Szeregowanie zadań

dr Hanna Furmańczyk

17-05-2018

Szeregowanie bez przestojów (maszyny dedykowane)

Zastosowania

- układanie planów zajęć
- przemysł hutniczy, metalurgiczny
- przemysł spożywczy
- ...

Rodzaje

- praca ciągła maszyn (*no-idle*)
- brak przestojów pomiedzy operacjami zadania (*no-wait*)

Warunek *no-wait*

Twierdzenie

Problem $F3|no - wait|C_{max}$ jest NP-trudny.

Twierdzenie

Istnieje algorytm o złożoności $O(n \log n)$ rozwiązujący problem $F2|no - wait|C_t$ (harmonogram cykliczny) oraz $F_2|no - wait|C_{\max}$.

Algorithm (Gilmore and Gomory 1964)

STEP 1. Number the jobs such that $p_{2,j} \leqslant p_{2,j+1}$, $j = 1, \dots, n-1$. Initialize $G_1 = G_2 = \emptyset$.

STEP 2. Find a function $\phi(j)$, $j = 1, \dots, n$, such that $p_{1,\phi(j)} \leqslant p_{1,\phi(j+1)}$, $j = 1, \dots, n-1$.

STEP 3. Define a graph with n nodes (each representing a job) and no edges. The lengths $C_{j,j+1}$ of edges $(j, j+1)$, $j = 1, \dots, n-1$ that may be added later are given by $C_{j,j+1} = \max\{0, (\min\{p_{2,j+1}, p_{1,\phi(j+1)}\} - \max\{p_{2,j}, p_{1,\phi(j)}\})\}$ for $j = 1, \dots, n-1$.

STEP 4. Set $j = 1$.

STEP 4.1. If the undirected edge $(j, \phi(j))$ is not in the graph and $j \neq \phi(j)$, add it. Set $j = j + 1$.

STEP 4.2. If $j \leqslant n$, go to Step 4.1.

STEP 5. If the graph has only one connected component, go to Step 7. Otherwise, let $k = \operatorname{argmin}\{C_{j,j+1} | j$ and $j + 1$ are in different components$\}$, breaking ties arbitrarily.

STEP 6. Add the undirected edge $(k, k+1)$ to the graph. If $p_{1,\phi(k)} \geqslant p_{2,k}$, set $G_1 = G_1 \cup \{(k, k+1)\}$. Otherwise set $G_2 = G_2 \cup \{(k, k+1)\}$. Go to Step 5.

STEP 7. If $G_1 = \emptyset$, let $s = 0$. Otherwise, let the elements of G_1 be $\{(r_1, r_1 + 1), \ldots, (r_s, r_s + 1)\}$, where $r_1 \geqslant \cdots \geqslant r_s$.

STEP 7.1. If $G_2 = \emptyset$, let $t = 0$. Otherwise, let the elements of G_2 be $\{(k_1, k_1 + 1), \ldots, (k_t, k_t + 1)\}$, where $k_1 \leqslant \cdots \leqslant k_t$.

STEP 8. Define for $1 \leqslant e, g, h \leqslant n$ a function $\alpha_{e,g}(h)$ as follows: $\alpha_{e,g}(e) = g$, $\alpha_{e,g}(g) = e$, and $\alpha_{e,g}(h) = h$ if $h \neq e, g$. Set $j = 1$.

STEP 8.1. If $t = s = 0$, set $\Psi(k) = \phi(k)$, $k = 1, \ldots, n$, and stop.

STEP 8.2. Set $y = j$. If $t = 0$, set $i = s$ and go to Step 8.5. Otherwise, set $i = t$.

STEP 8.3. Set $y = \alpha_{k_i, k_i + 1}(y)$ and $i = i - 1$.

STEP 8.4. If $i \geqslant 1$, go to Step 8.3. Otherwise, if $s = 0$, go to Step 8.6. Otherwise, set $i = s$.

STEP 8.5. Set $y = \alpha_{r_i, r_i + 1}(y)$ and $i = i - 1$.

STEP 8.6. If $i \geqslant 1$, go to Step 8.5. Otherwise, set $\Psi(j) = \phi(y)$ and $j = j + 1$.

STEP 8.7. If $j \leqslant n$, go to Step 8.2. Otherwise, stop.

Job j	1	2	3	4	5	6	7	8
$p_{1,j}$	10	12	3	5	6	11	9	4
$p_{2,j}$	7	8	2	3	9	12	13	6

Step 1 gives:

Job j	1	2	3	4	5	6	7	8
$p_{2,j}$	2	3	6	7	8	9	12	13
$p_{1,j}$	3	5	4	10	12	6	11	9

G&G przykład cd.

Steps 2 and 3 give:

Job j	$p_{2,j}$	$p_{1,\phi(j)}$	$\phi(j)$	$\max\{p_{2,j}, p_{1,\phi(j)}\}$	$\min\{p_{2,j}, p_{1,\phi(j)}\}$	$C_{j,j+1}$
1	2	3	1	3	2	0
2	3	4	3	4	3	1
3	6	5	2	6	5	0
4	7	6	6	7	6	1
5	8	9	8	9	8	0
6	9	10	4	10	9	1
7	12	11	7	12	11	0
8	13	12	5	13	12	–

At Step 4, the edges in the graph are (2,3), (4,6), and (5,8).

At Step 5, the graph has components {1}, {2,3}, {4,6}, {5,8}, and {7}.

At Step 6, edges (1,2), (3,4), (5,6), and (7,8) are added, $G_1 = \{(1,2), (5,6)\}$, $G_2 = \{(3,4), (7,8)\}$.

At Step 7, $r_1 = 5$, $r_2 = 1$, $k_1 = 3$, $k_2 = 7$.

At Step 8, for $j = 1$ we have: $y = \alpha_{7,8}(1) = 1$, $\alpha_{3,4}(1) = 1$, $\alpha_{1,2}(1) = 2$, $\alpha_{5,6}(2) = 2$, $\Psi(1) = \phi(2) = 3$. The steps are similar for $j = 2, \dots, 8$.

Thus the optimal sequence, with a cycle time of 64, is given by:

Job j	1	2	3	4	5	6	7	8
$\Psi^*(j)$	3	1	6	2	4	8	5	7

Podwójne kryteria minimalizacyjne

Twierdzenie

$P2||C_{\max}(\sum C_j)$ jest NP-trudny.

Dowód

Redukcja problemu *Even-Odd Partition*: Dany jest zbiór liczb całkowitych nieujemnych $A = \{a_1, \ldots, a_{2r}\}$, w którym $a_i \geq a_{i+1}$ $(i = 1, \ldots, 2r - 1)$. Czy istnieje $S \subset A$ taki, że $\sum_{j \in S} a_j = \sum_{j \notin S} a_j$ oraz S zawiera dokładnie jeden element spośród $\{a_i, a_{i+1}\}$ $(i = 1, \ldots, 2r - 1)$.

Znane są algorytmy przybliżone...

Twierdzenie

$P|pmtn|C_{\max}(\sum C_j)$ jest rozwiązywalny w czasie wielomianowym.

Algorytm Leung'a-Young'a

1. Mamy n zadań i m maszyn. Zakładamy, że n jest wielokrotnością m (jeśli nie, dodajemy zadania pozorne). Porządkujemy zadanie wg niemalejących czasów ich wykonywania:
$$p_1 \leq p_2 \leq \cdots \leq p_n.$$
2. Uszereguj zadania $\{Z_1, Z_2, \ldots, Z_{n-m}\}$ stosując regułę SPT (m najdłuższych zadań na razie nie szeregujemy). Otrzymujemy uszeregowanie S.
   f_i - czas zakończenia pracy maszyny M_i w uszeregowaniu S;

Algorytm Leung'a-Young'a cd.

3 Wylicz optymalną długość uszeregowania D. Dla każdego $i = 1, 2, \ldots, m$:

$$t_i = (1/i)(\sum_{j=1}^{i} f_j + \sum_{j=1}^{i} p_{n-j+1})$$

$$D = \max\{t_i\}$$

4 Uszereguj kolejno zadania $Z_{n-m+1}, \ldots, Z_n$ na m maszynach (jedno zadania do jednej maszyny). Załóżmy, że szeregujemy zadanie Z_j:

- J. $p_j \leq D - f_m$, uszereguj zadanie Z_j w całości na maszynie m. Usuń maszynę M_m i zmniejsz m o 1. Zaindeksuj maszyny ponownie tak aby były one ustawione w rosnącym porządku czasów zakończenia pracy maszyn.

Algorytm Leung'a-Young'a cd.

- J. istnieje maszyna M_i tż. $p_j = D - f_i$, uszereguj zadanie Z_j w całości na maszynie M_i. Usuń maszynę M_i i zmniejsz m o 1. Zaindeksuj maszyny ponownie tak aby były one ustawione w rosnącym porządku czasów zakończenia pracy maszyn.
- W przeciwnym przypadku, istnieje maszyna M_i tż. $p_j < D - f_i$ i $p_j > D - f_{i+1}$. Uszereguj $D - f_{i+1}$ jednostek zadania Z_j na maszynie M_{i+1} i resztę jednostek zadania przydziel do maszyny M_i. Usuń maszynę M_{i+1}, uaktualnij czas zakończenia pracy maszyny M_i, zmniejsz m o 1, przeindeksuj maszyny.

Przykład

Zasoby internetowe

- http://www2.informatik.uni-osnabrueck.de/knust/class/
- http://www-desir.lip6.fr/ durrc/query/
- bazy danych *benchmarków*
  http://web.emn.fr/x-auto/clahlou/mdl/Benchmarks.html