ZESZYTY NAUKOWE POLITECHNIKI ŚLASKIEJ 2000
Seria: AUTOMATYKA z. 129 Nr kol. 1474

Józef GRABOWSKI, Aneta MARCHEWKA
Politechnika Wrocławska

ZAGADNIENIE SZEREGOWANIA WYROBÓW W PRZEPŁYWOWYCH PROCESACH MONTAŻOWYCH. ALGORYTM EWOLUCYJNY

Streszczenie. W pracy przedstawia się zagadnienie szeregowania wyrobów w przepływowym procesie produkcyjnym. Proces montażu oraz proces produkcyjny są rozpatrywane jednocześnie. Jednoczesne rozpatrywanie obu procesów wymusza zamodelowanie struktury wyrobu za pomocą grafu – drzewa.

SCHEDULING OF JOBS IN ASSEMBLY FLOW-SHOP PROBLEM. EVOLUTION ALGORITHM

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. Opis zagadnienia

W niniejszej pracy rozważamy dyskretny przepływowy proces produkcyjny wraz z procesem montażu. W tego rodzaju zagadnieniach struktura wyrobu jest modelowana za pomocą grafu–drzewa, w którym poszczególne węzły reprezentują operacje technologiczne oraz montażowe, natomiast droga pomiędzy wierzchołkiem początkowym a końcowym „przedstawia" daną linię technologiczną. Oznacza to, że część operacji technologicznych dla danego wyrobu może być wykonywana równocześnie – taka sytuacja ma miejsce wówczas, gdy dane operacje należą do różnych linii technologicznych, natomiast operacje, które należą do danej linii technologicznej, wykonywane są szeregowo, czyli rozpoczęcie wykonywania danej operacji technologicznej jest możliwe wtedy, gdy operacja ją poprzedzająca została wykonana. Ponadto, każda operacja montażowa może rozpocząć się dopiero wtedy, gdy wszystkie elementy składowe wyrobu dla danej operacji montażowej są już dostępne i zostały dostarczone z produkcji lub innych stanowisk montażowych. Zagadnienie optymalizacji procesu sprowadza się do wyznaczenia kolejności realizacji wyrobów i ich elementów składowych, aby minimalizować przyjęte kryterium optymalizacji.

2. Model matematyczny zagadnienia

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

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

Każda partia składa się ze zbioru jednakowych wyrobów W_i. Liczba wyrobów partii P_i wynosi n_i, czyli $|P_i| = n_i$. Dla każdej partii P_i jest podany żądany termin zakończenia jej realizacji D_i.

Ogólnie, każdy wyrób W_i może być przedstawiony w postaci grafu-drzewa (rys.1):

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

gdzie: $N_i = \{O_1^i, O_2^i, ..., O_{n_i}^i\}$ jest zbiorem wszystkich operacji (obróbczych i montażowych), które należy zrealizować w trakcie produkcji wyrobu W_i, natomiast PT_i jest zbiorem relacji (par operacji) reprezentującym 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}.

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, a 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:

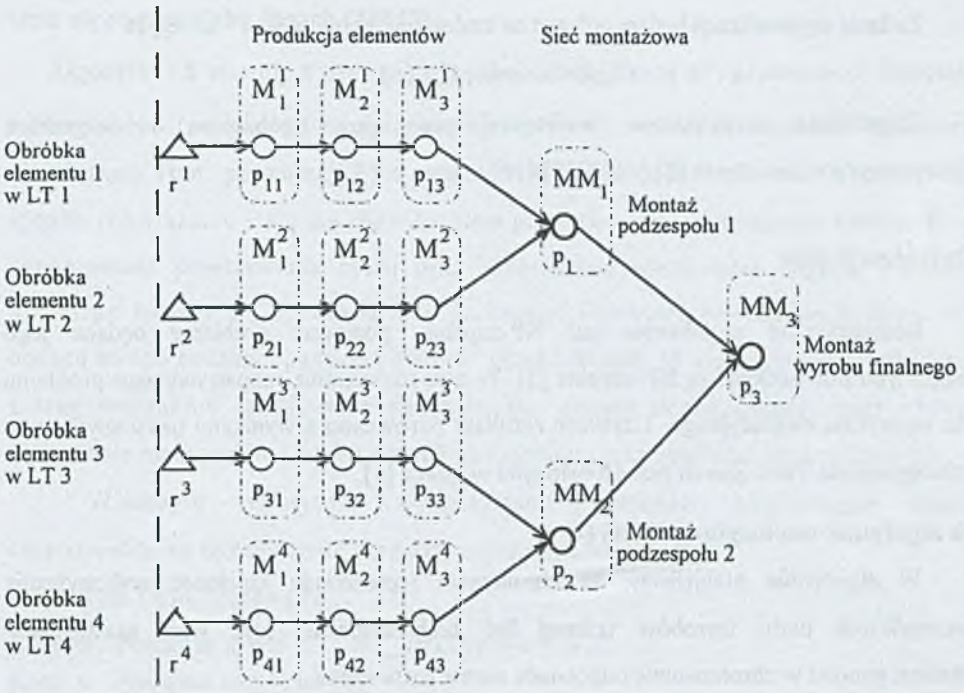
$$MOB = \{M_1^1, M_2^1, ..., M_{m_1}^1, M_1^2, M_2^2, ..., M_{m_2}^2, ..., M_1^t, M_2^t, ..., M_{m_t}^t\},$$

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.



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

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

Rys.1. Model wyrobu
Fig.1. An example of job

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_i^k oraz π_j oznaczają odpowiednio permutację określającą kolejność wykonywania partii $P = \{P_1, P_2, ...P_n\}$, na M_i^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],[8],[9],[10].

3. Opis algorytmów

Rozpatrywane zagadnienie jest NP-zupełne, ponieważ problemy będące jego szczególnymi przypadkami są NP-zupełne [5]. W celu rozwiązania rozpatrywanego problemu użyto algorytmu ewolucyjnego. Uzyskane rezultaty porównano z wynikami uzyskanymi przy użyciu algorytmu Tabu Search przedstawionymi w pracy [1].

Opis algorytmu ewolucyjnego (GA) [4]

W algorytmie przyjęliśmy, że chromosom reprezentuje kolejność wykonywania poszczególnych partii wyrobów w danej linii technologicznej oraz sieci montażowej. Natomiast genowi w chromosomie odpowiada numer partii zadań.

Poszczególne kroki algorytmu:

1. Ustal populację początkową, *iter*:=0.
2. Oceń poszczególne chromosomy i dokonaj selekcji;
   - oblicz wartość dopasowania *eval(v$_i$)* dla *i=1..pop*, gdzie *pop* - wielkość populacji, *eval(v$_i$)* – wartość funkcji celu,
   - oblicz całkowite dopasowanie populacji $F = \sum_{i=1}^{pop} eval(v_i)$ dla *i=1..pop*,
   - oblicz prawdopodobieństwo wyboru $p_i = eval(v_i)/F$ dla *i=1..pop*,
   - oblicz dystrybuantę $q_i = \sum_{j=1}^{i} p_j$ dla *i=1..pop*,
   - wygeneruj liczbę przypadkową *r* z zakresu [0,1],
   - wybierz chromosom v_i, dla którego $q_{i-1} < r \leq q_i$.
3. Wybierz (losowo) i dobierz w pary chromosomy do krzyżowania. Dla każdej pary wybranych chromosomów wygeneruj punkt przecięcia i utwórz nowe chromosomy pobierając dla pierwszego potomka pierwszy odcinek łańcucha od pierwszego rodzica i drugi odcinek łańcucha od drugiego rodzica, przetwarzając według ciągu odwzorowań określone geny. Drugi potomek powstaje poprzez połączenie pierwszej części łańcucha drugiego rodzica z drugą częścią łańcucha pierwszego rodzica.
4. Wybierz losowo chromosomy do mutacji. Wylosuj dwa geny, które należy wymienić miejscami w danym chromosomie.
5. Jeżeli *iter>maxiter*, to stop, w przeciwnym przypadku przejdź do kroku 2.

Opis algorytmu Tabu Search (TS) [7]

Algorytm TS startuje z pewnej permutacji początkowej π^o, przeszukuje wszystkich jej sąsiadów (poprzez przesunięcie zadań w π^o) w celu znalezienia 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.

Algorytm Tabu Search (TS)

Krok 0 . Podstaw $\pi = \pi^*, C^* = C_{max}(\pi), iter := 0, T = \emptyset$.

Krok 1 . Podstaw *iter*:=*iter*+1;
wyznacz ruch v, dla którego $C_{max}(\pi_v) = \min_{w \in NT}(C_{max}(\pi_w))$ i który nie jest ruchem zabronionym. Jeżeli w=0, to usuń najstarszy element z listy tabu T i przejdź do kroku 1.

Krok 2. .Jeżeli $C_{max}(\pi_v) < C^*$, to zapamiętaj $C^* = C_{max}(\pi_v), \pi^* = \pi_v$.
Dołącz v do listy tabu T.

Krok 3. Jeżeli *iter*≥*maxiter*, to stop. W przeciwnym przypadku idź do kroku 1.

Sąsiedztwo dla algorytmu generowane jest przez ruch $v=\{h,a,b\}$ polegający na usunięciu partii z pozycji a i wstawieniu na pozycję b w permutacji π_h, przy czym usuwane są tylko partie ze ścieżki krytycznej. Przy generowaniu sąsiedztwa nie wykonuje się ruchów polegających na przesuwaniu partii wewnątrz bloku.

W celu wyznaczenia początkowej permutacji π^o (populacji początkowej) założyliśmy, że dla linii technologicznych oraz stanowisk montażowych uszeregowanie wyrobów jest takie samo, czyli $\pi^o = \pi^1 = \pi^2 = ... = \pi^I = \pi_1 = ... = \pi_m$. W celu znalezienia π^o – permutacji początkowej (populacji początkowej) wykorzystaliśmy odpowiednio zmodyfikowany algorytm NEH [5]. Podstawowym elementem tego algorytmu jest lista priorytetów dla wyrobów. Dla potrzeb 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,(uszeregowania) dające najmniejszą wartość terminu zakończenia wykonywania powyższych wyrobów. Po wykonaniu n głównych kroków otrzymujemy poszukiwaną permutację początkową π^0 (populację początkową).

4. Wyniki obliczeniowe

Algorytm ewolucyjny został zaprogramowany w języku Delphi i uruchomiony na komputerze PC Pentium II/200 MHz. Algorytm Tabu Search zaprogramowano w języku Turbo Pascal i uruchomiono na komputerze IBM RISC System/6000, 200 MHz. Obydwa algorytmy zostały przetestowane dla tych samych danych, generowanych losowo. Wartości czasów realizacji poszczególnych operacji dla n partii wygenerowano 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 $lm=3t+t-1=4t-1$ 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, ... , 31×40, 31×80, wygenerowano 10 przykładów testujących.

Dla każdego przykładu wyznaczono następujące wielkości:

Co - wartość funkcji celu dla permutacji początkowej,

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

CB - najlepsza wartość funkcji celu otrzymana w ciągu pierwszych 10000 iteracji.

Na podstawie tych wartości wyznaczyliśmy $PR(x) = 100\% (Co - Cx)/Co$ – średnią (procentowa) wartość poprawy funkcji celu względem wartości początkowej *Co*, $x \in \{A,B\}$.

W tabeli 1 przedstawiono wyniki obliczeniowe.

Tabela 1

Wyniki obliczeniowe

	Algorytm ewolucyjny (GA)		*Algorytm Tabu Search (TS)*	
lm × n	*PR(A)*	*PR(B)*	*PR(A)*	*PR(B)*
7 ×10	2,14	2,83	6,29	6,29
7×20	2,14	3,74	7,41	8,35
7×40	2,19	4,48	10,44	10,61
7×80	2,49	6,29	11,92	11,92
15×10	2,10	3,40	12,40	12,42
15×20	2,83	4,78	14,00	14,90
15×40	2,86	5,65	14,15	14,19
15×80	3,21	5,96	14,99	15,08
31×10	2,23	5,27	17,21	17,21
31×20	3,30	5,60	17,70	17,77
31×40	3,58	6,10	20,23	20,23
31×80	4,20	8,58	20,93	20,94

Obydwa algorytmy obliczenia rozpoczynały startując z podobnych rozwiązań początkowych. Dla algorytmu Tabu Search była to permutacja początkowa uzyskana za pomocą zmodyfikowanego algorytmu NEH, a dla algorytmu ewolucyjnego była to populacja początkowa uzyskana również za pomocą algorytmu NEH. Analiza poprawy wartości funkcji celu zarówno dla pierwszych tysiąca iteracji czy też dla całości przebiegu algorytmu wskazuje na wyraźną przewagę algorytmu typu Tabu Search. Algorytm ewolucyjny rozwiązanie początkowe poprawiał w granicach od 2,8 do 8,5%, natomiast poprawa wartości początkowej dla algorytm typu Tabu Search wahała się od 6 do 20%. Przy czym tak znaczącą poprawę algorytm ewolucyjny najczęściej osiągał po pełnym przebiegu, natomiast za pomocą algorytmu Tabu Search już po 1000 iteracji osiągaliśmy zadowalające rezultaty. Na tak wyraźne przyśpieszenie obliczeń w przypadku algorytmu Tabu Search ma wpływ ograniczenie sąsiedztwa przeszukiwań poprzez wykorzystanie własności eliminacyjnych bloków. Opierając się na wynikach obliczeniowych można stwierdzić, że dla zagadnień montażowych algorytm oparty na technice Tabu Search daje korzystniejsze rezultaty niż algorytm ewolucyjny.

LITERATURA

1. Grabowski J., Marchewka A.: Zagadnienie szeregowania wyrobów w przepływowych procesach montażowych. Zeszyty Naukowe Politechniki Śląskiej, s. Automatyka z. 124, Gliwice 1998, 29-36.
2. Hitomi K., Ham I.: Group scheduling for multiproduct, multistage manufacturing systems. Transaction ASME – Journal 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 100,370-374, 1978.
4. Michalewicz Z.: Algorytmy genetyczne + struktury danych = programy ewolucyjne. WNT, Warszawa 1996.
5. Monma C.L., Potts C.N.: On the complexity of scheduling with batch set-up times. Operation Research 37, 798-804, 1989.
6. Nawaz M., Enscore Jr.E.E.,Ham I., A heuristic algorithm for m-machine, n-job flow-shop scheduling problem. OMEGA International Journal of Management Science, 1983, 11,91-95.
7 Nowicki E., Smutnicki C.: A fast tabu search algorithm for the permutation flow shop problem. European Journal of Operational Research 91,160-175,1996.
8. 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, 395-406, 1992.
9. Sawik T.: Planowanie i sterowanie produkcji w elastycznych systemach montażowych. WNT, Warszawa, 1996.
10. Zdrzałka S.: Przepływowe problemy harmonogramowania z przezbrojeniami maszyn. Raport serii: Preprinty nr 7/95 ICT Politechniki Wrocławskiej 1995.

Recenzent: Prof.dr hab.inż. T.Sawik

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 evolution algorithm. We compare results which results of algorithm based on the tabu search approach. The computational results are presented.