ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 124 Nr kol. 1390

Józef GRABOWSKI, Aneta MARCHEWKA
Politechnika Wrocławska

ZAGADNIENIA SZEREGOWANIA WYROBÓW W PRZEPŁYWOWYCH PROCESACH MONTAŻOWYCH

Streszczenie. W niniejszej pracy przedstawia się zagadnienie oraz algorytm harmonogramowania wyrobów w przepływowym procesie produkcyjnym, w którym proces przetwarzania zawiera operacje obróbcze oraz operacje montażowe, a ponadto wyroby są przedstawiane w postaci grafu drzewa.

SCHEDULING OF JOBS IN ASSEMBLY FLOW-SHOP PROBLEM

Summary. This paper deals with a algorithm for the flow-shop scheduling problem where the jobs are presented by a graph-tree. Such problem appears in a production, when several elements are assembled into one job.

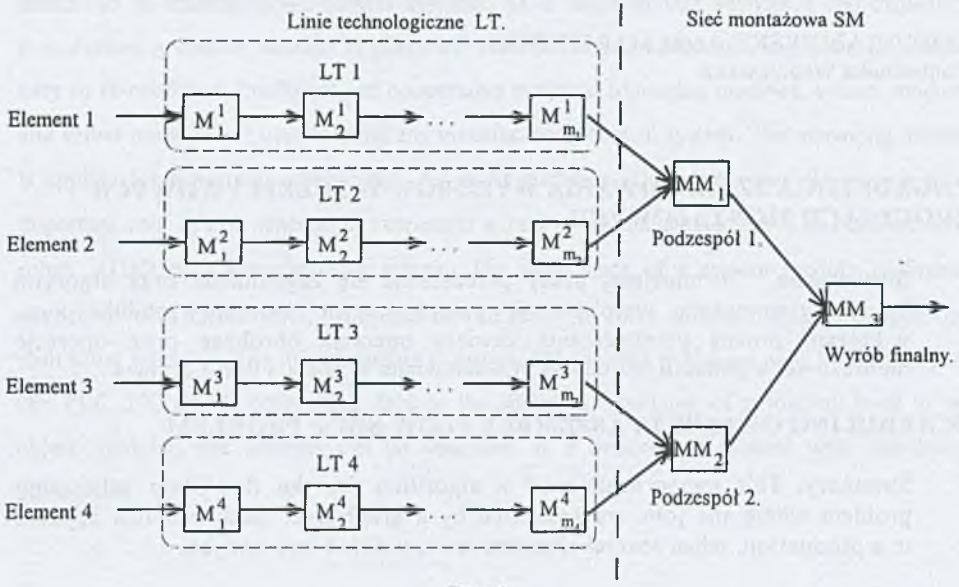
1. Wstęp

W wielu procesach, w dyskretnych systemach produkcyjnych, przy wyznaczaniu harmonogramu produkcji, oddzielnie rozpatruje się proces wytwarzania elementów oraz oddzielnie proces montażu tych elementów. Dla wyrobów o mało złożonej budowie montaż odbywa się na jednym stanowisku montażowym, który jest ostatnią operacją w danym cyklu produkcyjnym. Dla bardziej złożonych wyrobów montaż odbywa się na wielu stanowiskach, które stanowią linię montażową lub sieć montażową [7]. W niniejszej pracy przedstawia się zagadnienia harmonogramowania wyrobów, w któryych oprócz produkcji elementów, występuje sieć montażowa. Celem optymalizacji jest minimalizacja terminu zakończenia wszystkich wyrobów. Praca zawiera opis technologiczny procesu, model matematyczny zagadnienia, algorytm oparty na technice Tabu Search (TS) oraz wyniki obliczeniowe.

2. Opisowe przedstawienie zagadnienia

W rozważanym dyskretnym procesie przepływowym wyrób finalny jest uzyskiwany poprzez montaż podzespołów na ostatnim (finalnym) stanowisku

montażowym. Z kolei podzespoły te są przedmiotem produkcji, w tym również montażu, na wcześniejszych stanowiskach produkcyjnych.



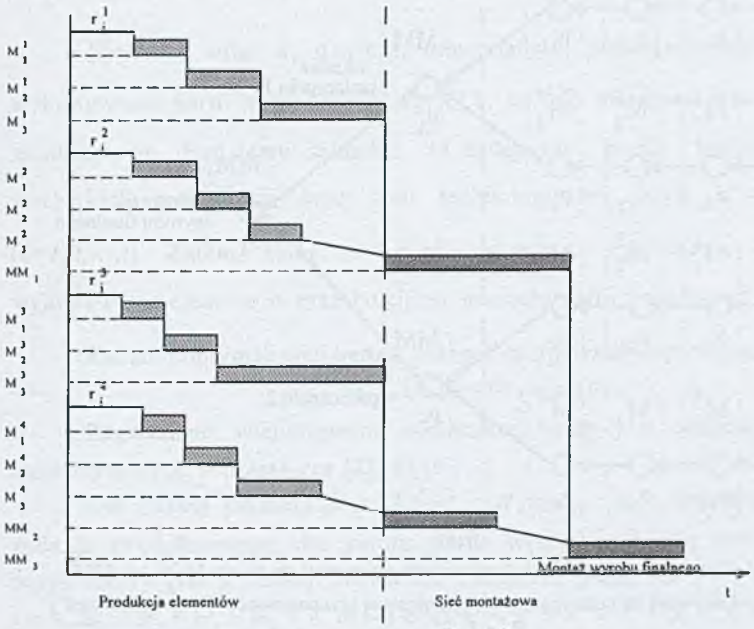
Rys.1.
Fig.1.

Ogólnie, proces produkcyjny składa się z dwóch części: produkcyjnej oraz sieci montażowej. W części produkcyjnej są realizowane czynności obróbcze wytwarzające poszczególne elementy składowe, które dalej są "składane" w sieci montażowej w podzespoły, zespoły itd. aż do zmontowania wyrobu gotowego (rys.1). Do czynności obróbczych zaliczane są również czynności przygotowawcze, pomiary, kontrola, prace wykończeniowe itp. Część produkcyjna składa się z linii technologicznych (LT) wyposażonych w maszyny służące do wytwarzania elementów. Maszyny te będziemy oznaczać symbolem M_i^k. Z kolei, sieć montażowa (SM) zawiera stanowiska montażowe, które będziemy również (dla wygody dalszych rozważań) nazywać maszynami i oznaczać symbolem MM_j.

W procesie produkcyjnym mogą być realizowane różne produkty finalne w postaci pojedynczych wyrobów lub ich partii. Czasy trwania operacji obróbczych są znane i różne dla różnych wyrobów. Zagadnienie optymalizacji procesu sprowadza się do wyznaczenia takiego harmonogramu (kolejności) realizacji wyrobów finalnych (oraz kolejności wykonywania ich elementów składowych i podzespołów), aby minimalizować określone kryterium optymalizacji. Zwykle, w tego rodzaju procesach, w początkowym etapie produkcyjnym realizowane są operacje przygotowawcze, takie

jak transport elementów surowych, czyszczenie, kontrola itp. Operacje te są wykonywane przy użyciu środków transportowych, grup pracowników itp. Łączna moc produkcyjna tych środków jest dostatecznie duża, tak że nie stanowią one wąskiego gardła (gniazda krytycznego), zatem nie musimy tutaj zakładać konieczności szeregowania wyrobów. Ten zbiór środków możemy traktować jako jedną maszynę o nieograniczonej przepustowości ("bez ograniczeń") realizującą pojedynczą operację o czasie trwania równym sumie czasów kolejnych operacji. Czas trwania tej operacji może być traktowany jako najwcześniejszy możliwy termin rozpoczęcia r_i^k realizacji elementu k dla wyrobu W_i.

Na wykresie Gantta przedstawiono proces produkcji wyrobu W_i.



Rys. 2.
Fig.2.

3. Model matematyczny zagadnienia

Załóżmy, że zadanie produkcyjne jest zadane w postaci zbioru partii różnych wyrobów:

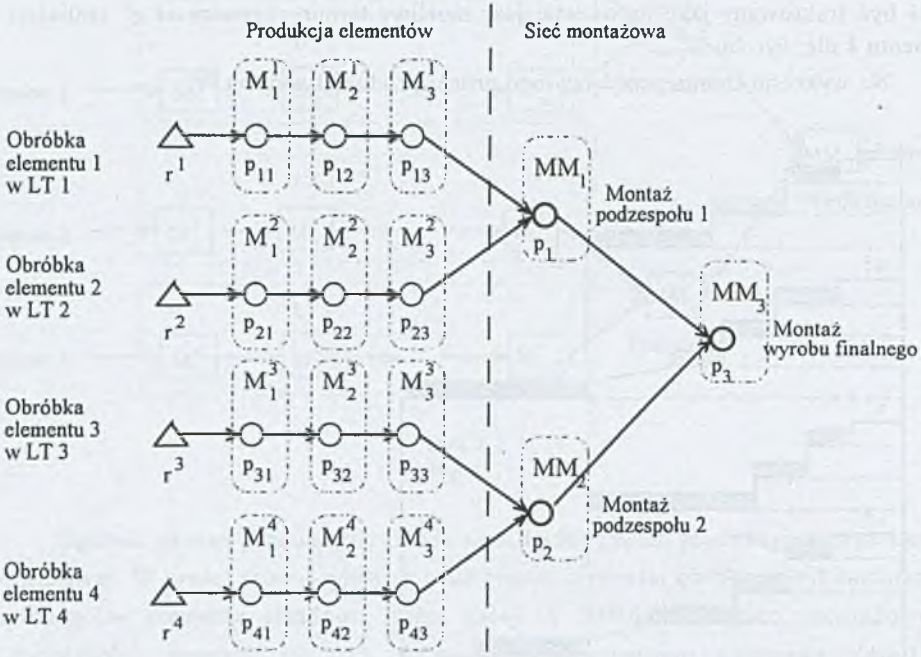
$$P = \{P_1, P_2, \ldots P_n\}.$$

Każda partia składa się ze zbioru jednakowych wyrobów W_i. Liczba wyrobów w partii P_i wynosi n_i, czyli $|P_i| = n_i$.

Każdy wyrób W_i może być przedstawiony w postaci grafu-drzewa (rys.3):

$$W_i = \langle N_i, PT_i \rangle,$$

gdzie : $N_i = \{O_1^i, O_2^i, ..., O_{w_i}^i\}$ jest zbiorem wszystkich operacji (obróbczych i montażowych), które należy zrealizować w trakcie produkcji wyrobu W_i, natomiast PT_i jest relacją (zbiorem par operacji) reprezentujących porządek technologiczny wykonywania operacji. Jeżeli $\langle O_k^i, O_l^i \rangle \in PT_i$, to oznacza, że operacja O_k^i jest wykonywana przed operacją O_l^i. Operacja O_k^i jest wykonywana w czasie p_{ik}.



○ - oznacza operacje obróbcze lub montażowe wykonywane na maszynach ze zbioru MOB lub MMT.

△ - oznacza operacje wykonywane na maszynach o nieograniczonej przepustowości ("bez ograniczeń").

Rys.3.
Fig.3.

Zbiór operacji N_i możemy przedstawić w postaci:

$$N_i = N_i^1 \cup N_i^2,$$

gdzie :N_i^1- jest zbiorem operacji obróbczych związanych z produkcją elementów dla wyrobu W_i. N_i^2- jest zbiorem wszystkich operacji montażowych występujących w trakcie produkcji wyrobu W_i.

Operacje zbioru N_i^1 są wykonywane przy użyciu zbioru maszyn:

$$MOB = \{M_1^1, M_2^1, ..., M_{m_1}^1, M_1^2, M_2^2, ..., M_{m_2}^2, ..., M_1^t, M_2^t, ..., M_{m_t}^t\},$$

gdzie : $\{M_1^k, M_2^k, ..., M_{m_k}^k\}$ jest podzbiorem maszyn w k-tej linii technologicznej,

m_k - liczba maszyn w tej linii, t - liczba linii technologicznych.

Operacje ze zbioru N_i^2 są realizowane na odpowiednich stanowiskach montażowych, których zbiór oznaczamy przez:

$$MMT = \{MM_1, MM_2, ..., MM_m\},$$

gdzie : m jest liczbą stanowisk montażowych.

Zadanie optymalizacji całego procesu polega na znalezieniu takiego uszeregowania (harmonogramu) partii wyrobów finalnych, ich podzespołów i elementów składowych, aby minimalizować termin zakończenia wszystkich wyrobów $C_{max} = \max_i C_i$, gdzie C_i jest terminem zakończenia realizacji wyrobów z partii P_i.

Niech π_l^k oraz π_j oznacza, odpowiednio, permutację określającą kolejność wykonywania partii pomp $P = \{P_1, P_2, ..P_n\}$, na M_l^k maszynie oraz MM_j stanowisku montażowym. Będziemy zakładać, że kolejność partii będzie jednakowa na wszystkich maszynach danej linii technologicznej, czyli $\pi_1^k = \pi_2^k = ... = \pi_{m_k}^k = \pi^k$, $(k=1,2,...,t)$. Zatem, zbiór $\pi = \{\pi^1, \pi^2, ..., \pi^t, \pi_1, \pi_2, ..., \pi_m\}$ będzie określał kolejność wykonywania elementów oraz kolejność montażu partii wyrobów P.

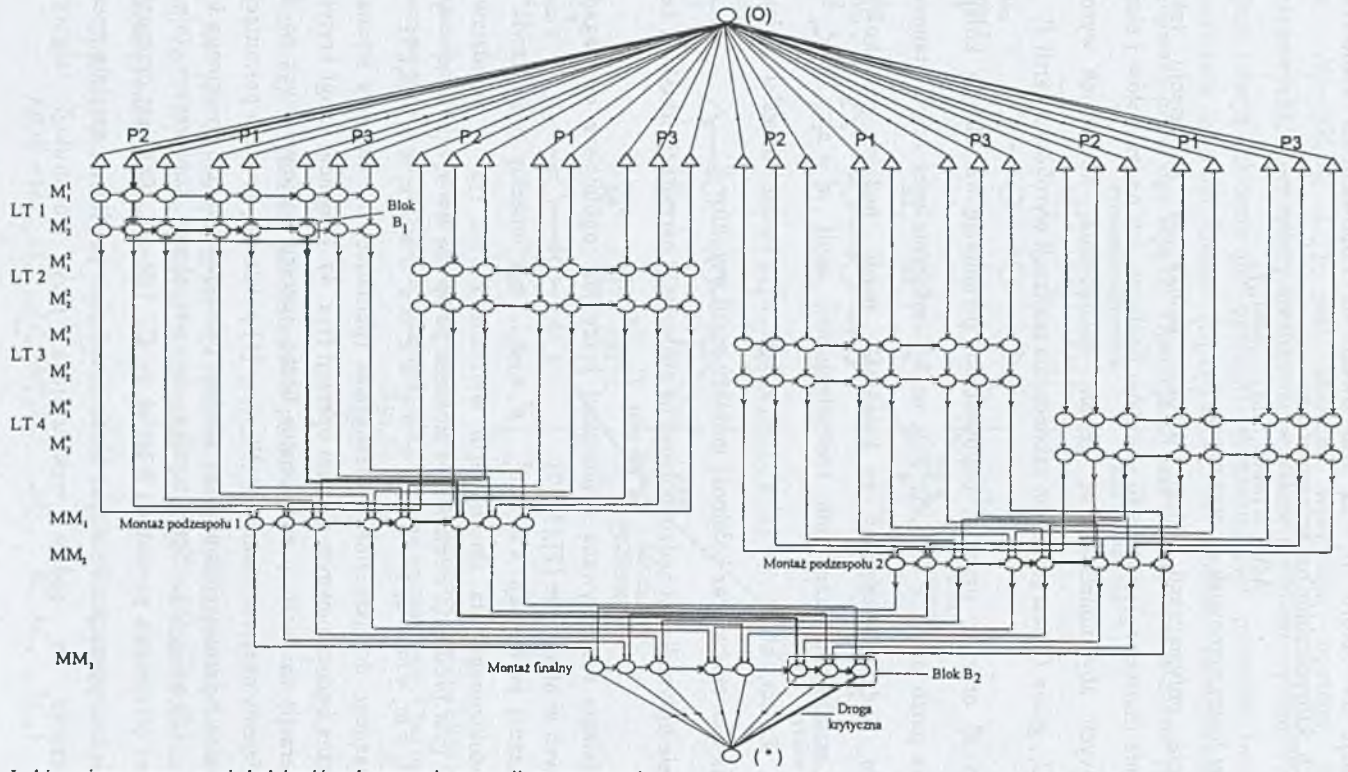
Zadanie optymalizacji będzie polegać na znalezieniu permutacji π^* takiej, że

$$C_{max}(\pi^*) = \min_{\pi} C_{max}(\pi).$$

Zagadnienie rozpatrywane w niniejszej pracy jest ogólniejsze od zagadnień rozpatrywanych w literaturze [2],[3],[9].

Dla każdej permutacji $\pi = \{\pi^1, \pi^2, ..., \pi^t, \pi_1, \pi_2, ..., \pi_m\}$ możemy przedstawić graf zadania produkcyjnego dla zbioru partii wyrobów P. Na rys. 4 przedstawiono przykładowy graf produkcji elementów i montażu pomp dla $n = 3$, $n_1 = 2$, $n_2 = n_3 = 3$, $t = 4$, $m_1 = m_2 = m_3 = m_4 = 2$, $m = 3$ oraz $\pi^1 = \pi^2 = \pi^3 = \pi^4 = \pi_1 = \pi_2 = \pi_3 = (2,1,3)$.

Dla każdego dopuszczalnego rozwiązania (permutacji π) możemy wyznaczyć drogę krytyczną będącą pewnym ciągiem operacji (rys. 4). Fragment drogi krytycznej (podciąg operacji) zawierający maksymalną liczbę operacji wykonywanych na danej maszynie będziemy nazywać blokiem B. W pracy [1] pokazano, że jeżeli permutacja β jest otrzymana z permutacji π poprzez zamianę operacji wewnątrz dowolnego bloku w π, to $C_{max}(\beta) \geq C_{max}(\pi)$. Stąd wynika, że warunkiem koniecznym (ale nie dostatecznym) otrzymania permutacji β takiej, że $C_{max}(\beta) < C_{max}(\pi)$, jest przesunięcie co najmniej jednej operacji z wewnątrz bloku przed pierwszą lub za ostatnią operację bloku. Własność ta będzie wykorzystana przy konstrukcji algorytmu.



Łuki poziome reprezentują kolejność wykonywania operacji na maszynach.
Pozostałe łuki reprezentują porządek technologiczny

Rys. 4.
Fig.4.

4. Algorytm przybliżony

Rozpatrywane zagadnienie jest NP-zupełne, bowiem problemy będące jego szczególnymi przypadkami, są NP-zupełne [4],[6]. Stąd też możliwości skonstruowania algorytmu dokładnego, który rozwiązywałby ten problem w realnym czasie, są raczej niewielkie. Zatem, algorytmy przybliżone mogą stanowić jedyne praktyczne podejście.

Dla rozwiązania naszego problemu proponujemy algorytm oparty na technice Tabu Search (TS). Technika TS jest aktualnie jednym z najlepszych podejść [5] dla rozwiązania tego rodzaju zagadnień szeregowania. Algorytm TS startuje z pewnej permutacji początkowej π^0, przeszukuje wszystkich jej "sąsiadów" (poprzez przesunięcie zadań w π^0) w celu znalezienia lepszego (najlepszego) rozwiązania (tzn. permutacji β) o najmniejszej wartości funkcji celu. Wybrane w ten sposób rozwiązanie staje się rozwiązaniem początkowym w następnym kroku. W celu zapobieżenia powstawania cyklu oraz zapewnienia możliwości wyjścia z minimum lokalnego tworzy się cykliczną listę przesunięć (ruchów) zabronionych (lista tabu), będącą swego rodzaju "pamięcią historii" przeszukiwań. W algorytmach TS stosuje się szereg warunków zakończenia działania, np. zadana liczba iteracji, czas obliczeń, wykonanie maksymalnej liczby iteracji bez zmniejszania wartości funkcji celu itp.
W naszym algorytmie wykorzystano własności eliminacyjne bloków, co pozwoliło na ograniczenie sąsiedztwa przeszukiwań. Ze względu na ograniczoną objętość artykułu, niestety nie możemy przedstawić wszystkich szczegółów algorytmu.

Podczas wyznaczania początkowej permutacji π^0 założyliśmy, że dla linii technologicznych oraz stanowisk montażowych uszeregowanie wyrobów jest takie samo, czyli $\pi^0 = \pi^1 = \pi^2 = \ldots = \pi^j = \pi_1 = \ldots = \pi_m$. W celu znalezienia π^0 wykorzystaliśmy odpowiednio zmodyfikowany algorytm NEH [8]. Podstawowym elementem tego algorytmu jest lista priorytetów dla wyrobów. Na potrzeby naszego zagadnienia, lista ta zawierała wyroby W_i uporządkowane zgodnie z nierosnącymi wartościami ich najdłuższych dróg grafu-drzewa, w którym węzły są obciążone czasami trwania operacji.

W pierwszym głównym kroku algorytmu wybiera się pierwsze zadanie z listy priorytetów. Tworzy ono jednoelementowe uporządkowanie W(1). W kolejnym s+1 głównym kroku do częściowo uporządkowanych s-wyrobów [W(1),W(2),...W(s)] dodajemy następny wyrób W(s+1) z naszej listy priorytetów oraz wykonujemy s+1 lokalnych kroków, polegających na wstawianiu naszego wyrobu pomiędzy uszeregowane już wyroby. Otrzymujemy s+1 uszeregowań: [W(s+1),W(1),W(2),...,W(s)], [W(1),W(s+1),W(2),...W(s)],...,[W(1),W(2),..W(s),W(s+1)], z których wybieramy uszeregowanie, dające najmniejszą wartość terminu zakończenia wykonywania

powyższych wyrobów. Po wykonaniu n głównych kroków otrzymujemy poszukiwaną permutację początkową π^0.

5. Wyniki obliczeniowe

Algorytm Tabu Search został zaprogramowany w języku Pascal i uruchomiony na komputerze IBM RISC System/6000,200 MHz oraz został przetestowany dla danych wygenerowanych losowo. Wartości czasów realizacji poszczególnych operacji oraz wartości r_i dla n partii zostały wygenerowane z przedziału [1,100] o jednostajnym rozkładzie. W badaniach eksperymentalnych przyjęliśmy, że wyroby W_i posiadają jednakową strukturę w postaci grafu-drzewa binarnego, wtedy mamy $m=t-1$. Przyjęliśmy również, że $m_k=3$ dla $k=1,2,...,t$. Zatem liczba wszystkich maszyn (obróbczych i montażowych) wynosi $lm=3 \cdot t+t-1=4 \cdot t-1$ Dla eksperymentu obliczeniowego przyjęliśmy $t=2,4,8$, czyli $lm=7,15,31$. Dalej, przyjęliśmy, że $n=10,20,40,80$ oraz że liczby wyrobów w partiach są jednakowe i wynoszą $n_i=10$ $(i=1,2,...,n)$. Dla każdego przykładu o rozmiarach $lm \times n$: 7×10, 7×20, 7×40,7×80, 15 ×10,...,15×80,31×10,...,31×80 wygenerowano 10 przykładów testujących.

Dla każdego przykładu wyznaczono następujące wielkości:

C_O - wartość funkcji celu dla permutacji początkowej,

C_A - najlepsza wartość funkcji celu otrzymana w ciągu pierwszych 1000 iteracji,

C_B - najlepsza wartość funkcji celu otrzymana w ciągu pierwszych 10000 iteracji,

C_{lok} - wartość funkcji celu otrzymana przez algorytm dla minimum lokalnego,

TIME - czas obliczeń CPU.

W oparciu o te wartości wyznaczyliśmy:

$PR(lok) = 100\%(C_O - C_{lok})/C_O$ - średnia (procentowa) wartość poprawy funkcji celu dla minimum lokalnego względem wartości początkowej C_O.

$PR(x) = 100\%(C_O - C_x)/C_O$ - średnia (procentowa) wartość poprawy funkcji celu względem wartości początkowej C_O, $x \in \{A,B\}$.

time/iter - średni czas jednej iteracji CPU (w milisekundach).

W tablicy 1 przedstawiono wyniki obliczeniowe.

Zaproponowany algorytm poprawiał rozwiązanie początkowe w granicach 6-20% dla pierwszego 1000 iteracji oraz 6-21% dla pierwszych 10000 iteracji. Największa poprawa (ponad 90%) została uzyskana w ciągu pierwszych 1000 iteracji. Poprawa funkcji celu zwiększa się wraz ze wzrostem liczby maszyn *lm*, jak i liczby partii *n* i osiąga poziom stabilny ok. 21%.

Tablica 1

Wyniki obliczeniowe

lm × n	*PR(lok)*	*PR(A)*	*PR(B)*	*time/iter*
7 ×10	2.63	6.29	6.29	0.3
7×20	3.98	7.41	8.35	0.9
7×40	4.96	10.44	10.61	2.4
7×80	5.77	11.92	11.92	9.0
15×10	7.05	12.40	12.42	0.6
15×20	8.43	14.00	14.90	1.6
15×40	9.10	14.15	14.19	5.3
15×80	9.72	14.99	15.08	10.0
31×10	9.98	17.21	17.21	0.8
31×20	11.58	17.70	17.77	2.2
31×40	12.34	20.23	20.23	6.1
31×80	15.76	20.93	20.94	12.8

Analiza wyników pokazuje, że przy poszukiwaniu minimum lokalnego wielkość poprawy również się zwiększa wraz ze wzrostem *n* i *lm*. W oparciu o uzyskane dane można stwierdzić, że ewentualne zawężenie działania algorytmu do znalezienia minimum lokalnego byłoby niekorzystne, bowiem jest on bardziej skuteczny przy pełnym przebiegu, a zwłaszcza dla pierwszego 1000 iteracji. Ponadto, analiza wyników obliczeniowych pokazuje, że algorytm NEH (zastosowany w celu znalezienia początkowego rozwiązania) daje zadowalające rezultaty, chociaż dla większych wartości n oraz lm nie są one zbyt imponujące. Stąd też w przyszłych badaniach należałoby rozważyć możliwość zastosowania innych specjalistycznych algorytmów heurystycznych znajdujących rozwiązanie początkowe.

LITERATURA

1. Grabowski J., Skubalska E., Smutnicki C. : On flow-shop with release and due dates to minimize maximum lateness. *Journal of Operational Research Society 34,1983,278-285.*
2. Hitomi K., Ham I.: Group scheduling for multiproduct, multistage manufacturing systems. *Transactions ASME - Journal of Engineering for Industry 1977,759-765.*
3. Hitomi K.,Ham I.: Machine loading and product-mix analysis for group technology. *Transactions ASME - Journal of Engineering for Industry 100, 1978,370-374.*

4. Monma C.L., Potts C.N.: On the complexity of scheduling with batch set-up times. *Operation Research 37,1989, 798-804.*
5. Nowicki E., Smutnicki C. : A fast tabu search algorithm for the permutation flow shop problem. *European Journal of Operational Research 91,1996,160-175.*
6. Potts C.N., Van Wassenhove L.N. : Integrating scheduling with batching and lot-sizing : a review of algorithms and complexity. *Journal of Operational Research Society 43,1992, 395-406.*
7. Sawik T.: Planowanie i sterowanie produkcji w elastycznych systemach montażowych. *WNT, Warszawa 1996.*
8. Taillard E.: Some effeicient heuristic methods for flow shop sequencing. *European Journal of Operational Research 47,1990,65-74.*
9. Zdrzałka S.: Przepływowe problemy harmonogramowania z przezbrojeniami maszyn. *Raport serii: Preprinty nr 7/95 ICT Politechniki Wrocławskiej 1995.*
10. Zdrzałka S.: Algorytmy dla niektórych zagadnień szeregowania zadań z przezbrojeniami maszyn. *Raport serii: Preprinty nr 18/94 ICT Politechniki Wrocławskiej 1994.*

Recenzent: Dr hab.inż. Mirosław Zaborowski, prof.Pol.Śl.

Abstract

This paper deals with the assembly flow-shop problem which arises in production of jobs which are assembled with several elements, so that, the structure of each job can be presented by a graph-tree. The problem arises to determine a sequence of the batches of jobs, taking into account technological requirements, that maximum completion time of jobs is minimized. To solve the problem, we propose an approximation algorithm based on the tabu search approach.
The computational results are presented.