Parallel identical processor scheduling with weighted completion time

A column generation approach

Simone Cavana
219833{at}studenti.unimore.it

18 febbraio 2021

$P||\sum w_j C_j$

Introduzione

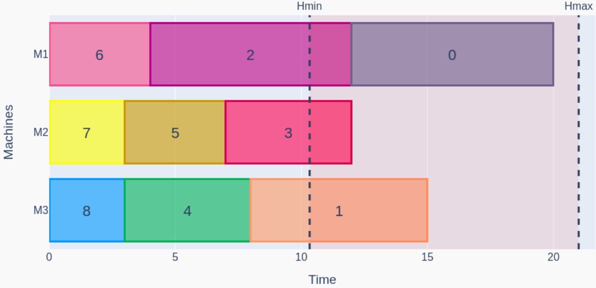
$$
\begin{aligned}
M &= \text{set di m macchine identiche} \\
J &= \text{set di n job} \\
p_j &= \text{tempo d'esecuzione di un job} \\
w_j &= \text{peso di un job} \\
C_j &= \text{tempo di completamento di un job in una schedula}
\end{aligned}
$$

- 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

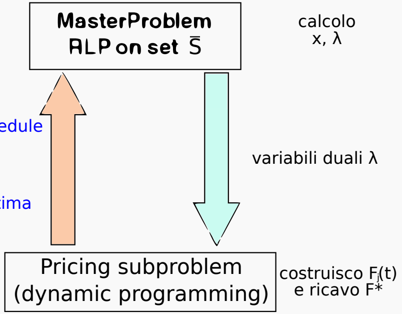
$$\begin{aligned} \min \quad & \sum_{s \in S} c_s x_s \\ \text{s.t.} \quad & \sum_{s \in S} x_s = m & (1) \\ & \sum_{s \in S} a_{js} x_s = 1, \quad j \in J & (2) \\ & x_s \in \{0,1\}, \qquad s \in S & (3) \end{aligned}$$

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

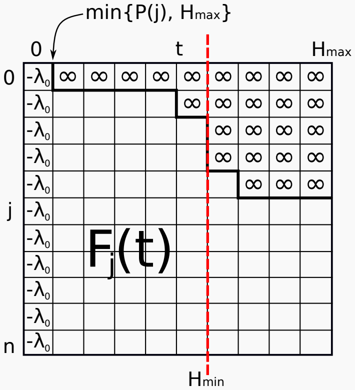
if F* < 0 **then**
  // aggiungere n nuove schedule
  $S' \leftarrow$ backtracing
  $\bar{S} \leftarrow \bar{S} \cup S'$
else
  // abbiamo la soluzione ottima
  break



- 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

Initialization

- $\bar{S} \rightarrow$ 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: $\bar{S}$

$\bar{S}, s \leftarrow \{\}$

$jobs \leftarrow$ smith_order(n, w, p)

$n\_iter \leftarrow 0$

while $n\_iter < N$ **do**

    $s \leftarrow$ create_rand_sched$(jobs)$

    $\bar{S} \leftarrow \bar{S} \cup s$

    $n\_iter + +$

$\bar{S} \leftarrow$ extract_best$(\bar{S}, 10)$

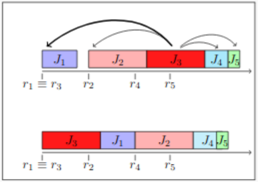
return neighborhood_search$(\bar{S})$

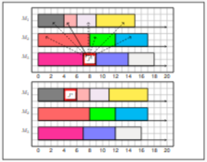
Randomized List Heuristic

Neighborhood Search

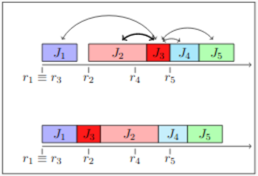
- Insert: spostare un job da una macchina ad un altra
- Swap: scambiare due job schedulati su macchine differenti

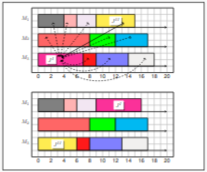
- **Insert move**: move one "object" to another "position"





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





Integralità della soluzione

Branch & Bound

- Caso speciale frazionario che rispetta **Teorema 1**
- Lower bound dato da x^*, soluzione ottima di RLP

1. Branching sugli **intervalli d'esecuzione**
   - $\exists\, j \ t.c. \sum_{s \in S^*} C_j(s) x_s^* > \min\{C_j(s), s \in S^*\} \equiv C_j^{min}$
   - $C_j(s) \leq C_j^{min}$ $\qquad$ $C_j(s) \geq C_j^{min} + 1$

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	NIter	ExecTime	GAP	NB&B
2	5	3	1,0	•	0
2	8	3,33	2,0	•	0
2	10	5	2,0	•	0
2	12	6,16	3,0	•	1
3	6	4	1,0	•	0
3	7	3	1,0	•	0
3	9	4,83	2,0	•	0
3	11	5	3,0	•	2
3	13	8,33	3,0	•	0
3	15	9,33	4,0	•	0
3	20	8,33	6,0	•	1
3	25	10,5	7,0	•	1
4	20	12	6,5	•	0
5	9	4,75	2,0	•	0

Risultati

Random benchmark

m	n	NIter	ExecTime	GAP	NB&B
3	20	•	•	•	•
3	30	•	•	•	•
3	40	•	•	•	•
3	50	•	•	•	•
4	20	•	•	•	•
4	30	•	•	•	•
4	40	•	•	•	•
4	50	•	•	•	•
5	20	•	•	•	•
5	30	•	•	•	•
5	40	•	•	•	•
5	50	•	•	•	•

Conclusioni

Bibliografia

- Marjan van den Akker, Han Hoogeveen, and Steef van de Velde. Parallel machine scheduling by column generation. Operations Research, 47(6):862–872, 1999.
- 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.
- J. Wesley Barnes and J. J. Brennan. An improved algorithm for scheduling jobs on identical machines. AIIE Transactions, 9(1):25–31, 1977.