ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2006
Seria: AUTOMATYKA z. 144 Nr kol. 1727

Arkadiusz ANTCZAK, Paweł ANTCZAK, Tadeusz WITKOWSKI
Politechnika Warszawska

ZASTOSOWANIE PROCEDURY GRASP DO HARMONOGRAMOWANIA PRODUKCJI MAŁOSERYJNEJ

Streszczenie. W pracy przedstawiono zadanie planowania produkcji małoseryjnej typu gniazdowo–otwartego. Do rozwiązania tej klasy zadań opracowano algorytm na podstawie procedury GRASP (Greedy Randomized Adaptive Search Procedure). Dla przedstawionego algorytmu opisano wyniki eksperymentu komputerowego.

USED GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE (GRASP) FOR FLEXIBLE JOB SHOP SCHEDULING PROBLEM

Summary. The paper presents task formulation job shop – open shop problem. The algorithm based on the on procedure GRASP, has been prepared for the class of mentioned task. The results of the computer experiment for the algorithm has been presented.

1. Wprowadzenie

Do klasycznych problemów szeregowania zadań produkcyjnych zalicza się problemy przepływowe (flow shop), gniazdowe (job shop) i otwarte (open shop). Wymienione problemy szeregowania można ogólnie określić następująco. W systemie produkcyjnym znajduje się określona liczba maszyn, na których należy wykonać określoną liczbę zadań produkcyjnych. Każde zadanie produkcyjne składa się z pewnej liczby operacji. Do danej operacji przyporządkowana jest dokładnie jedna maszyna, na której należy ją wykonać bez przerw w zadanym czasie. W danej chwili zadanie może być wykonywane tylko na jednej maszynie. W problemie przepływowym i gniazdowym przyjmuje się, że kolejność wykonywania operacji danego zadania jest ustalona w odróżnieniu od problemu otwartego.

Przydatność narzędzi harmonogramowania do analizy odpowiednich modeli produkcyjnych zależy od wielkości i charakteru produkcji. Podstawowe metody rozwiązywania zadania harmonogramowania produkcji małoseryjnej dzielimy na przybliżone i dokładne. Do tego typu zadań w praktyce wykorzystuje się właściwie tylko metody przybliżone, do których zaliczamy m.in. algorytm GRASP wykorzystywany w danej pracy.

2. Sformułowanie problemu

Przedstawimy formalne sformułowanie rozwiązywanego problemu. Może on być opisany w następujący sposób. Określono zbiór maszyn M, zbiór operacji O, elementami którego są poszczególne technologiczne operacje. Każdej operacji $\sigma \in O$ przyporządkowano podzbiór maszyn $M(\sigma) \in M$, które mogą je wykonywać. Zbiór O – zbiór częściowo uporządkowany, tj. określono zbiór następstwa kolejności wykonania $C = \{\sigma \prec \delta\}$, który określa kolejność wykonania operacji ($«\sigma \prec \delta»$ oznacza, że operacja σ powinna być wykonana przed rozpoczęciem wykonania operacji δ). Na zbiorze O określono także relację ekwiwalentności φ. Operacje związane relacją φ nazywamy operacjami ekwiwalentnymi. Relacja φ pozwala podzielić zbiór O na klasy ekwiwalentne. Oznaczmy $K(\sigma)$ – klasa elementów ze zbioru O ekwiwalentnych σ. Niech funkcja:

$$\chi(\sigma,\delta) = \begin{cases} 1, \textit{ jeże li } \sigma \notin K(\delta) \\ 0, \textit{ w przeciwnym przypadku} \end{cases}$$

Dla operacji σ są określone: $p(\sigma)$ – liczba jednostek czasu niezbędna do jej wykonania (jednakowa dla wszystkich maszyn), $t(\sigma)$ – liczba jednostek czasu niezbędna dla przezbrajania maszyny przed wykonaniem tej operacji, gdzie $p(\sigma)$, $t(\sigma) \geq 0$, przy czym $\forall \sigma \in O$. Problem określenia harmonogramu polega na tym, aby dla każdej operacji $\sigma \in O$ wybrać maszynę ze zbioru M i następnie określić porządek wykonania operacji na maszynach z M w ten sposób, aby sumaryczny maksymalny czas wykonania prac był minimalny. Niech $S(\sigma)$ – liczba jednostek czasu od początku wykonania prac do początku wykonywania operacji σ, $m(\sigma)$ – maszyna wybrana dla wykonania operacji σ. Wtedy matematyczne sformułowanie problemu można przedstawić następująco:

$$\min \max_{\sigma \in O}[S(\sigma) + p(\sigma)], \tag{1}$$

przy ograniczeniach:

$$S(\sigma) + p(\sigma) \leq S(\delta) \quad \forall \sigma,\delta \in O, \sigma \prec \delta, \tag{2}$$

$$S(\sigma) \geq t(\sigma); \quad S(\sigma) + p(\sigma) \leq S(\delta) - \chi(\sigma,\delta)t(\delta) \vee S(\delta) + p(\delta) \leq S(\sigma) - \chi(\sigma,\delta)t(\sigma), \tag{3}$$

$$\forall \sigma,\delta \in O, m(\sigma) = m(\delta), \tag{4}$$

$$m(\sigma) \in M(\sigma) \; \forall \sigma \in O. \tag{5}$$

Zależność (1) określa jako kryterium optymalności kryterium Johnsona; (2) określa ograniczenia dotyczące kolejności wykonania operacji zgodnie z procesem technologicznym. Zależność (3) wymaga uwzględnienia czasu wykonania przezbrojenia maszyny przed rozpoczęciem operacji. Ograniczenia na zasoby (maszyna może jednocześnie wykonywać tylko jedną operację) przedstawiają zależności (4), które uwzględniają także czas niezbędny na przezbrajanie maszyn przy wykonywaniu

operacji. Zależność (5) wymaga, aby operacje wykonywane były na przeznaczonych do tego celu maszynach.

Charakterystyczną cechą rozpatrywanego problemu jest to, że kolejność wykonywania operacji danego zadania w pewnej fazie jego realizacji jest nieustalona w odróżnieniu od realizacji operacji w problemie gniazdowym. W danej pracy jako kryterium optymalności przyjęto kryterium Johnsona (minimalny sumaryczny czas wykonania prac).

Dla każdej $\sigma \in O$ określimy zbiór $JP(\sigma)$ zawierający operacje $\delta \in O$ takie, że $\delta \prec \sigma$ i nie istnieje operacja $\omega \in O$, taka, że $\delta \prec \omega \prec \sigma$. Jeżeli określono porządek wykonania operacji, to można wykorzystać oznaczenie $MP(\sigma)$ dla operacji, która wykonywana jest bezpośrednio przed $\sigma \in O$, tj. między wykonaniem $MP(\sigma)$ i σ maszyna $m(\sigma)$ nie wykonuje żadnych innych operacji. Niech $JS(\sigma)$ – zbiór operacji $\delta \in O$, dla których $\sigma \in JP(\sigma)$; $MS(\sigma)$ – operacja, dla której $MP(MS(\sigma)) = \sigma$.

3. Metaheurystyczna procedura GRASP

W pracy wykorzystano schemat procedury GRASP [2] dla opracowania algorytmu rozwiązania zadania (1)-(5). Procedura GRASP składa się z dwóch podstawowych etapów: konstruowanie początkowego rozwiązania – etap I i przeszukiwania lokalnego – etap II. Na etapie konstruowania początkowego rozwiązania generowane jest dopuszczalne rozwiązanie dla zadania (1)-(5) i jego sąsiedztwo badane jest na etapie przeszukiwania lokalnego. Te etapy są powtarzane, dopóki nie będzie spełnione kryterium zatrzymania. Najlepsze znalezione rozwiązanie spośród wszystkich iteracji zwracane jest jako wynik zastosowania procedury.

Dla otrzymania dopuszczalnego rozwiązania należy określić wielkości $S(\sigma)$ i $m(\sigma)$ tak, aby spełnione były ograniczenia zadania. Na etapie konstruowania początkowego rozwiązania najpierw tworzone jest rozwiązanie spełniające ograniczenia zadania. Dla konstrukcji takiego rozwiązania wykorzystuje się pojęcie „kandydata do wstawienia". Dla zadania (1)-(5) kandydatem do wstawienia jest para: operacja i numer maszyny, na którą ona pretenduje.

Oznaczmy *RCL* i *CL*, listy kandydatów do wstawienia, przy czym w liście *CL* będą przechowywani kandydaci już wstawieni do częściowego harmonogramu, a w liście *RCL* kandydaci pretendujący do wstawienia do tego harmonogramu; $F(CL)$ – wartość kryterium Johnsona dla częściowego harmonogramu określonego listą *CL*. Przeszukiwanie lokalne kolejno zamienia rozwiązanie bieżące na rozwiązanie lepsze z jego sąsiedztwa (lepsze rozwiązanie rozumiemy w sensie kryterium Johnsona). Etap przeszukiwania lokalnego kończy się, kiedy w sąsiedztwie nie zostało znalezione lepsze rozwiązanie.

Poniżej przedstawiono algorytm konstrukcji rozwiązania początkowego (etap I) procedury GRASP, wynikiem której jest dopuszczalne rozwiązanie zadania (1)-(5).

1: $CL = RCL = \emptyset$
2: while $CL \neq |O|$ do
3: for all $\delta \in O$ do
4: if $[\exists(\delta,k) \in CL, \forall k \in M] \wedge$
$\wedge [\exists(\omega,l) \in CL, l \in M, \forall \omega \in JP(\delta)]$ then
5: for all $k \in M(\delta)$ do
6: $RCL = RCL \cup (\delta,k)$
7: wartość $[(\delta,k)] = F(CL \cup (\delta,k))$
8: end for
9: end if
10: end for
11: $LB = \min_{(\delta,k) \in RCL}$ [wartość $[(\delta,k)]$]
12: $UB = \max_{(\delta,k) \in RCL}$ [wartość $[(\delta,k)]$]
13: α - liczba losowa z wartość $[0,1]$
14: Wybrać losowego kandydata (δ,k) ze zbioru Set $= \{(\delta,l) : (\delta,k) \in RCL,$ wartość $[(\delta,k)]] \leq LB + \alpha(UB - LB)\}$
15: wartość $CL = CL \cup (\sigma,k)$
16: $RCL = \emptyset$
17: end while
18: Zwrócić wartość $F(CL)$

4. Przeszukiwanie lokalne

Na etapie przeszukiwania lokalnego było użyte sąsiedztwo przedstawione w [2,4], które wykorzystuje pojęcie drogi krytycznej. Przyporządkowujemy każdej operacji $\sigma \in O$ zbiór $JS(\sigma)$, składający się z operacji σ takich, że $\sigma \in JP(\sigma)$ i operację $MS(\sigma)$ taką że $MP(MS(\sigma))$, jeżeli taka operacja istnieje.

Operacja $\sigma \in O$ jest krytyczna, jeżeli $F(\sigma) + tail(\sigma) = F(CL)$, przy czym $tail(\sigma)$ określa, ile czasu jest wymagane do zakończenia wszystkich operacji, które nie mogą być rozpoczęte do zakończenia wykonania operacji σ, zgodnie z ograniczeniami technologicznymi. Operacje krytyczne to te, które nie posiadają rezerwy czasowej (luzu czasowego). Droga krytyczna jest to zbiór operacji krytycznych, które nie są wykonywane jednocześnie. Zgodnie z tym określeniem może istnieć kilka krytycznych dróg. Przy realizacji przeszukiwania lokalnego wykorzystywana była jedna droga krytyczna. Blokiem krytycznym CB nazywa się zbiór operacji krytycznych $\sigma \in O$, które wykonuje się na jednej maszynie bez przerw. W ten sposób zbiór CB można uporządkować względem czasów początku wykonania operacji. Operacja z najmniejszym (największym) czasem rozpoczęcia wykonania nazywa się pierwszą (ostatnią) operacją bloku krytycznego.

Sąsiedztwo przedstawione w [4] składa się z rozwiązań otrzymanych z rozwiązania początkowego (wyjściowego) na drodze zmiany porządku dwóch pierwszych lub dwóch ostatnich operacji w uporządkowanych zbiorach bloków krytycznych. Sąsiedztwo tego typu dla zadania (1)-(5) traci jedną ważną właściwość w porównaniu z zadaniem *job shop scheduling*: rozwiązania z jego sąsiedztwa nie zawsze są dopuszczalne. Dlatego przeprowadzane było sprawdzenie rozwiązania ze względu na dopuszczalność rozwiązania.

5. Eksperyment komputerowy

Dla rozwiązania zadania (1)-(5) opracowano oprogramowanie dla algorytmu, opartego na procedurze GRASP [2]. Eksperymenty przeprowadzono komputerowo

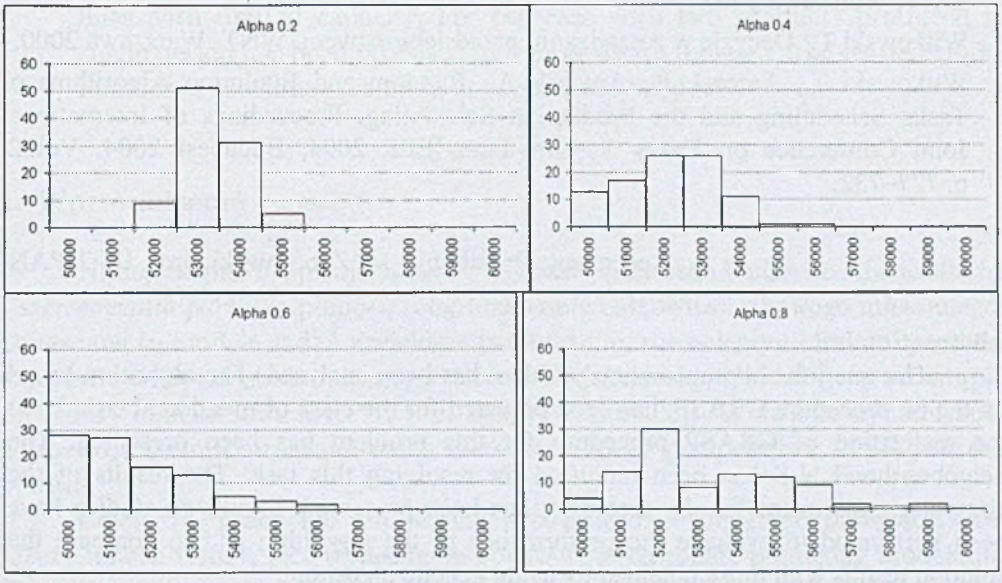
dla danych przedstawionych w [6]. Liczba operacji dla tych danych – 160, liczba maszyn – 26. Wszystkie eksperymenty przeprowadzono z wykorzystaniem komputera z procesorem Pentium 733MHz i 256MB pamięci operacyjnej.

Parametr α, który wykorzystuje się do wyboru kandydatów do wstawienia do harmonogramu na etapie konstrukcji rozwiązania, wybierany był dla każdego etapu konstrukcji losowo z przedziału [0,1]. Najlepsze rozwiązanie dla przebiegu szeregowo-równoległego, równe 33774 min (tab. 1.) znaleziono po 60 iteracjach, przeprowadzając 5 prób dla każdej liczby iteracji. Dla rozwiązania tego zadania zastosowano także algorytm genetyczny, przy czym wyniki były nieznacznie lepsze [7]. Wyniki dla GRASP do szeregowego przebiegu produkcji przedstawiono na rysunku 1.

Tabela 1

Wartości F(H) w zależności od liczby iteracji dla algorytmu GRASP

Liczba iteracji	1	2	3	4	5	6	7	8	9	10	20	30	40	50	60		70	80
Próba dająca najlepszą wartość Fi(H) (i-liczba prób, i=1...5)											(2)		(1)		(3)	(4)	(5)	
Wartość kryterium optymalizacji Fi(H) [min]											33730		33577		33851	33491	33633	
Średnia wartość kryterium Fi/5 [min]	34716	34643	34681	34265	34328	34263	34294	34093	34140	34177	33974	34037	33826	33820	**33774**		33877	33814



Rys. 1. Porównanie wyników eksperymentu dla różnych wartości parametru α

6. Wnioski

Dla problemu gniazdowootwartego szeregowania zadań przedstawiono algorytm bazujący na procedurze GRASP. Eksperyment obliczeniowy pokazał, że najbardziej pracochłonną częścią algorytmu jest etap konstrukcji rozwiązania. Głównymi wadami realizowanego algorytmu jest niezależność poszczególnych iteracji i wysoka pracochłonność konstrukcji rozwiązania. Pierwszą wadę można usunąć, wprowadzając do algorytmu schemat intensyfikacji poszukiwania rozwiązania w obszarze najlepszego rozwiązania, tak jak to zrealizowano w [6] lub stosując procedurę path-relinking [1]. W celu usunięcia drugiej wady przy rozwiązywaniu sformułowanego zadania można zastosować inne metody, np. typu TABU, która nie wymaga stałej generacji nowych rozwiązań.

LITERATURA

1. Aiex R.M., Binato S., Resende M.G.C.: Parallel GRASP with path-relinking for job shop scheduling. Parallel Computing, Vol. 29, 2003, p. 393–430.
2. Binato S. i in.: A Greedy Randomized Adaptive Search Procedure For Job Shop Scheduling. AT&T Labs Technical Report, 2000, p.1-19.
3. Janiak A.: Wybrane problemy i algorytmy szeregowania zadań i rozdziału zasobów. Akademicka Oficyna Wydawnicza PLJ, Warszawa 1999.
4. Nowicki E., Smutnicki Cz.: A Fast Taboo Search Algorithm for the Job Shop Problem. Management Science, Vol. 42, No 6, 1996, p. 797–813.
5. Smutnicki Cz.: Algorytmy szeregowania. Akademicka Oficyna Wydawnicza EXIT, Warszawa 2000.
6. Witkowski T.: Decyzje w zarządzaniu przedsiębiorstwem. WNT, Warszawa 2000.
7. Witkowski T., Antczak P., Antczak A.: Random and Evolution Algorithms of Tasks Scheduling and the Production Scheduling. Proceedings of International Joint Conference on Fuzzy Systems-Fuzz-IEEE 2004, Budapest 2004, Vol. 2, p. 727–732.

Recenzent: Dr hab. inż: M. Zaborowski, prof. IISiT PAN

Abstract

The one job shop-open shop problem has been analysed. The algorithm based on the on procedure GRASP, has been prepared for the class of mentioned task. Next the realization of GRASP procedure for this problem has been presented. The neighbourhood N-S has been modified for resolving this task. The results of the computer experiment for the algorithm has been presented. Simulated studies have been performed to evaluate the performance of the algorithm and to compare the solution results with those obtained by using genetic algorithm.