Parallel identical processor scheduling with weighted completion time

A column generation approach

Simone Cavana
219833{at}studenti.unimore.it

25 febbraio 2021

$P||\sum w_j C_j$

Introduzione

M = set di m macchine identiche

J = set di n job

p_j = tempo d'esecuzione di un job

w_j = peso di un job

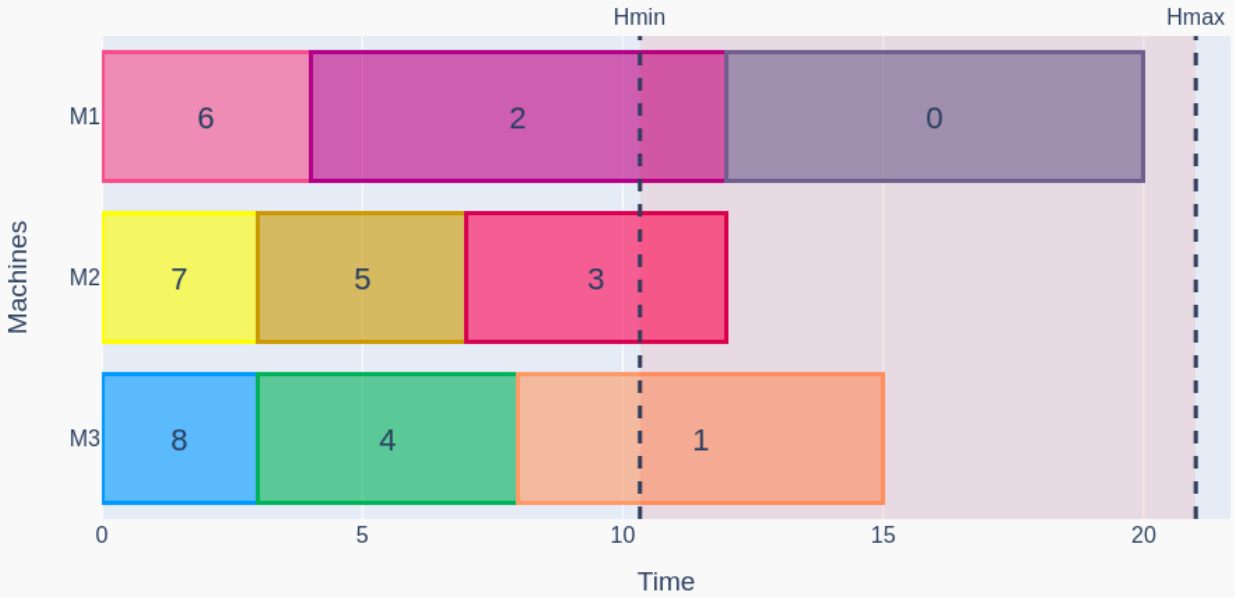
C_j = tempo di completamento di un job in una schedula

- Ogni macchina è disponibile dall'istante 0 e può elaborare al più un job per istante;
- Non è ammessa preemption;
- I job devono essere eseguiti in modo contiguo (no idle).

$P||\sum w_j C_j$

Esempio

job	w_j	p_j
1	1	8
2	4	7
3	7	8
4	5	5
5	5	5
6	6	4
7	7	4
8	7	3
9	9	3



- $H_{min} = (\sum_{j \in J} p_j - (m-1)p_{max})/m$
- $H_{max} = (\sum_{j \in J} p_j + (m-1)p_{max})/m$
- $p_{max} = \max_{j \in J} p_j$

Set-covering formulation

- Definiamo **schedula** $s \in S$, un insieme di job ammissibili assegnabile ad una qualsiasi macchina m;
- L'**ordinamento di Smith** ci indica di effettuare un ordinamento in ordine decrescente sulla base del rapporto w_j/p_j per avere la condizione di ottimalità.

$$a_{js} = \begin{cases} 1 & \text{se il job j è assegnato alla schedula s,} \\ 0 & \text{altrimenti} \end{cases}$$

$$C_j(s) = \sum_{k=1}^{j} a_{ks} p_k$$

$$c_s = \sum_{j \in J} w_j a_{js} C_j(s) = \sum_{j \in J} w_j a_{js} \left[\sum_{k=1}^{j} a_{ks} p_k \right]$$

$$x_s = \begin{cases} 1 & \text{se la schedula s è selezionata,} \\ 0 & \text{altrimenti} \end{cases}$$

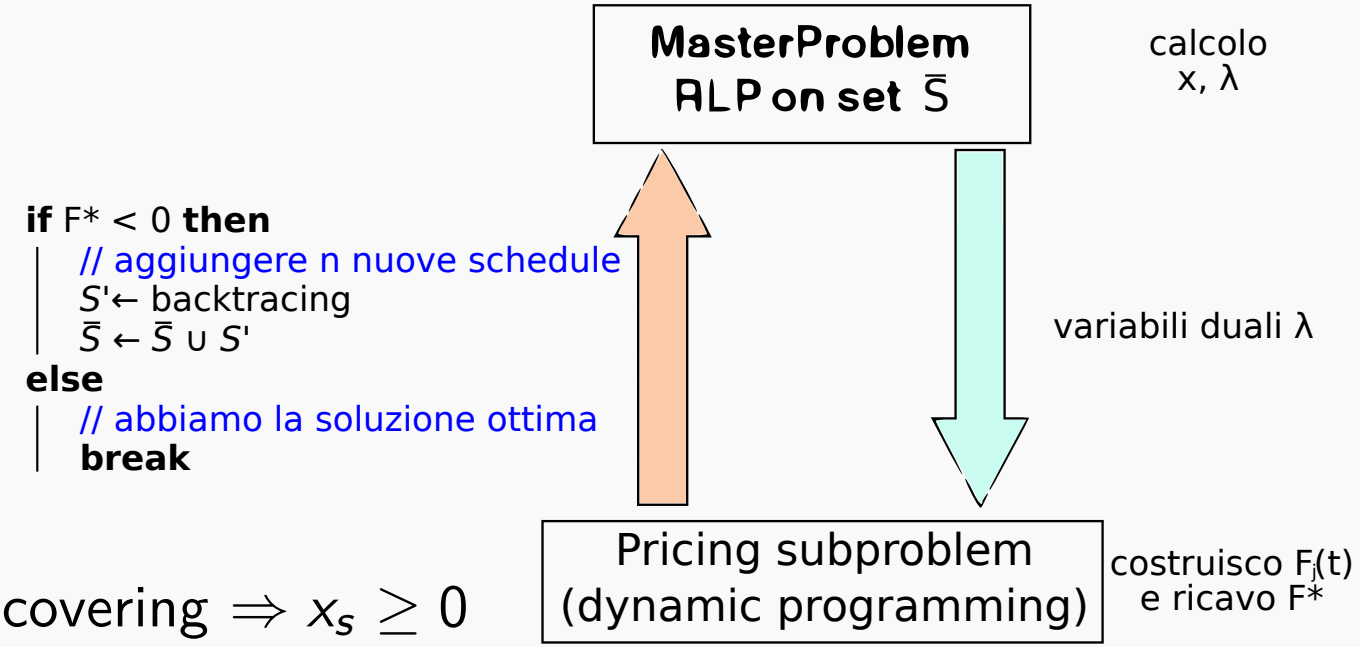
Set-covering model

$$
\begin{aligned}
\min \quad & \sum_{s \in S} c_s x_s & & \\
\text{s.t.} \quad & \sum_{s \in S} x_s = m & & (1) \\
& \sum_{s \in S} a_{js} x_s = 1, \quad & j \in J \quad & (2) \\
& x_s \in \{0,1\}, \quad & s \in S \quad & (3)
\end{aligned}
$$

Costi ridotti (si ricavano dal duale):

$$
\bar{c}_s = c_s - \sum_{j \in J} a_{js} \lambda_j = \sum_{j \in J} \left[w_j \left(\sum_{k=1}^{j} a_{ks} p_k \right) - \lambda_j \right] a_{js}
$$

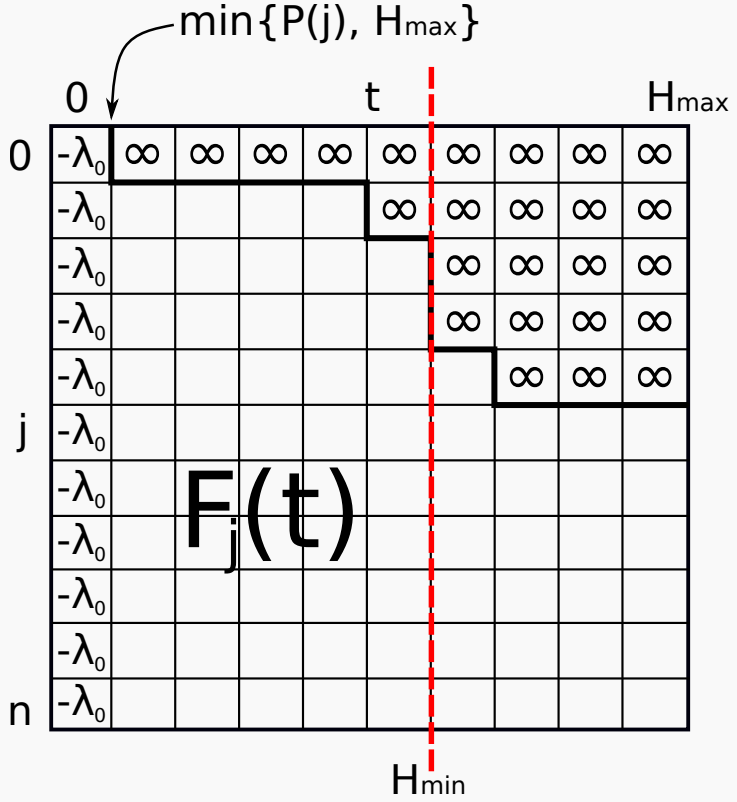
Column Generation Approach



- Rilasso set-covering $\Rightarrow x_s \geq 0$
- Risolvo RLP su $\bar{S}$
- Ricavo le variabili duali:
  - λ_0 è costante, relativa ad (1)
  - λ_j relative a (2)
- **se** $F^* < 0 \Rightarrow$ aggiungo le n schedule con costi ridotti più negativi tramite backtracing su $F_j(t)$

Pricing algorithm

Programmazione dinamica



$$P(j) = \sum_{k=1}^{j} p_k$$

Inizializzazione:

$$F_j(t) = \begin{cases} -\lambda_0 & \text{se } j = 0, \text{ e } t = 0, \\ \infty & \text{altrimenti} \end{cases}$$

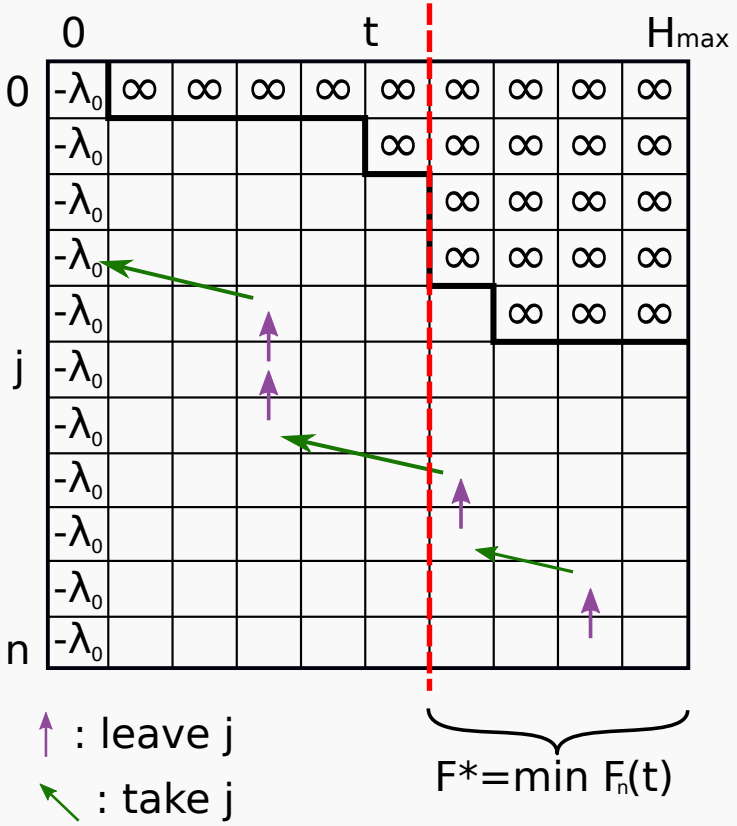
per i successivi step $j = 1, \ldots, n,\ t = 0, \ldots, \min\{P(j), H_{max}\}$:

$$F_j(t) = \begin{cases} \min\{F_{j-1}(t), F_{j-1}(t - p_j) + w_j t - \lambda_j\} & \text{se } r_j + p_j \leq t \leq d_j \\ F_{j-1}(t) & \text{altrimenti} \end{cases}$$

$$F^* = \min_{H_{min} \leq t \leq H_{max}} F_n(t)$$

Pricing algorithm

Backtracing



- $F_j(t)$ rappresenta i costi ridotti minimi dati dalla schedula composta dai job $\leq j$ che termina in t
- Le nuove schedule da inserire in $\bar{S}$ sono quelle associate ad $F_n(t)$ più negative
- La strategia per risalire alla schedula dato il costo ridotto $F_j(t)$ è di ripercorrere a ritroso la matrice dove:
  - salgo in verticale se $F_j(t) == F_{j-1}(t)$ e non inserisco j in $\bar{S}$
  - altrimenti salgo al job precedente e mi sposto nel tempo tanto quanto p_j, in modo da inserire j in $\bar{S}$

Randomized List Heuristic

Inizializzazione

- $\bar{S} \rightarrow$ schedule iniziali
- $2000 \leq N \leq 5000$
- Estrazione basata su c_s
- NS migliorativo finale

Algorithm 1: Heuristic

Input: n, m, w, p, N

Output: $\bar{S}$

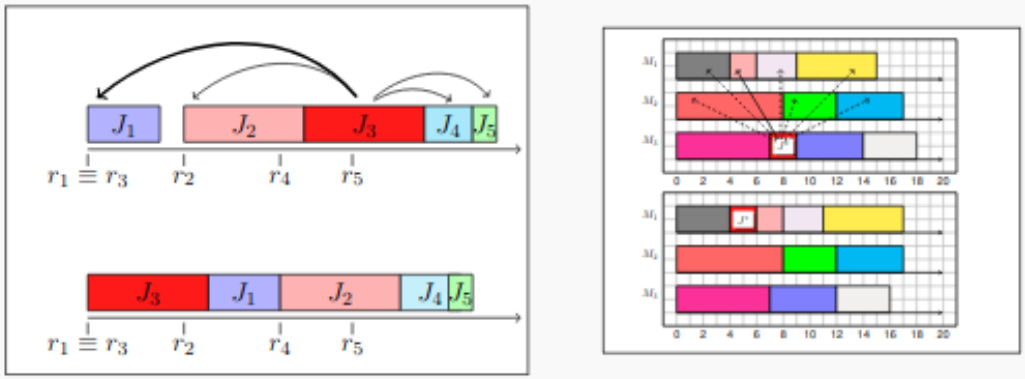
```
1 S̄, s ← {}
2 jobs ← smith_order(n, w, p)
3 n_iter ← 0
4 while n_iter < N do
5     s ← create_rand_sched(jobs)
6     S̄ ← S̄ ∪ s
7     n_iter + +
8 S̄ ← extract_best(S̄, 10)
9 return neighborhood_search(S̄)
```

Randomized List Heuristic

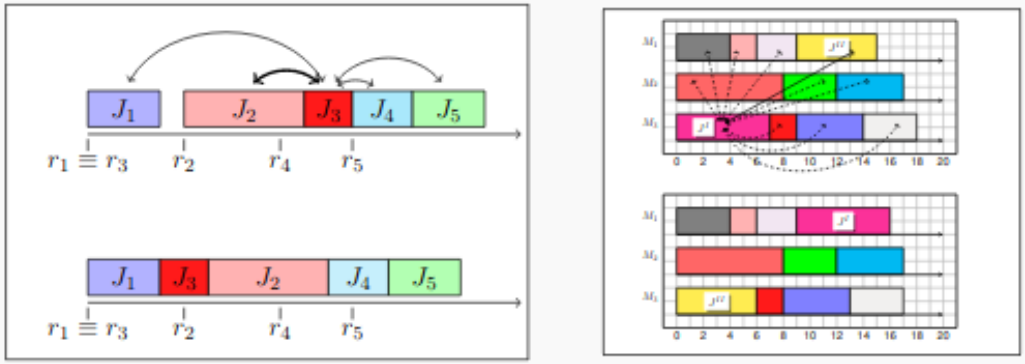
Neighborhood Search

- Insert: spostare un job da una macchina ad un altra;
- Swap: scambiare due job schedulati su macchine differenti;

- **Insert move**: move one "object" to another "position"



- **SWAP move**: exchange two "objects"



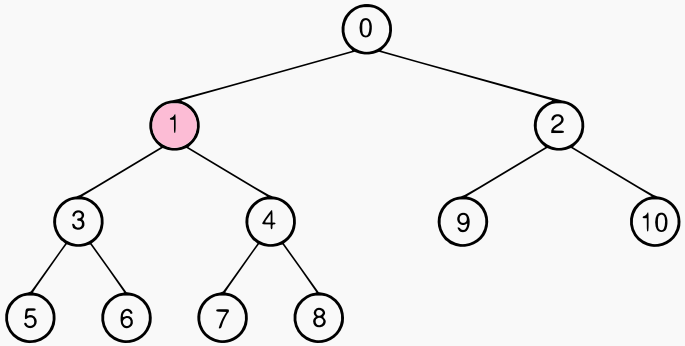
Integralità della soluzione

- Caso speciale frazionario che rispetta **Teorema 1**
  - $C_j(s) = C_j \; \forall \; j \in J$ and $\forall \; s$ with $x_s^* > 0$
- Branch & Bound con LB dato da x^*, soluzione ottima di RLP
  1. branching sugli **intervalli d'esecuzione** $[r_j, d_j]$
     - $\exists \; j \; t.c. \sum_{s \in S^*} C_j(s) x_s^* > \min\{C_j(s), s \in S^*\} \equiv C_j^{min}$

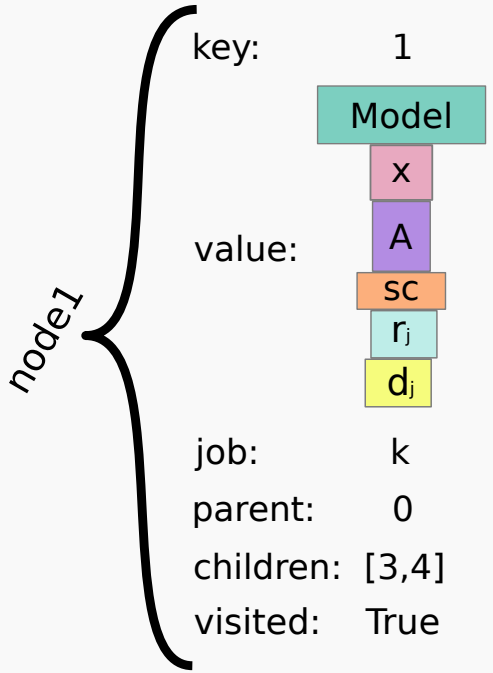
Left	**Right**
$C_j(s) \leq C_j^{min}$	$C_j(s) \geq C_j^{min} + 1$
$d_j \leftarrow C_j^{min}$	$r_j \leftarrow C_j^{min} + 1 - p_j$
$d_k \leftarrow \min\{d_k, d_j - p_j + p_k\}$	$r_k \leftarrow \max\{r_k, r_j\}$
$\forall J_k \in \mathcal{P}_j$	$\forall J_k \in \mathcal{S}_j$

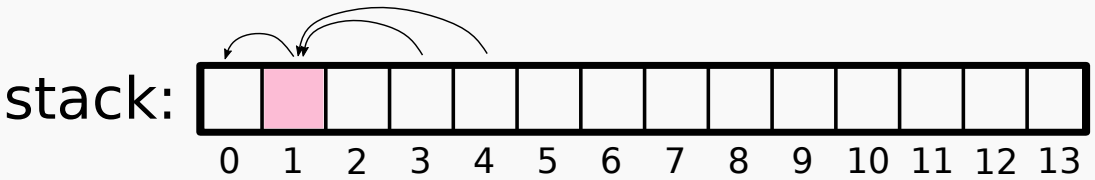
Branch & Bound

Feasibility and Details



- Schedules:
  - **Left**: $max(C_j - d_j) < 0$
  - **Right**: $min((C_j - p_j) - r_j) > 0$
- Nodes:
  - $\forall j \in J, \quad d_j - r_j \geq p_j$





Benchmark

- 15 istanze “semplici” **fornite** da Barnes & Brennan;
- Istanze generate **random** con:
  - m estratto da [3,4,5];
  - n da [20, 30, 40, 50];
  - tempi d’esecuzione e pesi:
    1. p_j ricavati da [1, 10], w_j da [10, 100];
    2. sia p_j che w_j estratti da una distribuzione normale fra [1, 100];
    3. sia p_j che w_j estratti da una distribuzione normale fra [1, 20];

Risultati

Barnes & Brennan benchmark

m	n	HTime	CGTime	NB	MNN	MGAP	ILP	TOT
2	5	0.75	0.00	57	0	0	57	57
	8	1.00	0.11	27	0	0	27	27
	10	1.73	2.33	27	1	0	33	33
	12	1.82	0.24	30	1	0	33	33
3	6	0.89	0.00	28	0	0	28	28
	7	0.65	0.00	17	0	0	17	17
	9	1.68	0.52	21	2	0	25	25
	11	1.76	0.18	15	2	0	17	17
	13	2.60	8.55	10	2	0	20	20
	15	2.76	0.34	29	0	0	29	29
	20	4.19	0.89	26	2	0	36	36
	25	5.83	0.46	20	3	0	24	24
4	20	4.06	0.75	32	0	0	32	32
5	9	1.93	0.07	30	0	0	30	30

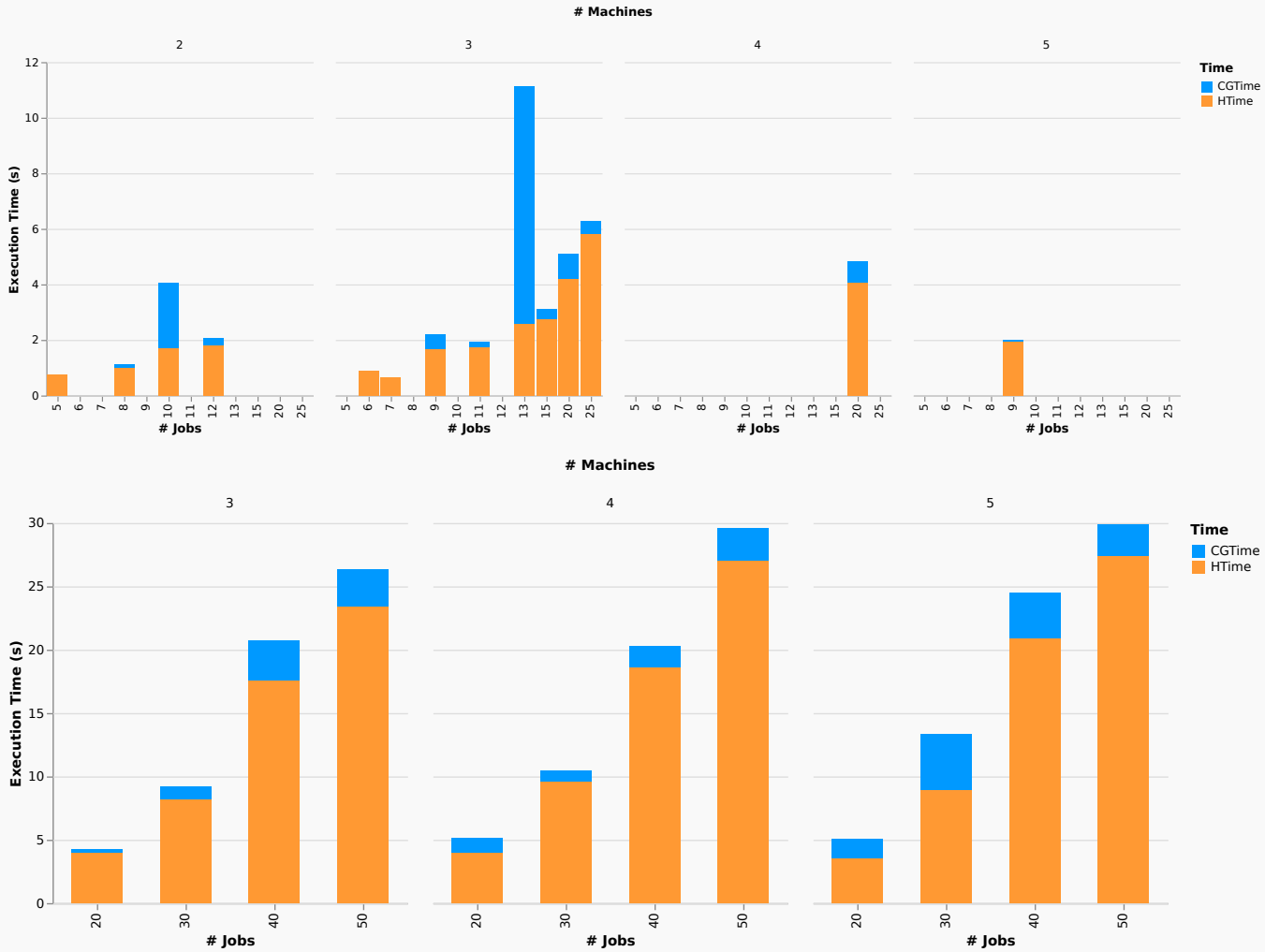
Risultati

Random benchmark

m	n	HTime	CGTime	NB	MNN	MGAP	ILP	TOT
3	20	4.03	0.22	35	1	0	36	36
	30	8.21	1.03	34	0	0	34	34
	40	17.62	3.14	29	0	0	29	29
	50	23.41	2.95	37	0	0	37	37
4	20	4.03	1.13	36	1	0	38	38
	30	9.60	0.87	29	1	4	29	30
	40	18.67	1.64	33	0	0	33	33
	50	27.00	2.58	43	0	0	43	43
5	20	3.56	1.50	33	1	2	35	36
	30	8.95	4.40	14	7	16	18	20
	40	20.89	3.61	25	1	0	28	28
	50	27.43	2.45	44	0	0	44	44

Conclusioni

Analisi risultati



- Le performance dell'euristica sono correlate ad n;
- CG invece dipende molto da $\bar{S}$ e dall'utilizzo di B&B.

Conclusioni

Considerazioni e Sviluppi futuri

- Performance migliori per la maggior parte delle istanze;
- Ottimizzazione di pricing come descritto in appendice [1,2];
- Sviluppo Branching sull'immediato successore [4] ed eventuale paragone fra i B&B.

Bibliografia

[1] Marjan van den Akker, Han Hoogeveen, and Steef van de Velde. Parallel machine scheduling by column generation. Operations Research, 47(6):862–872, 1999.

[2] Marjan van den Akker, Han Hoogeveen, and Steef van de Velde. Applying Column Generation to Machine Scheduling, pages 303–330. Springer US, Boston, MA, 2005.

[3] J. Wesley Barnes and J. J. Brennan. An improved algorithm for scheduling jobs on identical machines. AIIE Transactions, 9(1):25–31, 1977.

[4] Zhi-Long Chen and Warren B. Powell. Solving parallel machine scheduling problems by column generation. INFORMS Journal on Computing, 11(1):78–94, 1999.