ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 125 Nr kol. 1391

Józef GRABOWSKI, Jarosław PEMPERA
Politechnika Wrocławska

ZAGADNIENIE SZEREGOWANIA ZADAŃ W SYSTEMACH TAŚMOWYCH ZE SPECYFICZNYMI WYMAGANIAMI

Streszczenie. W niniejszej pracy przedstawia się szereg algorytmów heurystycznych dla zagadnienia kolejnościowego taśmowego z wymaganiami typu "bez czekania" oraz z terminami dostępności zadań. Przedstawiono wyniki obliczeniowe algorytmów oraz analizę porównawczą.

SEQUENCING OF JOBS IN CONSTRAINED FLOW-SHOP PROBLEM

Summary. In the paper flow-shop scheduling problem with "no wait" requirements and with job release dates is considered. Some approximation algorithms, computation results and discussion of the performance of algorithms are presented.

1. Wstęp

W niniejszej pracy rozpatrywane jest zagadnienie szeregowania zadań w systemie taśmowym, charakteryzującym się brakiem możliwości czekania ("bez czekania") zadań w trakcie ich realizacji. Tego rodzaju specyficzne wymagania występują w procesach związanych z "obróbką" chemiczną lub zmianami stanu parametrów fizycznych wyrobów (np. produkcja wyrobów betonowych, odlewanie i obróbka wyrobów metalowych). Drugim istotnym wymaganiem (organizacyjnym) jest istnienie terminów dostępności realizowanych zadań. Termin ten może również wynikać z wykonywania zadań na maszynach o nieograniczonej przepustowości ("bez ograniczeń") podczas optymalizacji gniazd krytycznych w dyskretnych procesach produkcyjnych. Celem optymalizacji jest znalezienie takiej kolejności wykonywania zadań, aby termin zakończenia wszystkich był minimalny. W pracy przedstawiono: sformułowanie zagadnienia, krótki przegląd literatury, algorytmy heurystyczne, wyniki obliczeniowe oraz analizę porównawczą.

2. Sformułowanie problemu

Rozpatrywane w pracy zagadnienia można sformułować następująco : dany jest zbiór zadań $J=\{J_1,J_2,J_3,...,J_n\}$, które należy wykonać przy użyciu zbioru maszyn $M=\{M_1,M_2,...M_m\}$. Zadanie J_j musi być wykonane kolejno na maszynach $M_1,M_2,...M_m$, czas trwania realizacji zadania J_j na maszynie M_k wynosi p_{jk}. Każda maszyna może wykonywać w danej chwili nie więcej niż jedno zadanie, wykonywanie zadania na danej maszynie nie może być przerywane, kolejność wykonywania zadań na każdej maszynie jest taka sama. Realizacja zadania na maszynach musi następować niezwłocznie ("bez czekania"). Ponadto z każdym zadaniem związany jest jego termin dostępności r_j, tj. najwcześniejszy termin, po którym można rozpocząć realizację zadania J_j. Zagadnienie optymalizacji polega na określeniu takiej kolejności wykonywania zadań, aby minimalizować termin zakończenia realizacji wszystkich zadań $C_{max}=max_jC_j$, gdzie C_j jest terminem zakończenia realizacji zadania J_j. Oznaczmy przez S_{jk} termin rozpoczęcia wykonywania zadania J_j na maszynie M_k oraz przez $C_{jk}=S_{jk}+p_{jk}$ termin jego zakończenia na tej maszynie. Dalej oznaczmy przez $\pi=\{\pi(1),\pi(2),\pi(3),...\pi(n)\}$ permutację wykonywania zadań na maszynach. Zauważmy, że zestaw liczb S_{jk} jest rozwiązaniem dopuszczalnym dla danej permutacji π wtedy i tylko wtedy, gdy spełnione są następujące nierówności :

$$S_{\pi(j+1)k} \geq C_{\pi(j)k}=S_{\pi(j)k}+p_{\pi(j)k} \qquad k=1,2,...,m \tag{1}$$

$$S_{\pi(j)1} \geq r_{\pi(j)} \tag{2}$$

$$C_{\pi(j)k-1}=S_{\pi(j)k-1}+p_{\pi(j)k-1}=S_{\pi(j)k} \qquad k=2,3,...,m \tag{3}$$

Nierówność (1) opisuje ograniczenia wynikające z przyjętej kolejności π wykonywania zadań, nierówność (2) przedstawia ograniczenia związane z terminem dostępności, natomiast zależność (3) wynika z wymagań typu "bez czekania" (tzn. wykonywanie zadania $\pi(j)$ na maszynie k-tej musi się rozpocząć niezwłocznie po zakończeniu wykonywania tego zadania na k-1 maszynie. Ponadto, jeżeli najwcześniejszy termin zakończenia wykonywania zadania $\pi(j)$ na pierwszej maszynie $C_{\pi(j)1}=max(r_{\pi(j)}, S_{\pi(j-1)1}+p_{\pi(j-1)1})+p_{\pi(j)1}$ jest mniejszy od terminu zwolnienia drugiej maszyny przez zadanie $\pi(j-1)$, wówczas należy przesunąć termin rozpoczęcia wykonywania zadania $\pi(j)$ na maszynie pierwszej, czyli $S_{\pi(j)1}=C_{\pi(j-1)2}-p_{\pi(j)1}$).

Z zależności (1)-(3) wynika, że dla danej permutacji π rozwiązanie dopuszczalne może być wyznaczone za pomocą wzorów rekurencyjnych (dla $j=1,2,3,...,n$):

$$S_{\pi(j)1}=max(S_{\pi(j-1)1}+p_{\pi(j-1)1}, r_{\pi(j)}), \tag{4}$$

$$S_{\pi(j)k}=max(S_{\pi(j-1)k}+p_{\pi(j-1)k}, S_{\pi(j)k-1}+p_{\pi(j)k-1}), \qquad k=1,2,...,m \tag{5}$$

$$S_{\pi(j)k-1}=S_{\pi(j)k}-p_{\pi(j)k-1} \qquad k=1,2,...,m \tag{6}$$

$$S_{\pi(0)k}=0, p_{\pi(0)k}=0, \qquad k=1,2,...,m$$

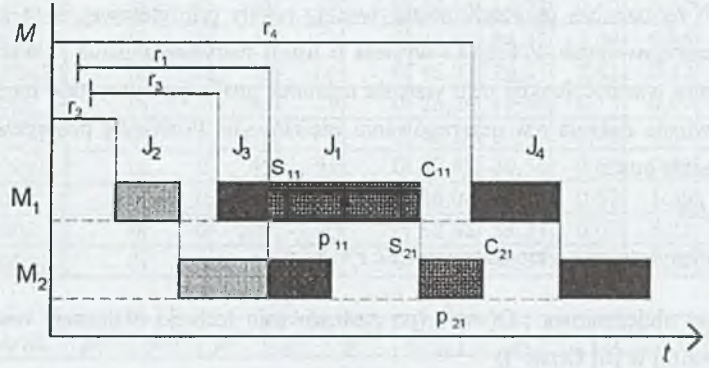
Równanie (4) określa najwcześniejszy termin rozpoczęcia wykonywania zadania $\pi(j)$ na pierwszej maszynie, równanie (5) przedstawia najwcześniejszy termin rozpoczęcia wykonywania zadania $\pi(j)$ na k-tej maszynie, równanie (6) określa przesunięcie terminów

rozpoczęcia wykonywania zadania $\pi(j)$ na k-1 oraz k-tej maszynie, co wynika z wymagań "bez czekania".

Teraz problem nasz możemy sformułować następująco: znaleźć kolejność wykonywania zadań π^* taką, że $C_{max}(\pi^*)=min_{\pi}(C_{\pi(n)m})=min_{\pi}(S_{\pi(n)m}+p_{\pi(n)m})$, gdzie $S_{\pi(n)m}$ może być wyznaczony przy użyciu zależności (4) - (6).

Na rys.1 przedstawiono przykład dopuszczalnego uszeregowania dla n=4, m=2, π=(2, 3, 1, 4). Rozpatrywane zagadnienie, już dla $m\geq 2$ jest NP-zupełne. Stąd też ewentualne zastosowanie metod dokładnych staje się niemożliwe dla zagadnień o rozmiarach występujących w praktycznych zastosowaniach. Dlatego też wydaje się celowe poszukanie szybkich algorytmów heurystycznych. Właśnie w niniejszej pracy przedstawia się szereg tego typu algorytmów opartych na statycznych i dynamicznych regułach priorytetowych oraz na technice "wstawiania" [4][5]. Wszystkie te algorytmy posiadają wielomianową złożoność obliczeniową.

Pewne szczególne przypadki naszego zagadnienia były rozpatrywane w literaturze. I tak, dla r_j=0, j=1,2,...,n przegląd algorytmów został przedstawiony w pracy [2]. Natomiast, jeżeli dodatkowo założymy, że m=2, to w pracy [1] zaprezentowano algorytm wielomianowy o złożoności O($n$ln$n$), znajdujący rozwiązanie optymalne.



Rys. 1. Przykład uszeregowania dopuszczalnego

Fig. 1. An example of feasible solution

3. Opis algorytmów

3. 1. Algorytmy oparte na statycznej regule priorytetowej

Opis : Uszereguj zadania według statycznej reguły priorytetowej.

a) algorytm *H1*

reguła priorytetowa: niemalejące r_j

złożoność obliczeniowa : O($n$ln$n$)

b) algorytm *H2*

reguła priorytetowa: niemalejąca wartość $r_j + \sum_{k=1}^{n} p_{jk}$

złożoność obliczeniowa : $O(mn+n\ln n)$

3. 2. Algorytmy oparte na dynamicznej regule priorytetowej

Opis: Niech K oznacza zbiór zadań dotychczas jeszcze nie uszeregowanych. Ze zbioru K wybierz zadanie j według dynamicznej reguły priorytetowej i wstaw je jako ostatnie w uszeregowaniu częściowym. Usuń zadanie j ze zbioru K. Powtarzaj postępowanie aż zbiór K będzie zbiorem pustym.

algorytm *H3*

dynamiczna reguła priorytetowa : najmniejsza wartość $st_j = \sum_{k=1}^{n} (S_{jk} - C_k)$, gdzie C_k jest terminem zakończenia wykonywania zadań w uszeregowaniu częściowym na k-tej maszynie;

złożoność obliczeniowa : $O(mn^2)$

3. 3. Algorytmy oparte na technice wstawiania

Opis: Niech Ls oznacza uporządkowaną, według reguły priorytetowej, listę zadań jeszcze nieuszeregowanych. Z listy Ls wybierz (i usuń) pierwsze zadanie j. Wstaw zadanie j tak, aby wartość funkcji celu wzrosła najmniej, analizując wszystkie możliwe miejsca wstawienia zadania j w uszeregowaniu częściowym. Powtarzaj postępowanie aż lista Ls będzie pusta.

algorytm *H4*

reguła priorytetowa : nierosnąca wartość $r_j + \sum_{k=1}^{n} p_{jk}$

złożoność obliczeniowa : $O(mn^3)$ (po zastosowaniu techniki obliczania wartości funkcji celu zawartej w [6] $O(mn^2)$)

3. 4. Algorytm oparty na technice losowania kolejności wykonywania zadań

algorytm *H5*. Wybierz losowo kolejność wykonywania zadań π

złożoność obliczeniowa : $O(mn)$

4. Wyniki badań testujących

Algorytmy *H1, H2, H3, H4* i *H5* zostały zakodowane w języku C++ i były testowane na komputerze IBM RISC System/6000 200 MHz. Badania testowe przeprowadzono na przykładach, dla których p_{jk} zostały wygenerowane losowo wg rozkładu równomiernego ze zbioru {1, 2, ..., 100}, r_j wg rozkładu równomiernego ze zbioru $\{0,1,...,\alpha*R*n\}$, gdzie α jest

pewnym parametrem, natomiast $R*n$ jest wartością oczekiwaną C_{max} dla naszego problemu (r_j=0). Dla każdego z n=10, 50, 100, 200, m=2, 5, 10 oraz α=0.0, 0.5, 1.0, 2.0, 5.0 wygenerowano 100 przykładów testujących. Dla każdego przykładu wyliczono następujące wielkości :

C^H - wartość funkcji celu otrzymana algorytmem H dla $H \in \{H1, H2, H3, H4, H5\}$,

$C^m = min_H(C^H)$ - najmniejsza wartość funkcji celu otrzymana jednym z algorytmów $H1, H2, H3, H4, H5$.

Na podstawie tych wielkości obliczono :

$r^H = 100\%(C^H - C^m)/C^m$ - względny błąd rozwiązania otrzymanego algorytmem H względem najlepszego rozwiązania otrzymanego jednym z algorytmów $H1, H2, H3, H4$ i $H5$,

$r^H_{śr}$ - średni względny błąd dla algorytmu H,

p^H - procent przykładów dla których $C^H = C^m$.

Tablica 1

Wyniki badań

nxm	p^{H1}	p^{H2}	p^{H3}	p^{H4}	p^{H5}	$r^{H1}_{śr}$	$r^{H2}_{śr}$	$r^{H3}_{śr}$	$r^{H4}_{śr}$	$r^{H5}_{śr}$
α=0,0										
10x 2	0	1	17	85	0	19,81	12,62	4,31	0,23	20,09
10x 5	0	0	10	90	0	29,01	18,79	7,29	0,19	31,03
10x10	0	0	10	92	0	29,05	17,98	6,76	0,07	29,98
50x 2	0	0	53	49	0	24,06	14,73	0,62	0,62	24,33
50x 5	0	0	46	54	0	41,69	24,32	1,04	0,85	41,41
50x10	0	0	15	86	0	46,73	26,27	2,42	0,13	47,34
100x 2	0	0	81	19	0	26,24	16,64	0,13	0,97	26,26
100x 5	0	0	92	8	0	46,20	27,49	0,08	2,36	46,85
100x10	0	0	45	55	0	52,88	30,56	0,60	0,45	53,43
200x 2	0	0	96	4	0	28,01	17,71	0,02	1,29	27,95
200x 5	0	0	100	0	0	52,82	33,21	0,00	5,21	53,20
200x10	0	0	99	1	0	60,70	36,31	0,00	2,29	60,80

nxm	p^{H1}	p^{H2}	p^{H3}	p^{H4}	p^{H5}	$r^{H1}_{śr}$	$r^{H2}_{śr}$	$r^{H3}_{śr}$	$r^{H4}_{śr}$	$r^{H5}_{śr}$
α=0,5										
10x 2	0	1	31	74	0	11,75	13,81	3,24	0,57	37,72
10x 5	0	1	34	66	0	15,84	15,77	3,52	0,99	36,66
10x10	0	0	28	72	0	15,60	17,02	4,52	0,72	37,36
50x 2	0	0	92	9	0	19,76	20,40	0,06	2,54	65,74
50x 5	0	0	97	3	0	27,67	28,43	0,06	3,72	73,63
50x10	0	0	91	9	0	29,88	30,44	0,09	4,41	76,71
100x 2	0	0	98	2	0	23,15	23,34	0,00	3,65	73,09
100x 5	0	0	100	0	0	35,03	35,16	0,00	7,06	90,58
100x10	0	0	100	0	0	38,81	38,82	0,00	8,04	94,73
200x 2	0	0	100	0	0	25,86	26,06	0,00	4,53	80,12
200x 5	0	0	100	0	0	42,20	41,90	0,00	10,32	103,89
200x10	0	0	100	0	0	45,79	45,83	0,00	11,67	109,69

α=1,0

10x 2	24	14	46	74	0	4,68	6,30	1,57	0,49	43,80
10x 5	14	15	41	77	0	5,25	6,48	1,95	0,46	40,55
10x10	23	18	52	78	0	4,95	6,32	1,90	0,51	37,57
50x 2	6	6	67	51	0	4,41	4,96	0,18	0,61	75,73
50x 5	7	7	63	51	0	4,46	4,63	0,36	0,58	73,91
50x10	7	3	64	52	0	5,09	4,95	0,36	0,57	73,17
100x 2	3	1	68	42	0	4,37	4,50	0,11	0,75	83,89
100x 5	7	2	66	57	0	3,61	3,91	0,15	0,46	82,18
100x10	9	3	65	53	0	3,27	3,70	0,16	0,40	80,66
200x 2	2	3	81	34	0	3,71	3,81	0,03	0,55	89,51
200x 5	4	4	73	42	0	3,28	3,39	0,06	0,40	87,67
200x10	4	2	67	48	0	2,91	3,05	0,07	0,38	86,52

α=2,0

10x 2	71	63	78	96	0	0,85	1,14	0,29	0,04	27,52
10x 5	72	68	79	98	2	0,72	1,03	0,47	0,03	24,99
10x10	72	67	81	98	1	0,88	1,32	0,28	0,02	24,00
50x 2	74	76	89	94	0	0,14	0,16	0,04	0,01	38,49
50x 5	69	62	77	97	0	0,19	0,29	0,09	0,00	38,24
50x10	72	63	80	94	0	0,19	0,24	0,09	0,02	36,78
100x 2	74	60	83	91	0	0,06	0,16	0,02	0,01	41,58
100x 5	78	72	84	94	0	0,06	0,12	0,03	0,01	40,77
100x10	66	67	76	94	0	0,10	0,11	0,04	0,01	40,78
200x 2	74	69	82	92	0	0,03	0,05	0,01	0,01	44,22
200x 5	70	61	71	95	0	0,04	0,06	0,03	0,00	42,90
200x10	70	61	85	97	0	0,05	0,09	0,02	0,00	43,26

α=5,0

10x 2	97	98	97	99	3	0,03	0,02	0,02	0,00	11,30
10x 5	98	95	98	100	4	0,00	0,07	0,02	0,00	11,22
10x10	98	91	95	99	5	0,02	0,15	0,06	0,00	10,59

nxm	p^{H1}	p^{H2}	p^{H3}	p^{H4}	p^{H5}	r^{H1}_{ir}	r^{H2}_{ir}	r^{H3}_{ir}	r^{H4}_{ir}	r^{H5}_{ir}

α=5,0

50x 2	93	91	95	100	0	0,01	0,02	0,00	0,00	14,19
50x 5	91	92	96	98	0	0,01	0,02	0,01	0,00	14,01
50x10	92	90	94	100	0	0,01	0,02	0,01	0,00	13,69
100x 2	96	90	96	100	0	0,00	0,01	0,00	0,00	15,43
100x 5	94	89	95	100	0	0,00	0,01	0,00	0,00	15,28
100x10	90	88	93	100	0	0,01	0,01	0,00	0,00	15,03
200x 2	96	91	95	99	0	0,00	0,00	0,00	0,00	16,96
200x 5	91	87	97	98	0	0,00	0,01	0,00	0,00	24,84
200x10	89	84	88	99	0	0,01	0,01	0,01	0,00	32,52

Z tablicy 1 wynika, że najlepszymi algorytmami są *H3* i *H4*. Rozwiązania otrzymane jednym z nich, poza nielicznymi przypadkami, są najlepsze spośród rozwiązań otrzymanych badanymi algorytmami. Algorytm *H3* jest zdecydowanie najlepszym algorytmem dla dużej liczby zadań ($n \geq 50$). Średni błąd względny tego algorytmu zmniejsza się wraz ze wzrostem liczby zadań n i nieznacznie zwiększa się wraz ze wzrostem liczby maszyn m, osiągając średnią wartość mniejszą niż 0.1% dla $n=200$. Algorytm *H4* daje najlepsze wyniki dla małych wartości n (dla $n=10$ średni błąd względny nie przekracza 1%). Wraz ze wzrostem liczby zadań błąd ten zwiększa się osiągając wartość większą od 10% dla $\alpha=0.5$, $n=200$. Warto zauważyć, że dla klasycznego problemu taśmowego średni błąd względny tego typu algorytmu maleje wraz ze wzrostem n. Potwierdzają to badania zawarte w [3]. Średnie błędy względne algorytmów *H1* i *H2* dla małych wartości α są porównywalne z błędami algorytmu *H5*. Z tego powodu są one praktycznie bezużyteczne dla tego rodzaju danych, natomiast dla dużych wartości $\alpha=5.0$ (podobnie jak algorytmy *H3* i *H4*) znajdują one przeszło 80% najlepszych rozwiązań. Dla małych wartości n zdecydowanie najlepszym algorytmem jest algorytm *H4*, natomiast dla dużych wartości n, niezależnie od α, algorytm *H3*.

LITERATURA

1. Gilmore P. C., Gomory R. E. : Sequencing a One State Variable Machine: A Solvable Case of the Travelling Salesman Problem. *Operations Research 12, 1964, pp. 655-679.*
2. Hall N. G., Sriskandarajanh C. : A survey of machine scheduling problems with blocking and no-wait in process, *Operations Research 44, 1996, pp. 510-525.*
3. Nowicki E., Smutnicki C. : A fast tabu search algorithm for the permutation flow shop problem. *European Journal of Operational Research 91, 1966, pp. 160-175.*
4. Nawaz M., Enscore E. E., Ham Jr. I. : A heuristic algoritm for the m-machine n-job flow-shop sequencing problem. *Omega International Journal of Management Science 11(7) 1983, pp. 91-95.*
5. Rajedran C. : A no wait flow shop scheduling heuristic to minimize makespan. *Journal of Operational Research Society 45, 1994, pp. 472-478.*
6. Taillard E. : Some efficient heuristic methods for flow shop sequencing. *European Journal of Operational Research 47, 1990, pp. 65-74.*

Recenzent: Prof.dr hab.inż. Tadeusz Sawik

Abstract

This paper deals with sequencing of jobs in the flow-shop problem with release date and no-wait requirements. This problem can formulated as follows. There is the set of jobs $J_1,J_2,...,J_n$, each of n jobs has to be processed on machines $M_1,M_2,...,M_m$ in that order.

A machine can process only one job at a time and preemption of a job is not permitted. Furthermore, between the pairs of machines (M_k, M_{k+1}), $k=1,2,...,m-1$, there are "no wait" requirements. The purpose of the optimization is to find such a schedule of jobs on machines that maximum completion time of jobs is minimized. Our problem belongs to the class of NP-hard problems what justifies searching for heuristic algorithms. In the paper we propose several approximation algorithms based on statistical and dynamical priority rules. Finally, the computational results and discussion of the performance of algorithms are presented.