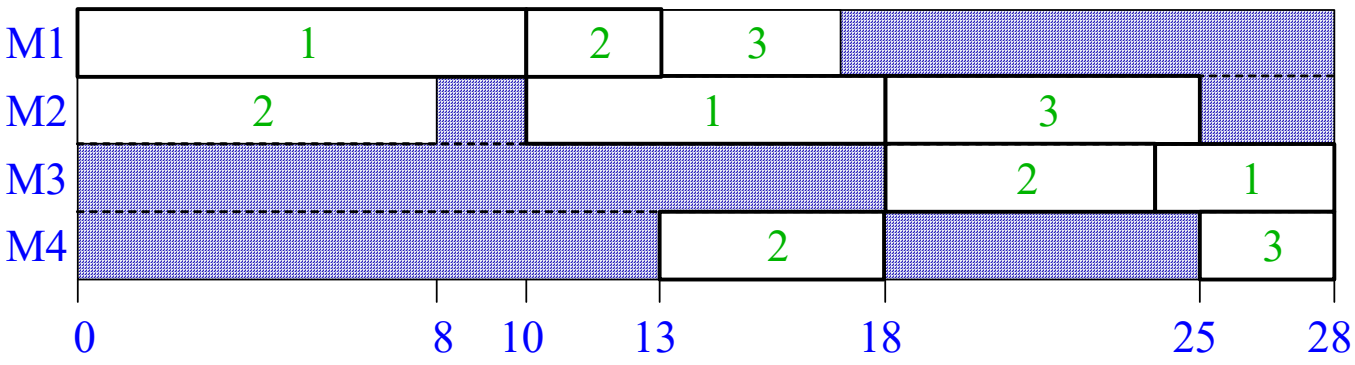
10.3 Euristica Shifting Bottleneck (continua)

L'euristica *shifting bottleneck* ha prodotto le seguenti *sequenze di job sulle macchine*

macchina	sequenza di job
1	(1, 2, 3)
2	(2, 1, 3)
3	(2, 1)
4	(2, 3)

Alla soluzione corrispondono i seguenti orientamento aciclico del grafo disgiuntivo e schedula di lunghezza 28.





La soluzione euristica nel caso specifico è anche ottima, perché di pari makespan a quella ottenuta con il B&B appresso discusso.

Modelli di shop scheduling

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound

Risolviamo il problema in modo esatto con il *B&B*

- Impiega uno *schema di branching* basato sulla *generazione* (*implicita*) di tutte le schedule attive.
- Ogni *nodo* dell'albero di ricerca definisce l'orientamento di un sottoinsieme di archi disgiuntivi, cioè una *schedula parziale* (*attiva*).
- Ogni *branch* da un nodo corrisponde alla *selezione di un task* $T_{i^*j} \in \Omega'$ (con $\Omega' \subseteq \Omega$, ove Ω è l'insieme dei task non schedulati i cui predecessori sono stati schedulati) dove Ω' è l'insieme dei task T_{i^*j} che richiedono *la macchina* i^* che *si libererebbe per prima* dopo aver eseguito uno dei task in Ω, e tale che T_{i^*j} risulta pronto prima di quando questa si libererebbe.
- Ciò implica l'orientamento degli archi $(T_{i^*j} \rightarrow T_{i^*h})$, per tutte le operazioni T_{i^*h} che devono ancora essere processate su i^*, e la creazione di un nuovo nodo nell'albero di ricerca, corrispondente alla schedula parziale ottenuta aggiungendo T_{i^*j} alla precedente.
- Per ogni nodo generato si calcola il lower bound LB2 sul valore della migliore schedula che completa quella parziale. Anche se per calcolare LB2 occorre risolvere m problemi strongly NP-hard, sperimentalmente l'uso di LB2 aumenta l'efficienza del B&B.

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

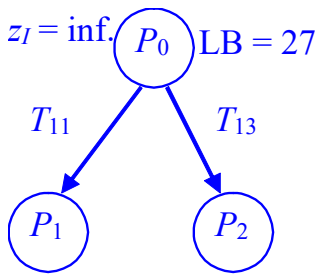
Esempio

Riconsideriamo l'esempio precedente.
Per motivi didattici non calcoliamo inizialmente un UB, quindi assumiamo $z_I = \infty$.

Il nodo P_0 dell'albero di ricerca corrisponde alla schedula parziale vuota. Il relativo sottoproblema ha $\mathrm{LB}(P_0) = \mathrm{LB2} = 27$.

L'insieme Ω dei task non schedulati i cui predecessori sono stati schedulati è $\Omega = \{T_{11}, T_{22}, T_{13}\}$. Essendo $t(\Omega) = \min_{T_{ij} \in \Omega}\{r_{ij} + p_{ij}\} = \min\{0 + 10, 0 + 8, 0 + 4\} = 4$ il tempo in cui terminerebbe per primo uno dei task di Ω e $i^* = 1$ la macchina che si libererebbe conseguentemente per prima, segue che $\Omega' = \Omega_{i^*} = \Omega_1 = \{T_{11}, T_{13}\}$ (T_{11} e T_{13} richiedono la macchina $i^* = 1$ e sono pronti prima dell'istante 4 in cui questa si libererebbe).

I possibili branch da P_0 sono quindi 2 corrispondenti a schedulare rispettivamente i task T_{11} e T_{13}.



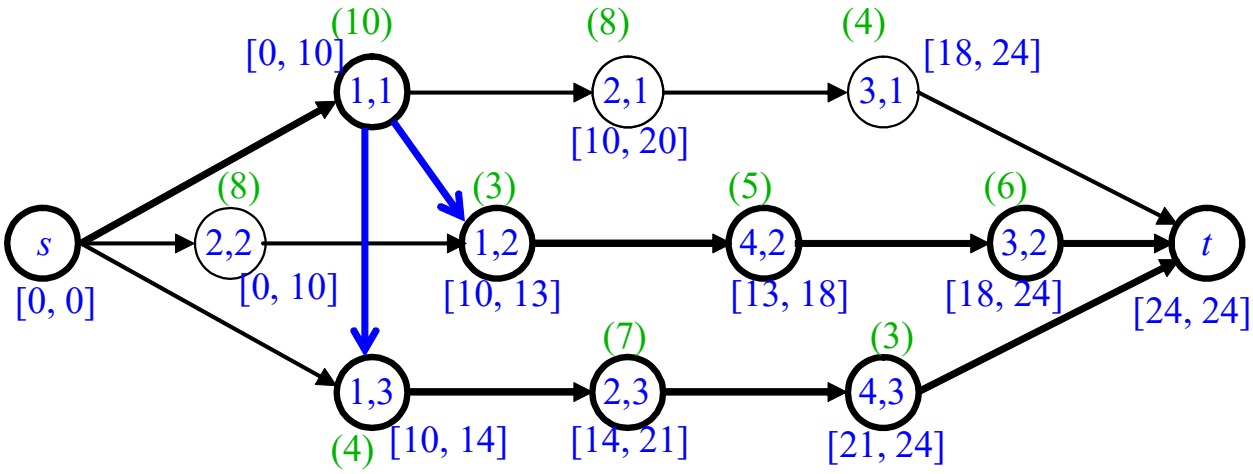
10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

Esaminiamo il sottoproblema relativo al nodo P_1.

Siccome T_{11} viene schedulato, consideriamo gli archi disgiuntivi orientati $(T_{11} \rightarrow T_{12})$, $(T_{11} \rightarrow T_{13})$.

Il sottoproblema relativo ha $\mathrm{LB1}(P_1) = 24$ pari alla lunghezza del cammino critico del grafo disgiuntivo depurato degli archi non orientati.



Calcoliamo $\mathrm{LB2}(P_1) = \mathrm{LB1}(P_1) + \max_i(\delta_i)$.

- Macchina 1 $\Rightarrow$ δ_1

  Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	2	3
p_j	3	4
r_j	10	10
d_j	13	14

La sequenza ottima è $(2, 3)$ con $L^*_{\max} = 3$. $\Rightarrow$ $\delta_1 = 3$

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

- Macchina 2 $\Rightarrow$ δ_2

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	2	3
p_j	8	8	7
r_j	10	0	14
d_j	20	10	21

La sequenza ottima è $(2, 1, 3)$ con $L^*_{\max} = 4$. $\Rightarrow \delta_2 = 4$

- Macchina 3 $\Rightarrow$ δ_3

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	2
p_j	4	6
r_j	18	18
d_j	24	24

La sequenza ottima è $(1, 2)$ con $L^*_{\max} = 4$. $\Rightarrow$ $\delta_3 = 4$

- Macchina 4 $\Rightarrow$ δ_4

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	2	3
p_j	5	3
r_j	13	21
d_j	18	24

La sequenza ottima è $(2, 3)$ con $L^*_{\max} = 0$. $\Rightarrow$ $\delta_4 = 0$

Pertanto $\mathrm{LB}(P_1) = \mathrm{LB2}(P_1) = \mathrm{LB1}(P_1) + \max(3, 4, 4, 0) = 24 + 4 = 28$.

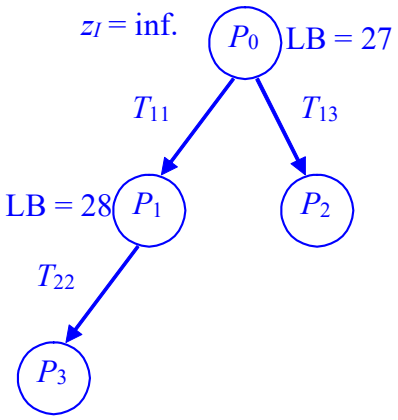
10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

Consideriamo ora il branching dal nodo P_1.

L'insieme Ω dei task non schedulati i cui predecessori sono stati schedulati è $\Omega = \{T_{21}, T_{22}, T_{13}\}$. Essendo $t(\Omega) = \min_{T_{ij} \in \Omega}\{r_{ij} + p_{ij}\} = \min\{10 + 8, 0 + 8, 10 + 4\} = 8$ il tempo in cui terminerebbe per primo uno dei task di Ω e $i^* = 2$ la macchina che si libererebbe conseguentemente per prima, segue che $\Omega' = \{T_{22}\}$ (T_{22} richiede la macchina $i^* = 2$ ed è l'unico pronto prima dell'istante 8 in cui questa si libererebbe).

C'è quindi un solo possibile branch da P_1 corrispondente a schedulare il task T_{22}.



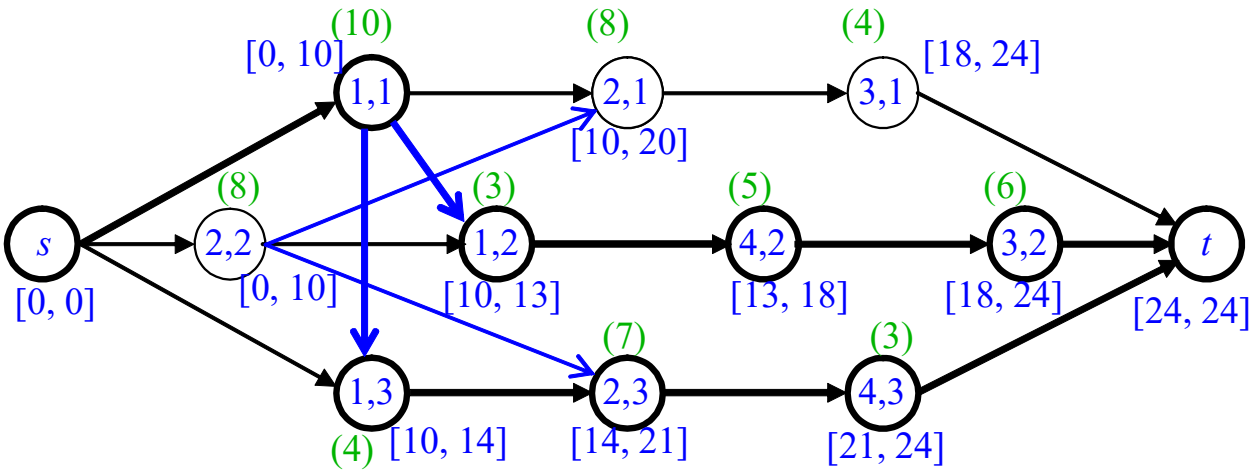
Esaminiamo il sottoproblema relativo al nodo P_3.

Siccome T_{22} viene schedulato, consideriamo gli archi disgiuntivi orientati $(T_{22} \rightarrow T_{21})$, $(T_{22} \rightarrow T_{23})$.

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

Il sottoproblema relativo ha $LB1(P_3) = 24$ pari alla lunghezza del cammino critico del grafo disgiuntivo depurato degli archi non orientati



Calcoliamo $LB2(P_3) = 24 + \max_i(\delta_i)$.

- Macchina 1 $\Rightarrow$ δ_1

  Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	2	3
p_j	3	4
r_j	10	10
d_j	13	14

  La sequenza ottima è $(2, 3)$ con $L^*_{\max} = 3$. $\Rightarrow$ $\delta_1 = 3$

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

- Macchina 2 $\Rightarrow$ δ_2

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	3
p_j	8	7
r_j	10	14
d_j	20	21

La sequenza ottima è $(1, 3)$ con $L^*_{\max} = 4$. $\Rightarrow$ $\delta_2 = 4$

- Macchina 3 $\Rightarrow$ δ_3

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	2
p_j	4	6
r_j	18	18
d_j	24	24

La sequenza ottima è $(1, 2)$ con $L^*_{\max} = 4$. $\Rightarrow$ $\delta_3 = 4$

- Macchina 4 $\Rightarrow$ δ_4

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	2	3
p_j	5	3
r_j	13	21
d_j	18	24

La sequenza ottima è $(2, 3)$ con $L^*_{\max} = 0$. $\Rightarrow$ $\delta_4 = 0$

Pertanto $\mathrm{LB}(P_3) = \mathrm{LB2}(P_3) = \mathrm{LB1}(P_3) + \max(3,4,4,0) = 24 + 4 = 28$.

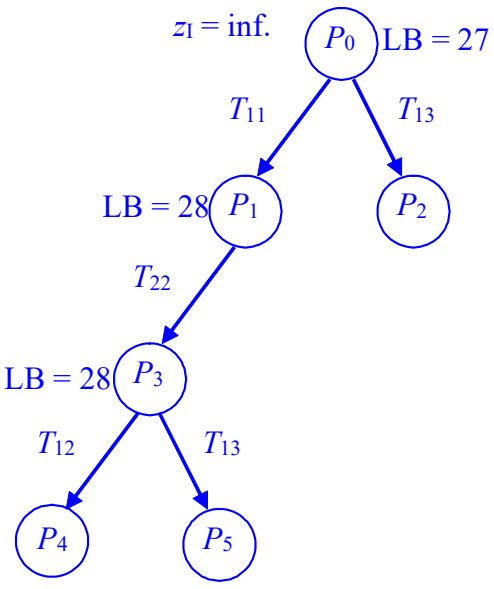
10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

Consideriamo ora il branching dal nodo P_3.

L'insieme Ω dei task non schedulati i cui predecessori sono stati schedulati è $\Omega = \{T_{21}, T_{12}, T_{13}\}$. Essendo $t(\Omega) = \min_{T_{ij} \in \Omega}\{r_{ij} + p_{ij}\} = \min\{10 + 8, 10 + 3, 10 + 4\} = 13$ il tempo in cui terminerebbe per primo uno dei task di Ω e $i^* = 1$ la macchina che si libererebbe conseguentemente per prima, segue che $\Omega' = \{T_{12}, T_{13}\}$ (entrambi richiedono la macchina $i^* = 1$ e sono pronti prima dell'istante 13 in cui questa si libererebbe).

Ci sono due possibili branch da P_3 corrispondenti a schedulare i task T_{12}, T_{13}.



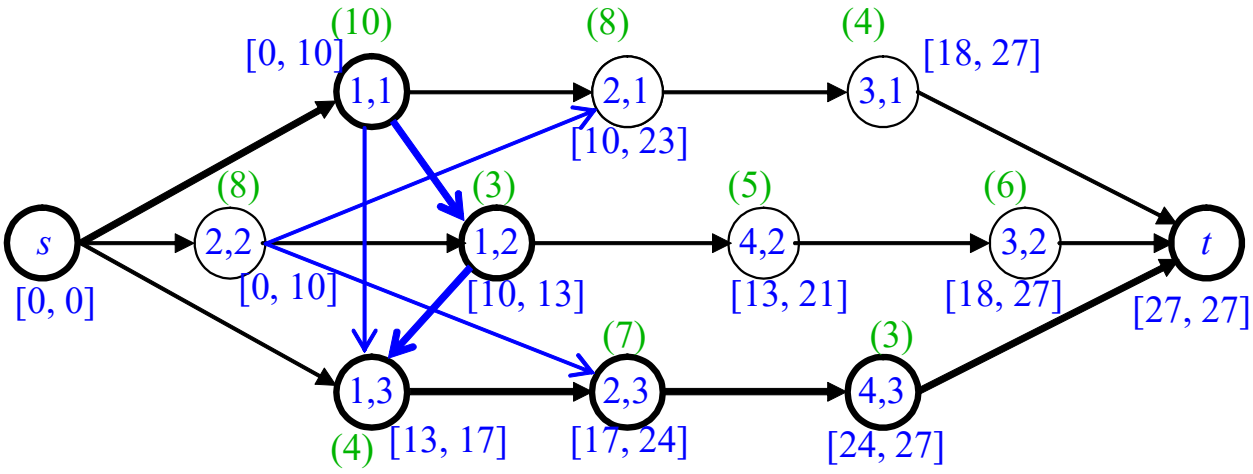
10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

Esaminiamo il sottoproblema relativo al nodo P_4.

Siccome T_{12} viene schedulato, consideriamo l'ulteriore arco disgiuntivo orientati $(T_{12} \rightarrow T_{13})$.

Il sottoproblema relativo ha $\mathrm{LB1}(P_4) = 27$ pari alla lunghezza del cammino critico del grafo disgiuntivo depurato degli archi non orientati



Calcoliamo $\mathrm{LB2}(P_4) = 27 + \max_i(\delta_i)$.

- Macchina 1 già schedulata.

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

- Macchina 2 $\Rightarrow$ δ_2

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	3
p_j	8	7
r_j	10	17
d_j	23	24

La sequenza ottima è $(1, 3)$ con $L^*_{\max} = 1$. $\Rightarrow$ $\delta_2 = 1$

- Macchina 3 $\Rightarrow$ δ_3

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	2
p_j	4	6
r_j	18	18
d_j	27	27

La sequenza ottima è $(1, 2)$ con $L^*_{\max} = 1$. $\Rightarrow$ $\delta_3 = 1$

- Macchina 4 $\Rightarrow$ δ_4

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	2	3
p_j	5	3
r_j	13	24
d_j	21	27

La sequenza ottima è $(2, 3)$ con $L^*_{\max} = 0$. $\Rightarrow$ $\delta_4 = 0$

Pertanto $\text{LB}(P_4) = \text{LB2}(P_4) = \text{LB1}(P_4) + \max(-, 1, 1, 0) = 27 + 1 = 28$.

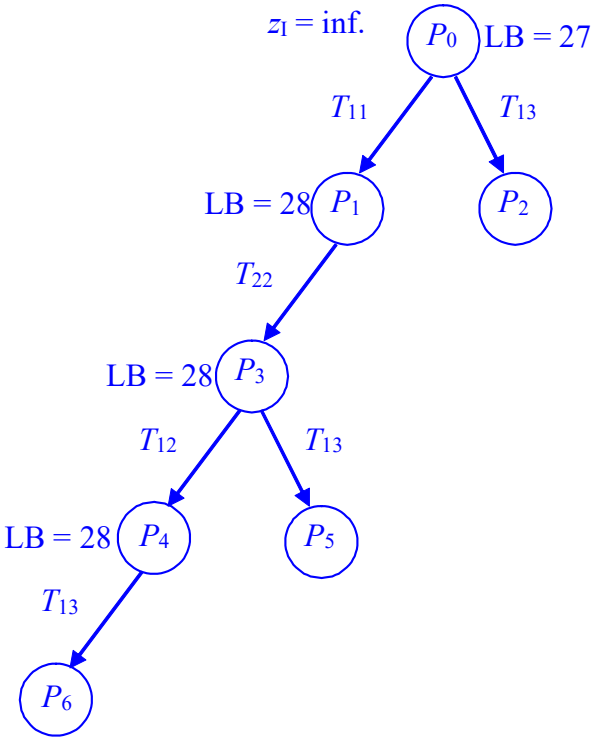
10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

Consideriamo ora il branching dal nodo P_4.

L'insieme Ω dei task non schedulati i cui predecessori sono stati schedulati è $\Omega = \{T_{21}, T_{42}, T_{13}\}$. Essendo $t(\Omega) = \min_{T_{ij} \in \Omega}\{r_{ij} + p_{ij}\} = \min\{10 + 8, 13 + 5, 13 + 4\} = 17$ il tempo in cui terminerebbe per primo uno dei task di Ω e $i^* = 1$ la macchina che si libererebbe conseguentemente per prima, segue che $\Omega' = \{T_{13}\}$ (solo T_{13} richiede la macchina $i^* = 1$ ed è pronto prima dell'istante 17 in cui questa si libererebbe).

C'è un solo branch da P_4 corrispondenti a schedulare T_{13}.



Modelli di shop scheduling

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

Esaminiamo il sottoproblema relativo al nodo P_6.

Siccome T_{13} viene schedulato, ma non si aggiungono orientandoli ulteriori archi disgiuntivi, abbiamo che $\mathrm{LB}(P_6) = \mathrm{LB}(P_4) = 28$.

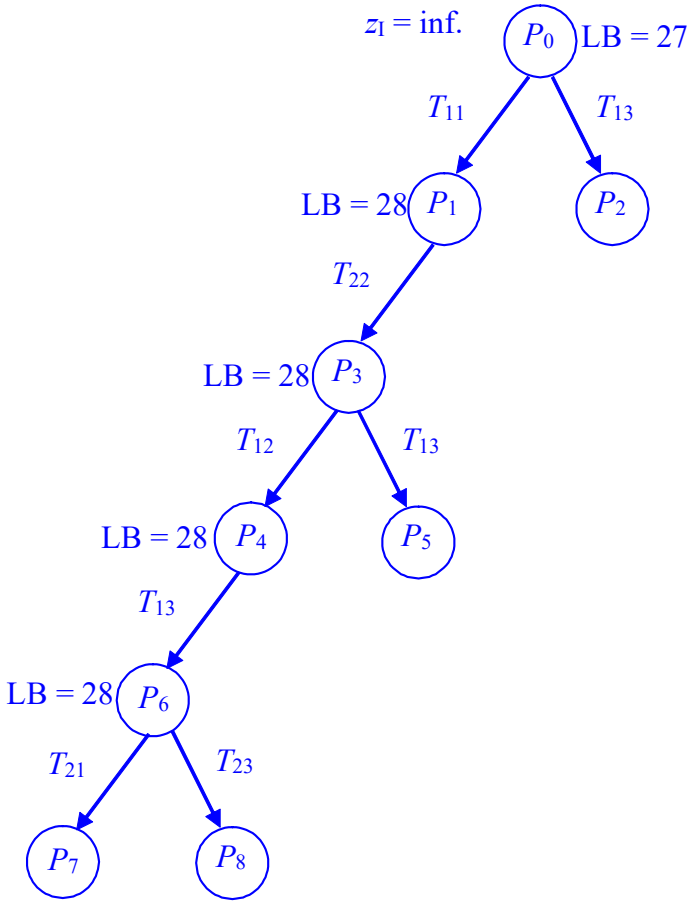
Consideriamo ora il branching dal nodo P_6.

L'insieme Ω dei task non schedulati i cui predecessori sono stati schedulati è $\Omega = \{T_{21}, T_{42}, T_{23}\}$. Essendo $t(\Omega) = \min_{T_{ij} \in \Omega}\{r_{ij} + p_{ij}\} = \min\{10 + 8, 13 + 5, 17 + 7\} = 18$ il tempo in cui terminerebbe per primo uno dei task di Ω e $i^* = 2$ la macchina che si libererebbe conseguentemente per prima, segue che $\Omega' = \{T_{21}, T_{23}\}$ (sia T_{21} che T_{23} richiedono la macchina $i^* = 2$ ed entrambi sono pronti prima dell'istante 18 in cui la macchina si libererebbe).

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

Ci sono due branch da P_6 corrispondenti a schedulare rispettivamente T_{21} e T_{23}.



Esaminiamo il sottoproblema relativo al nodo P_7.

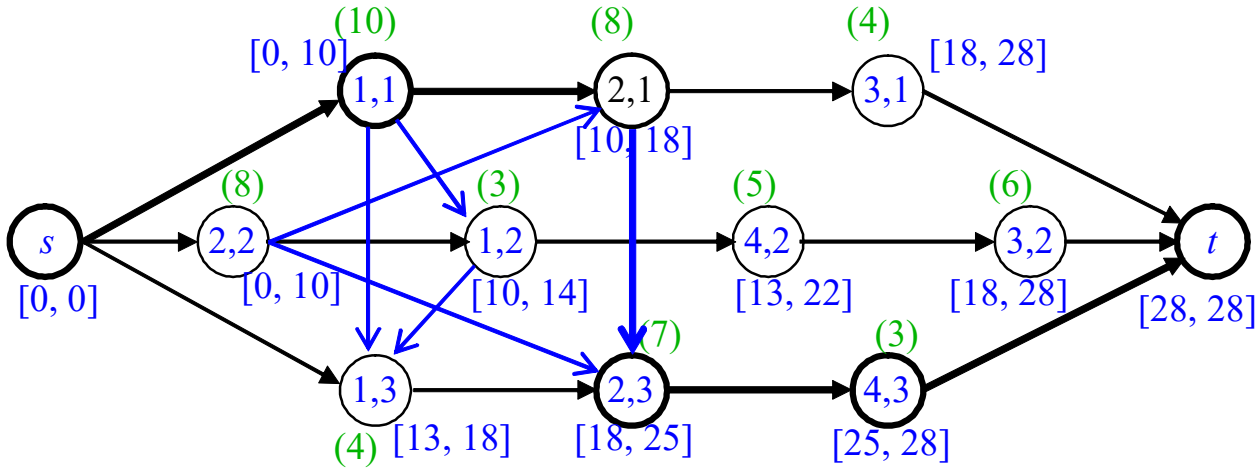
Siccome T_{21} viene schedulato, consideriamo l'ulteriore arco disgiuntivo orientato $(T_{21} \rightarrow T_{23})$.

Il sottoproblema relativo ha $\mathrm{LB1}(P_7) = 28$ pari alla lunghezza del cammino critico del grafo disgiuntivo depurato degli archi non orientati

Modelli di shop scheduling

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)



Calcoliamo $\mathrm{LB2}(P_7) = 28 + \max_i(\delta_i)$.

- Macchine 1 e 2 già schedulate.
- Macchina 3 $\Rightarrow$ δ_3

  Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	2
p_j	4	6
r_j	18	18
d_j	28	28

  La sequenza ottima è $(1, 2)$ con $L^*_{\max} = 0$. $\Rightarrow$ $\delta_3 = 0$
- Macchina 4 $\Rightarrow$ δ_4

  Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	2	3
p_j	5	3
r_j	13	25
d_j	22	28

  La sequenza ottima è $(2, 3)$ con $L^*_{\max} = 0$. $\Rightarrow$ $\delta_4 = 0$

10. Il modello $Jm||C_{\max}$

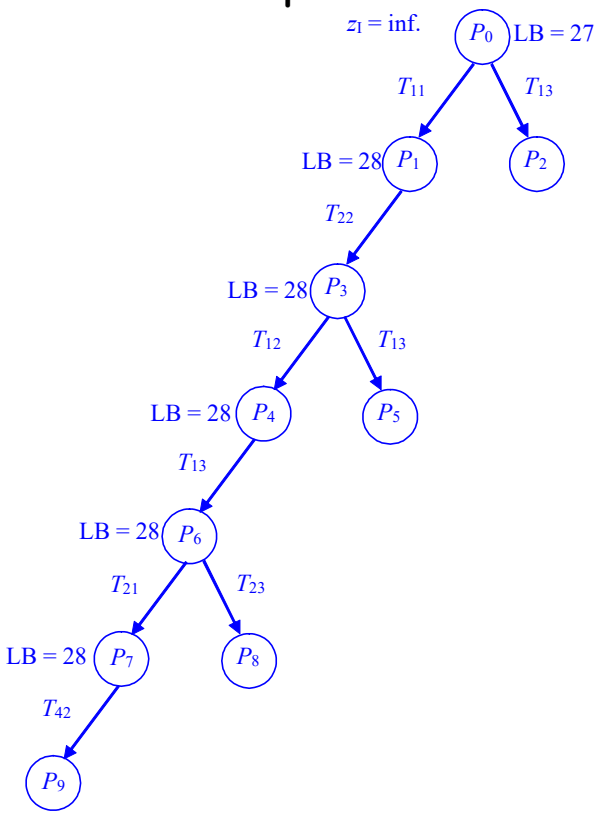
10.4 Algoritmo di Branch&Bound (continua)

$\mathrm{LB}(P_7) = \mathrm{LB2}(P_7) = \mathrm{LB1}(P_7) + \max(-, -, 0, 0) = 28 + 0 = 28$.

Consideriamo ora il branching dal nodo P_7.

L'insieme Ω dei task non schedulati i cui predecessori sono stati schedulati è $\Omega = \{T_{31}, T_{42}, T_{23}\}$. Essendo $t(\Omega) = \min_{T_{ij} \in \Omega}\{r_{ij} + p_{ij}\} = \min\{18 + 4, 13 + 5, 18 + 7\} = 18$ il tempo in cui terminerebbe per primo uno dei task di Ω e $i^* = 4$ la macchina che si libererebbe conseguentemente per prima, segue che $\Omega' = \{T_{42}\}$ (solo T_{42} richiede la macchina $i^* = 4$ ed è pronto prima dell'istante 18 in cui questa si libererebbe).

Un solo branch da P_7 corrispondente a schedulare T_{42}.



10. Il modello $Jm||C_{\max}$

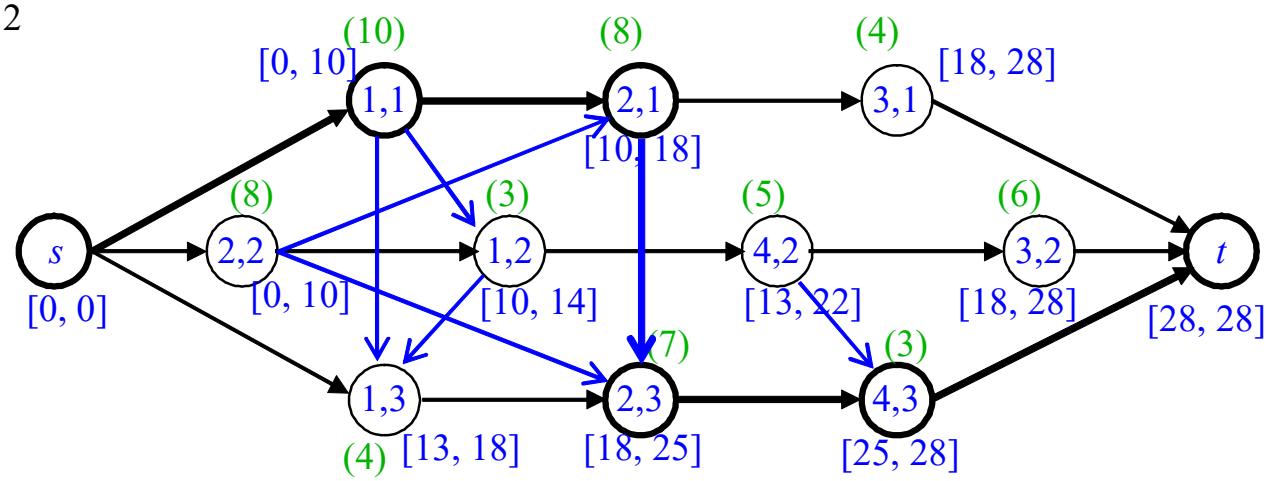
10.4 Algoritmo di Branch&Bound (continua)

Esaminiamo il sottoproblema relativo al nodo P_9.

Siccome T_{42} viene schedulato, consideriamo l'ulteriore arco disgiuntivo orientato $(T_{42} \rightarrow T_{43})$.

Il sottoproblema relativo ha $\mathrm{LB1}(P_9) = 28$ pari alla lunghezza del cammino critico del grafo disgiuntivo depurato degli archi non orientati

2



Calcoliamo $\mathrm{LB2}(P_9) = 28 + \max_i(\delta_i)$.

- Macchine 1, 2 e 4 già schedulate.
- Macchina 3 $\Rightarrow$ δ_3

  Risolviamo la seguente istanza di $1|r_j|L_{\max}$

job	1	2
p_j	4	6
r_j	18	18
d_j	28	28

La sequenza ottima è $(1, 2)$ con $L^*_{\max} = 0$. $\Rightarrow$ $\delta_3 = 0$

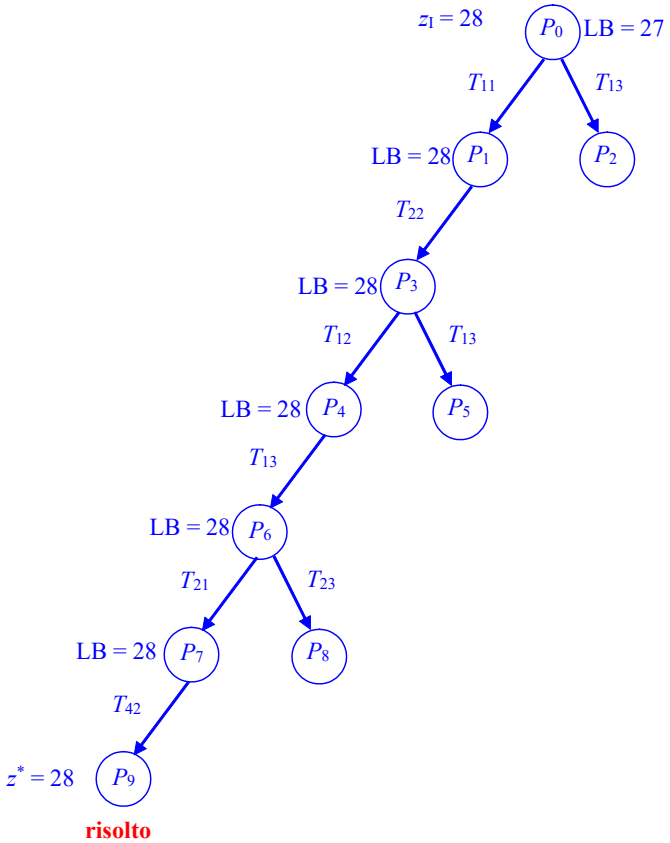
10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

Pertanto $\mathrm{LB}(P_9) = \mathrm{LB2}(P_9) = \mathrm{LB1}(P_9) + \max(-, -, 0, -) = 28 + 0 = 28$.

Si noti che con l'aggiunta dell'arco $(T_{3,1} \rightarrow T_{3,2})$, suggerita dal calcolo di $\mathrm{LB2}(P_9)$, il grafo contiene tutti gli archi disgiuntivi orientati in modo che il grafo è completamente orientato aciclicamente e quindi la *soluzione* fornita dal calcolo di $\mathrm{LB2}(P_9)$ è ammissibile e quindi *ottima* per P_9. Quindi P_9 è *risolto* con $z^*(P_9) = 28$.

Poniamo quindi $z_I = 28$.

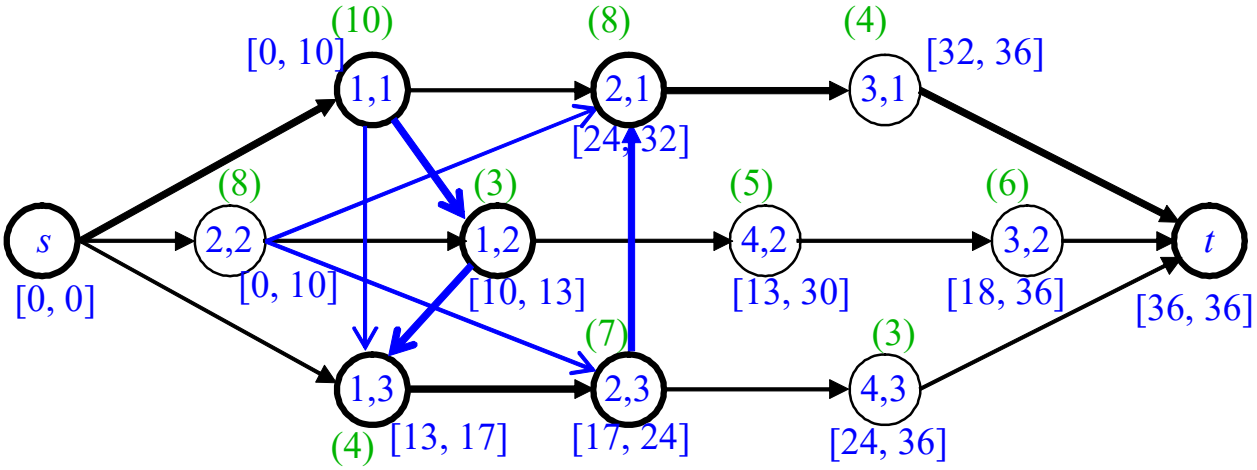


10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

Facciamo backtracking ed esaminiamo il sottoproblema relativo al nodo P_8.

Siccome T_{23} viene schedulato (oltre a T_{11}, T_{22}, T_{12} e T_{13}) il grafo parzialmente orientato comprensivo dell'arco disgiuntivo orientato $(T_{23} \rightarrow T_{21})$ è il seguente.



Il sottoproblema relativo ha $\mathrm{LB1}(P_8) = 36$.

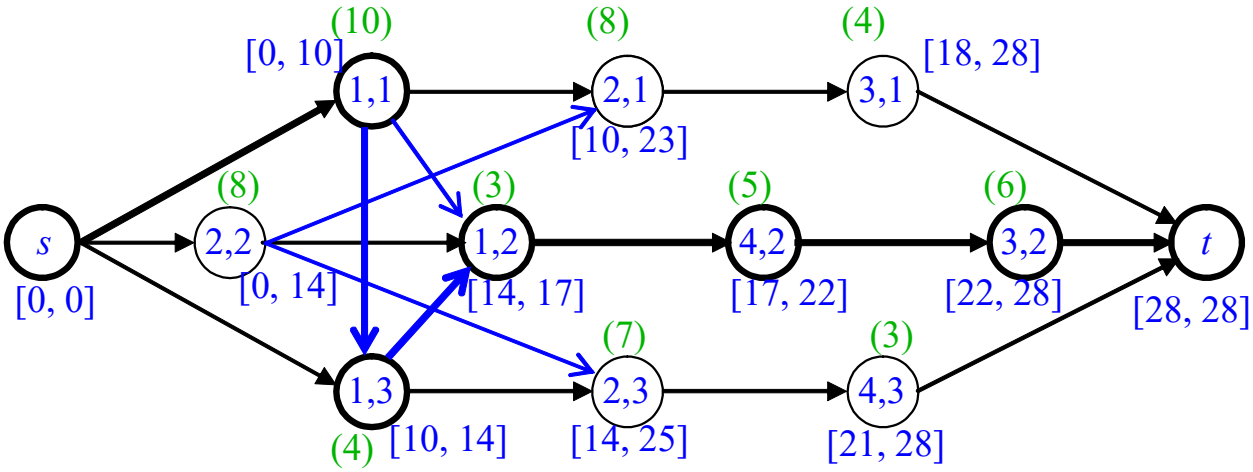
Siccome $\mathrm{LB1}(P_8) \geq z_I = 28$, P_8 è dominato e il nodo dichiarato *scandagliato*.

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

Facciamo backtracking ed esaminiamo il sottoproblema relativo al nodo P_5.

Siccome T_{13} viene schedulato (oltre a T_{11} e T_{22}) il grafo parzialmente orientato comprensivo dell'arco disgiuntivo orientato $(T_{13} \rightarrow T_{12})$ è il seguente.

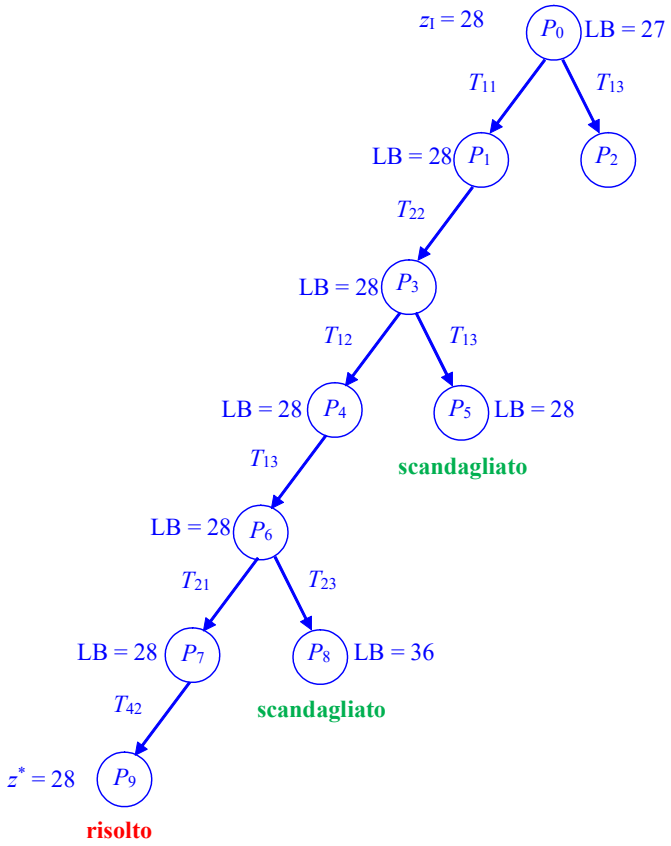


Il sottoproblema relativo ha $\mathrm{LB1}(P_5) = 28$.

Siccome $\mathrm{LB1}(P_5) \geq z_I = 28$, P_5 è dominato e il nodo dichiarato *scandagliato*.

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)



Facciamo backtracking ed esaminiamo il sottoproblema relativo al nodo P_2.

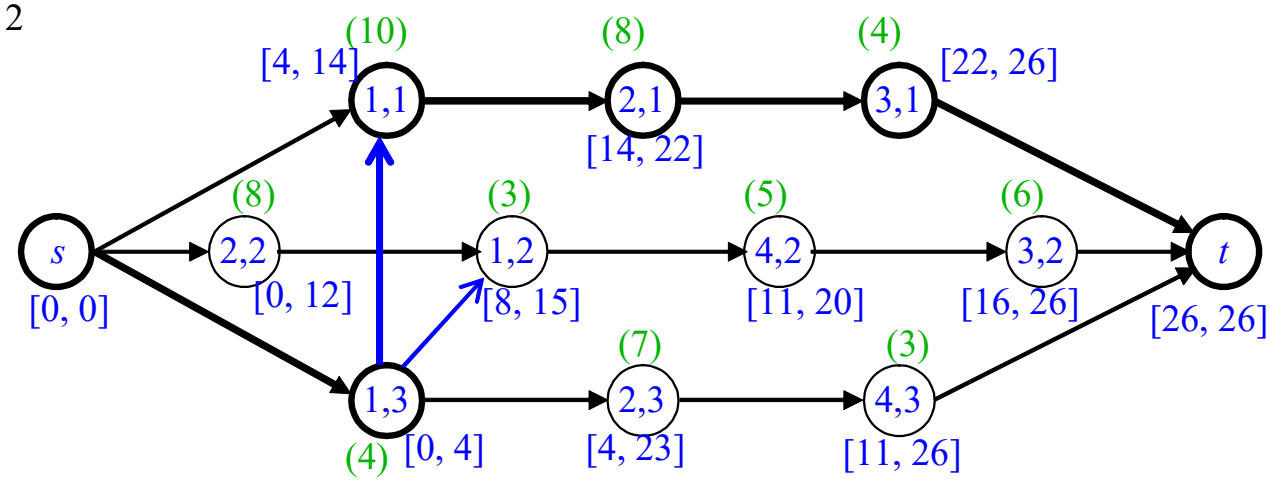
Siccome solo T_{13} viene schedulato il grafo parzialmente orientato contiene gli archi disgiuntivi orientati $(T_{13} \rightarrow T_{11})$ e $(T_{13} \rightarrow T_{12})$.

Modelli di shop scheduling

9. Il modello $Jm||C_{\max}$: Algoritmo di B&B (continua)

Il grafo corrispondente è

2



Il sottoproblema relativo ha $\mathrm{LB1}(P_2) = 26$.

Calcoliamo $\mathrm{LB2}(P_2) = \mathrm{LB1}(P_2) + \max_i(\delta_i)$.

- Macchina 1 $\Rightarrow$ δ_1

  Risolviamo la seguente istanza di $1|r_j|L_{\max}$

Job	1	2
p_j	10	3
r_j	4	8
d_j	14	15

  La sequenza ottima è $(1, 2)$ con $L^*_{\max} = 2$. $\Rightarrow$ $\delta_1 = 2$

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

- Macchina 2 $\Rightarrow$ δ_2

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

Job	1	2	3
p_j	8	8	7
r_j	14	0	4
d_j	22	12	23

La sequenza ottima è $(2, 3, 1)$ con $L^*_{\max} = 1$. $\Rightarrow \delta_2 = 1$

- Macchina 3 $\Rightarrow$ δ_3

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

Job	1	2
p_j	4	6
r_j	22	16
d_j	26	26

La sequenza ottima è $(2, 1)$ con $L^*_{\max} = 0$. $\Rightarrow$ $\delta_3 = 0$

- Macchina 4 $\Rightarrow$ δ_4

Risolviamo la seguente istanza di $1|r_j|L_{\max}$

Job	2	3
p_j	5	3
r_j	11	11
d_j	20	26

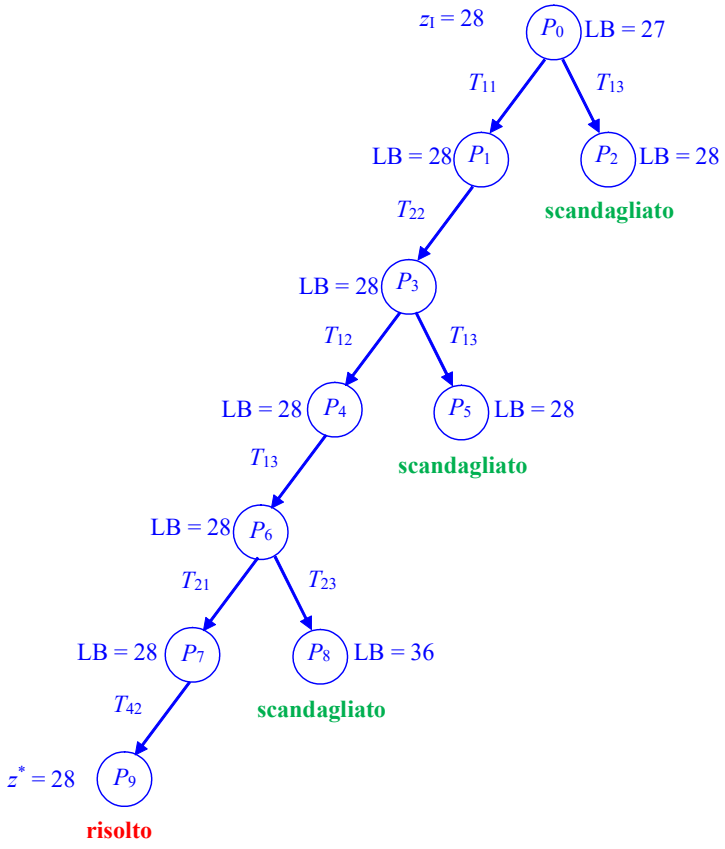
La sequenza ottima è $(2, 3)$ con $L^*_{\max} = -7$. $\Rightarrow$ $\delta_4 = 0$

Pertanto $LB(P_2) = LB2(P_2) = LB1(P_2) + \max(2,1,0,0) = 26 + 2 = 28$.

10. Il modello $Jm||C_{\max}$

10.4 Algoritmo di Branch&Bound (continua)

Siccome $\mathrm{LB1}(P_2) \geq z_I = 28$, P_2 è dominato e il nodo dichiarato *scandagliato*.



Facciamo backtracking. Non ci sono più nodi generati da esaminare. Stop.

Modelli di shop scheduling

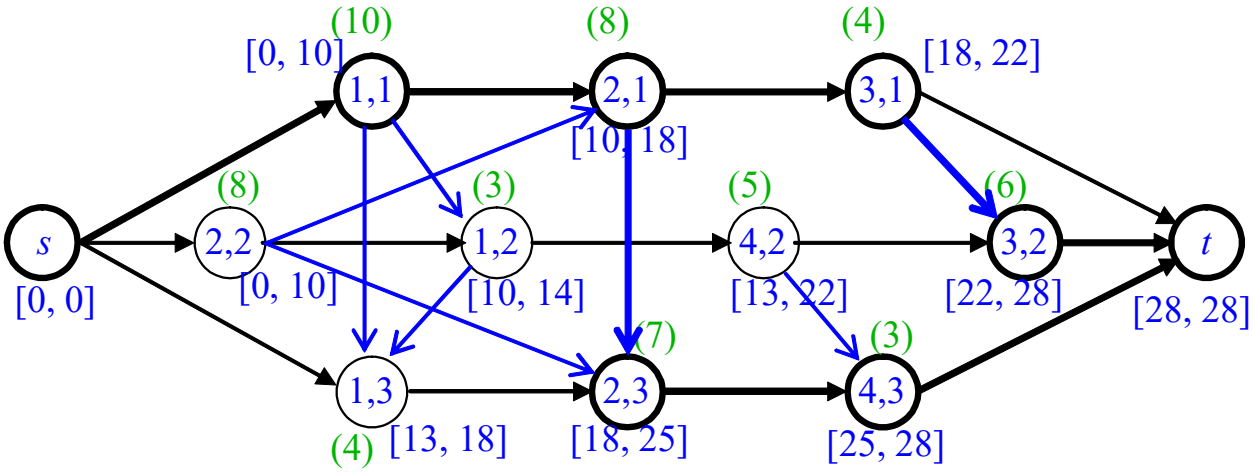
10. Il modello $Jm||C_{max}$

10.4 Algoritmo di Branch&Bound (continua)

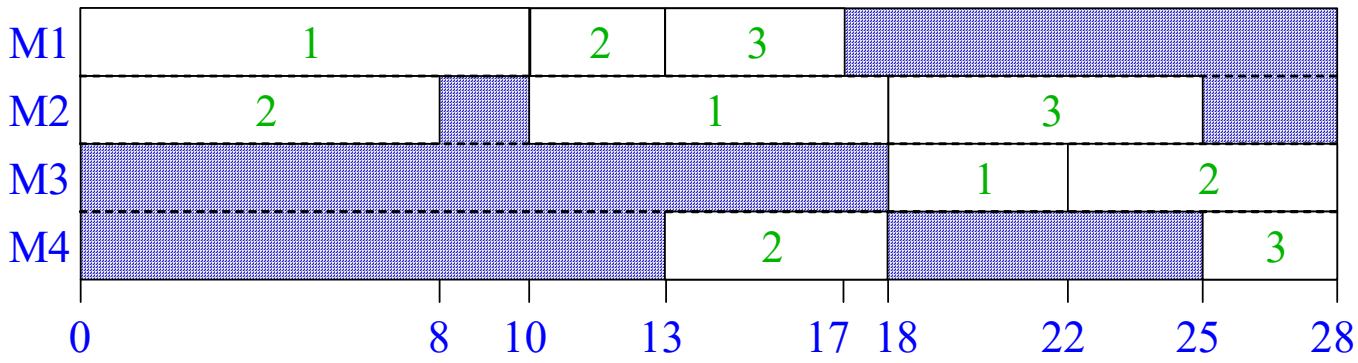
Al termina del B&B, la *soluzione ottima* è quella associata al nodo P_9 rappresentata dalle seguenti *sequenze di job sulle macchine*:

$\sigma_1 = (1, 2, 3)$; $\sigma_2 = (2, 1, 3)$; $\sigma_3 = (1, 2)$; $\sigma_4 = (2, 3)$.

e corrispondente al seguente *orientamento aciclico del grafo disgiuntivo* con cammini critici di lunghezza 28



dal quale si ricava la seguente schedula al più presto (earliest start) di lunghezza (makespan) 28.



Modelli di shop scheduling

11. Note finali sulla complessità

- La seguente tabella riassume i principali risultati sulla complessità dei problemi di shop scheduling.

Problema	Complessità		Note	Riferimenti
$O2\|\|C_{\max}$	P	$O(n)$	LAPT $O(n \log n)$	Gonzalez, Sahni (1976)
$O2\|\|\Sigma C_j$	NP-h			Achugbe, Chin (1982)
$O3\|\|C_{\max}$	NP-h			Gonzalez, Sahni (1976)
$Om\|n=2\|C_{\max}$	P	$O(m)$		Gonzalez, Sahni (1976)
$Om\|n=2\|\gamma\_reg$	P	$O(m)$		Shakhlevich, Strusevich (1993)
$Om\|n=3\|C_{\max}$	NP-h			Gonzalez, Sahni (1976)
$Om\|n=3\|\Sigma C_j$	NP-h			Sotskov, Shakhlevich (1995)
$F2\|\|C_{\max}$	P	$O(n \log n)$	Johnson's rule	Johnson (1954)
$F2\|\|\Sigma C_j$	NP-h			Du, Leung (1993)
$F3\|\|C_{\max}$	NP-h			Garey et al. (1976)
$Fm\|n=2\|C_{\max}$	P	$O(m \log m)$		Sotskov (1991)
$Fm\|n=2\|\gamma\_reg$	P	$O(m \log m)$		Sotskov (1991)
$Fm\|n=3\|C_{\max}$	NP-h			Sotskov (1991)
$Fm\|n=3\|\Sigma C_j$	NP-h			Sotskov (1991)
$J2\|\|C_{\max}$	P	$O(n \log n)$	two ind. $F2\|\|C_{\max}$	Johnson (1954)
$J2\|\|\Sigma C_j$	NP-h			Du, Leung (1993)
$J3\|\|C_{\max}$	NP-h			Garey et al. (1976)
$Jm\|n=2\|C_{\max}$	P	$O(r^2 \log r)$	r: num. op. che usano la stessa macchina	Brucker (1988)
$Jm\|n=2\|\gamma\_reg$	P	$O(r^2 \log r)$		Sotskov (1991)
$Jm\|n=3\|C_{\max}$	NP-h			Sotskov, Shakhlevich (1995)
$Jm\|n=3\|\Sigma C_j$	NP-h			Sotskov, Shakhlevich (1995)