ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2002
Seria: AUTOMATYKA z. 134 Nr kol. 1554

Wojciech BOŻEJKO[1], Mieczysław WODECKI[2]
[1]Politechnika Wrocławska
[2]Uniwersytet Wrocławski

PERMUTACYJNY PROBLEM PRZEPŁYWOWY. ALGORYTMY RÓWNOLEGŁE SYMULOWANEGO WYŻARZANIA

Streszczenie. W pracy rozpatrywany jest permutacyjny problem przepływowy z minimalizacją czasu wykonywania zadań. Przedstawiamy algorytmy (sekwencyjny i równoległy) oparte na metodzie symulowanego wyżarzania. W ich konstrukcji wykorzystano idee bloków z drogi krytycznej oraz dolne oszacowania wartości funkcji celu, a także różne schematy schładzania oraz funkcje akceptacji. Algorytmy testujemy na przykładach zaczerpniętych z pracy Taillarda [22].

PERMUTATION FLOW SHOP PROBLEM. PARALLEL SIMULATED ANNEALING ALGORITMS

Summary. This paper deals with the classic permutation flow shop scheduling problem with the make-span criterion. We describe an approximation algorithms (sequential and parallel) based on simulated annealing method. We research various accepting functions and cooling schedules.We propose neighbourhood using so called blocks of jobs on a critical path and also using lower bound of cost function.

1. Wprowadzenie

W permutacyjnym problemie przepływowym (ang. *permutation flow shop*) dany jest ustalony zbiór zadań oraz zbiór maszyn. Każde zadanie należy kolejno wykonać, bez przerywania, na każdej maszynie w określonym czasie. Problem polega na wyznaczeniu kolejności wykonywania zadań (takiej samej na każdej maszynie), która minimalizuje czas zakończenia wszystkich zadań ($C_{\max}$). W literaturze problem ten jest oznaczany przez $F|m|C_{\max}$. Dla dwóch maszyn ($F|2|C_{\max}$) można go rozwiązać korzystając z algorytmu Johnsona [11] o złożoności obliczeniowej $O(n\log n)$. Garey, Johnson oraz Seti [5] udowodnili, że gdy liczba maszyn jest większa lub równa 3, wówczas problem ten jest silnie *NP*-zupełny. Algorytmy dokładne dla

jego rozwiązywania, bazujące na metodzie podziału i ograniczeń ([9], [13], [6]), pozwalają na rozwiązywanie jedynie niewielkich przykładów. W ostatnich latach opublikowano wiele bardzo dobrych algorytmów przybliżonych (głównie metaheurystycznych) opartych na metodzie poszukiwania z tabu ([21], [19] i [16]), symulowanego wyżarzania ([18], [17]) oraz algorytmu genetycznego ([10], [20]). Efektywność tych algorytmów (jakość wyznaczanych przez nie rozwiązań) zależy od ich parametrów oraz w dużym stopniu od liczby wykonanych iteracji (czasu obliczeń). Głównym sposobem skracania tego czasu jest stosowanie obliczeń równoległych. Nie każdy algorytm sekwencyjny da się jednak zrównoleglić tak, aby uzyskać znaczne przyśpieszenie. Stąd konieczność konstrukcji nowych innych algorytmów bazujących także na dobrych i sprawdzonych metodach sekwencyjnych. W pracy przedstawiamy algorytmy równoległe oparte na metodzie symulowanego wyżarzania, dla rozwiązywania permutacyjne-go problemu przepływowego. Porównujemy główne elementy takiego algorytmu: schemat schładzania i funkcję akceptacji. Badamy ich wpływ na jakość wyznaczanych rozwiązań.

Praca ta stanowi kontynuację badań zamieszczonych w [2]. Składa się z siedmiu rozdziałów. Pierwsze dwa zawierają opis problemu oraz podstawy teoretyczne konstrukcji algorytmów zamieszczonych w rozdziale 4. Następny rozdział zawiera przegląd opisanych w literaturze schematów schładzania oraz funkcji akceptacji, których wyniki porównawcze zamieszczone są w rozdziale 6. Z kolei ostatni rozdział jest posumowaniem całości.

2. Sformułowanie problemu

Korzystając z notacji stosowanej w pracach [16], [7] i [2] omawiany problem można przedstawić następująco. Dany jest zbiór n zadań $J=\{1,2,\ldots,n\}$ oraz m maszyn $M=\{1,2,\ldots,m\}$. Zadanie $j\in J$ jest ciągiem m operacji O_{j1}, O_{j2},..., O_{jm}. Operacja O_{jk} odpowiada procesowi wykonywania, bez przerywania, zadania j na maszynie k. Czas jej wykonywania wynosi p_{jk}. Należy wyznaczyć kolejność zadań, minimalizującą całkowity czas ich wykonywania.

Niech $\pi=(\pi(1), \pi(1),\ldots,\pi(n))$ będzie permutacją zadań ze zbioru $\{1,2,\ldots,n\}$, a Π zbiorem wszystkich takich permutacji. Permutacja $\pi\in\Pi$ określa jednoznacznie kolejność wykonywania zadań na każdej maszynie. Należy więc wyznaczyć permutację $\pi^*\in\Pi$ taką, że:

$$C_{\max}(\pi^*)=\max_{\pi\in\Pi} C_{\max}(\pi),$$

gdzie $C_{\max}(\pi)$ jest czasem zakończenia wykonywania wszystkich zadań. Czas zakończenia wykonywania zadania $\pi(j)$ na maszynie k można wyznaczyć z zależności rekurencyjnej:

$$C_{\pi(j)k}=\max\{C_{\pi(j-1)k}, C_{\pi(j)k-1}\} + p_{\pi(j)k}\},$$

gdzie: $\pi(0)=0$, $C_{0k}=0$, $k=1,2,...,m$, $C_{0j}=0, j=1,2,...,n$. Oczywiście, $C_{\max}(\pi)=C_{\pi(n)m}$.

Dla permutacji $\pi\in\Pi$ definiujemy następujący digraf:

$$D(\pi)=(O, AV(\pi)\cup AH(\pi)),$$

gdzie $O=\{O_{j1}, O_{j2},..., O_{jm}\}$ jest zbiorem wierzchołków, a:

$$AV(\pi)=\bigcup_{i=1}^{m-1}\bigcup_{j=1}^{n}\{(O_{\pi(j)i}, O_{\pi(j)i+1})\},$$

zbiorem łuków zwanych poziomymi oraz:

$$AH(\pi)=\bigcup_{i=1}^{m}\bigcup_{j=1}^{n-1}\{(O_{\pi(j)i}, O_{\pi(j+1)i})\},$$

zbiorem łuków pionowych.

Najdłuższa droga w digrafie $D(\pi)$ z wierzchołka $O_{\pi(1)1}$ do $O_{\pi(n)m}$ zwana jest *drogą kryty-czną* w permutacji π. Można łatwo wykazać, że jej długość jest równa $C_{\max}(\pi)$. Drogę tę (per-mutację π) można rozbić na podciągi zadań B_1, B_2, ..., B_m zwanych *blokami*, przy czym:

a) $B_k=(\pi(f_k),\pi(f_k+1),...,\pi(l_k-1),\pi(l_k))$, $f_k\le l_k, f_1=1, l_k=k$, $\pi(l_k)=\pi(f_{k+1})$, $k=1,2,...,m-1$,

b) B_k zawiera operacje wykonywane na tej samej maszynie, $k=1, 2, ..., m$,

c) każde dwa bloki zawierają operacje wykonywane na różnych maszynach.

Operacje $\pi(f_k)$ oraz $\pi(l_k)$ bloku B_k nazywamy odpowiednio pierwszą i ostatnią operacją bloku. Z twierdzeń zamieszczonych w pracy Grabowskiego [6] wynika, że aby z permutacji π otrzymać permutację o mniejszej długości drogi krytycznej należy pewne zadanie z pewnego bloku przestawić przed pierwsze lub za ostatnie zadanie tego bloku.

Niech $B_k^f=B_k\setminus\{\pi(f_k)\}$ oraz $B_k^l=B_k\setminus\{\pi(l_k)\}$ będą podblokami k-tego bloku odpowiednio bez pierwszego i ostatniego zadania. Dla zadania $j\in B_k^f$ definiujemy:

$$\Delta_k^f(j)=\begin{cases} p_{jk-1}-p_{\pi(f_k)k-1}, & j\ne\pi(l_k),\\ p_{jk-1}-p_{\pi(f_k)k-1}+p_{\pi(l_k-1)k+1}-p_{\pi(j)k+1}, & j=\pi(l_k),\end{cases}$$

oraz podobnie, dla zadania $j\in B_k^l$:

$$\Delta_k^l(j)=\begin{cases} p_{jk+1}-p_{\pi(l_k)k+1}, & j\ne\pi(f_k),\\ p_{\pi(f_k+1)k-1}-p_{\pi(j)k-1}+p_{\pi(j)k+1}-p_{\pi(l_k)k+1}, & j=\pi(f_k),\end{cases}$$

gdzie $k=1,2,\ldots,m$ i $p_{\pi(i)j}=0$, $i>n$, $j<1$ lub $j>k$.

Korzystając z powyższych oznaczeń można wykazać [6], że jeżeli permutację β wygenerowano z π przez przestawienie zadania j, ($j\in B_k$):

a) przed pierwsze zadanie w bloku, to: $C_{\max}(\beta)\geq C_{\max}(\pi)+\Delta_k^f(j)$,

b) za ostatnie zadanie bloku, wówczas: $C_{\max}(\beta)\geq C_{\max}(\pi)+\Delta_k^l(j)$.

Tak więc $C_{\max}(\pi)+\Delta_k^f(j)$ oraz $C_{\max}(\pi)+\Delta_k^l(j)$ jest dolnym ograniczeniem $LB(\beta)$ wartości funkcji celu dla permutacji β.

Powyższe własności (które dokładniej zostały opisane w pracy [2]) są podstawą konstrukcji algorytmów, przedstawionych w dalszej części pracy, rozwiązywania permutacyjnego problemu szeregowania.

3. Metoda symulowanego wyżarzania

Ze względu na prostotę implementacji, a jednocześnie jej uniwersalizm, metoda ta jest z powodzeniem stosowana do rozwiązywania wielu problemów optymalizacyjnych. Po raz pierwszy zastosowali ją Kirkpatrick [12] oraz Černý [3]. Jej idea pochodzi z termodynamiki, stąd w jej opisie używa się pojęć z tej właśnie dziedziny. Polega ona na iteracyjnym polepszaniu bieżącego rozwiązania poprzez lokalne przeszukiwanie. Rozpoczyna się od pewnego rozwiązania początkowego (startowego). Następnie generuje się jego otoczenie (sąsiedztwo) oraz wyznacza pewne rozwiązanie z tego otoczenia, które przyjmuje się za rozwiązanie startowe w następnej iteracji. Podstawowym elementem algorytmu wykorzystującego metodę symulowanego wyżarzania jest *funkcja akceptacji* określająca prawdopodobieństwo, z jakim są przyjmowane (akceptowane) za rozwiązania startowe elementy zbioru rozwiązań dopuszczalnych. Jest ona zależna od parametru zwanego temperaturą, który zmienia się w trakcie działania algorytmu zgodnie z tzw. *schematem chłodzenia*.

Niech $\pi\in\Pi$ będzie dowolną permutacją (startową), $N(\pi)$ jej otoczeniem, a π^* najlepszym do tej pory znalezionym rozwiązaniem (na początek przyjmujemy za π^* permutację π). Przez $t=\varphi(i)$ oznaczmy schemat chłodzenia (i – iteracja algorytmu, t – temperatura), a przez $\Psi_t(\pi,\beta)$ funkcję akceptacji.

Standardowy algorytm symulowanego wyżarzania

```
repeat
    while i≤R do
    begin
        i←i+1;
                Wyznaczyć losowo permutację β∈N(π);

                if  Cmax(β) < Cmax (π*)  then  π*←β;
                if  Cmax(β) < Cmax (π) then  π←β
                else
        if  Ψ(π,β)>random[0,1) then  π←β
    end;{i}
        i←0;
    zmodyfikować parametr kontrolny t, zgodnie ze schematem chłodzenia;
    until Warunek zatrzymania;
```

Zmiana temperatury następuje po wykonaniu pewnej liczby iteracji zwanej *pętlą*; w algorytmie jest to parametr R. Najczęściej jest on równy mocy zbioru będącego otoczeniem.

4. Elementy algorytmu symulowanego wyżarzania

Konstrukcja algorytmu bazującego na metodzie symulowanego wyżarzania wymaga określenia pewnych jego elementów oraz ustalenia wartości parametrów.

4.1. Otoczenie

Korzystając z pojęć i oznaczeń wprowadzonych w poprzednim rozdziale, załóżmy, że B_k (k=1,2,...,m) jest k-tym blokiem w permutacji π, a B_k^f i B_k^l jego podblokami. Przez N_k^f oznaczmy zbiór wszystkich permutacji powstałych przez przestawienie dowolnego zadania $j \in B_k^l$ przed blok B_k (tj. przed pierwsze zadanie tego bloku $\pi(f_k)$). Podobnie, przez N_k^l oznaczmy permutacje powstałe przez przestawienie dowolnego zadania $j \in B_k^f$ za blok B_k (za ostatnie zadanie bloku $\pi(l_k)$). Otoczeniem permutacji π jest zbiór:

$$N(\pi) = \bigcup_{k=1}^{m} (N_k^f \cup N_k^l).$$

Liczba elementów zbioru $N(\pi)$ nie przekracza $2n$.

4.2. Funkcje akceptacji

Jeżeli permutacja $\beta \in N(\pi)$ oraz $C_{\max}(\beta) > C_{\max}(\pi^*)$, wówczas prawdopodobieństwo jej przyjęcia za rozwiązanie startowe w następnej iteracji określa funkcja akceptacji. Jest ona zależna od temperatury (t). W konstrukcji algorytmów stosować będziemy dwie funkcje:

a) z pracy [12], postaci:

$$\Psi_t(\pi,\beta) = \exp[-(C_{\max}(\beta) - C_{\max}(\pi))/t],$$

b) zamieszczoną w pracy [2]:

$$\Psi_t(\pi,\beta) = \exp[(-LB(\beta) + C_{\max}(\pi))\frac{-2\ln(t)}{C_{\max}(\pi)}],$$

gdzie $LB(\beta)$ jest dolnym ograniczeniem wartości $C_{\max}(\beta)$.

4.3. Schemat schładzania

W literaturze opisano wiele schematów schładzania (zmiany temperatury). Poniżej zamieszczamy te z nich, które z powodzeniem są stosowane w algorytmach symulowanego wyżarzania, dla problemów szeregowania zadań. Podajemy także zalecane wartości parametrów. W praktyce najlepsze parametry są niestety różne dla różnych przykładów.

Aarts [1]:

$$t_{k+1} = t_k/(1 + \frac{t_k \log(1+\delta)}{3\sigma_k}).$$

Zwykle przyjmuje się δ=0.1, a σ_k jest odchyleniem standardowym wartości funkcji celu w już wykonanych k pętlach algorytmu.

Geometryczny [12]:

$$t_{k+1} = \alpha t_k.$$

Ogólnie, parametr α<1 (zazwyczaj przyjmuje się $0.8 \le \alpha < 1$).

Lundy [14]:

$$t_{k+1} = \frac{t_k}{1+\beta t_k}, \text{ lub } t_k = \frac{t_1}{1+n\beta t_1},$$

gdzie β jest stałą, przy czym przyjmuje się:

$$\beta = \frac{t_1 - t_N}{t_1 \cdot t_N (N-1)},$$

a N jest liczbą iteracji algorytmu. Osman i Potts w pracy [18] proponują, aby dla permutacyjnego problemu szeregowania przyjąć:

$$t_1 = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}/5mn, \quad t_N = 1 \quad \text{oraz} \quad N = 10000, 50000, 20000.$$

Logarytmiczny[8]:

$$t_n = \frac{c}{\log(n+n_0)},$$

przy czym stała c jest liczbą całkowitą nie większą niż 1000.

Stała temperatura:

W pracy Cohn, Fielding [4] przedstawiony jest algorytm symulowanego wyżarzania dla problemu komiwojażera ze stałą temperaturą (z przedziału (0,60]). Odpowiednio dobrana wartość powoduje, że dla wielu przykładów algorytm ten działa bardzo dobrze.

5. Algorytm równoległy

Jako model obliczeń równoległych wybrana została maszyna SIMD. Zastosowano dwie metody zrównoleglania algorytmu. Pierwsza to równoległe i niezależne przeszukiwanie. Polega ono na jednoczesnym wykonywaniu, przez poszczególne procesory, pewnej liczby niezależnych algorytmów symulowanego wyżarzania. Następnie wybiera się najlepsze rozwiązanie spośród wszystkich. Druga metoda polega na wysyłaniu pozostałym procesorom komunikatów (*broadcast*) o najlepszym rozwiązaniu, otrzymanym przez któryś z procesorów.

Równoległy algorytm symulowanego wyżarzania z wysyłaniem komunikatów (dla P procesorów):

```
parfor j=1,2,...,P
while i≤R do
  begin
        i←i+1;
        Wyznaczyć losowo permutację β∈N(π);
        if Cmax(β) < Cmax(π*) then
    begin
      π*←β;
      broadcast π* do innych procesorów
        end;
        if Cmax(β) < Cmax(π) then  π←β
        else
    if  Ψt(π,β)>random[0,1) then  π←β
end;{i}
      i←0;
  zmodyfikować parametr kontrolny t, zgodnie ze schematem chłodzenia
end {parfor}
```

Komunikacja pomiędzy procesorami (wysyłanie nowego π^*) następuje tylko wówczas, gdy procesor znalazł lepsze rozwiązanie od własnego, aktualnego π^*. Skraca to całkowity czas komunikacji i ma bardzo niewielki wpływ na wartość rozwiązania.

6. Wyniki obliczeniowe

Przedstawione w poprzednich rozdziałach algorytmy zostały zaprogramowane w języku Ada95 na 4-procesorowym komputerze Sun Enterprise 4x400Mhz i były testowane na wielu przykładach zamieszczonych w pracy Taillarda [22]. Są to przykłady o różnych rozmiarach od 100 do 10 000 operacji, po 10 przykładów każdego rozmiaru. Zamieszczone są one pod adresem OR-Library: http://mscmga.ms.ic.ac.uk/info.html). Znajdują się tam także wartości najlepszych rozwiązań, a także ich dolne i górne oszacowania.

Przyjęto następujące wartości parametrów liczbowych algorytmów:

$R = N(\pi^*)$ – długość pętli, tj. liczba iteracji, po której następuje zmiana temperatury,

Max_iter – maksymalna liczba iteracji (***Warunek zatrzymania***). Jest ona równa $50*n$ dla równoległej implementacji na 4 procesorach oraz $200*n$ dla jednoprocesorowej implementacji.

Liczba iteracji algorytmów jest więc wielokrotnością 200 dla jednoprocesorowej implementacji i 50 na każdy procesor dla implementacji czteroprocesorowej.

Rozwiązania początkowe (startowe) były wyznaczane algorytmem NEH ([15]).

W pierwszej kolejności testowano wszystkie kombinacje par: dwóch funkcji akceptacji oraz pięciu schematów schładzania, przedstawione w rozdziale 4. W sumie więc porównano pomiędzy sobą 10 różnych wersji algorytmu. Okazało się, że zdecydowanie najlepsze rozwiązania wyznacza algorytm z funkcją akceptacji (4.3 punkt a)) oraz schematem schładzania Lundy [14] z parametrami zamieszczonymi w pracy Osmana i Pottsa [18]. Następnie rozwiązania wyznaczone przez tę właśnie wersję algorytmu porównano z najlepszymi znanymi w literaturze. Otrzymane wyniki (w procentach) zamieszczone są w tabeli 1.

Na podstawie wyników zamieszczonych w tabeli 1 można stwierdzić, że algorytm równoległy z komunikacją pomiędzy procesorami jest najlepszy i daje wyniki tylko niewiele gorsze od najlepszych. Niewiele także różnią się one od wyznaczanych przez najlepsze algorytmy przybliżone (bazujące na metodzie tabu search), np. [7]. Szczególnie jest to

widoczne dla przykładów o dużych rozmiarach. Należy tu także podkreślić łatwość implementacji algorytmów opartych na metodzie symulowanego wyżarzania, ich niewielkie zapotrzebowanie na pamięć oraz krótki czas działania.

Tabela 1

Średni błąd (RDD) względem najlepszego rozwiązania (Taillard [21])

$n \times m$	1 procesor	4 niezależne procesory	4 procesory z komunikacją
20×5	0.84	0.62	0.59
20×10	2.29	1.82	1.66
20×20	1.86	1.79	1.62
50×5	0.13	0.08	0.11
50×10	1.87	1.30	0.91
50×20	2.77	2.18	2.19
100×5	0.00	0.00	0.00
100×10	0.02	0.01	0.01
100×20	0.88	0.01	0.06
200×10	0.02	0.03	0.02
200×20	0.01	0.01	0.01
500×20	0.01	0.01	0.00
średnio	**0.89**	**0.66**	**0.60**

7. Podsumowanie

W pracy przedstawiliśmy algorytmy równoległe oparte na metodzie symulowanego wyżarzania dla rozwiązywania klasycznego problemu szeregowania – permutacyjnego problemu przepływowego. Wykorzystaliśmy idee bloków z drogi krytycznej oraz zamiast dokładnego liczenia wartości funkcji celu - jej dolne szacowanie. Może to nieznacznie pogarszać jakość rozwiązania, ale znacznie przyśpiesza obliczenia. Sprawdziliśmy także wpływ różnych schematów schładzania i funkcji akceptacji na efektywność działania algorytmów. Dla przykładów o dużych rozmiarach metoda symulowanego wyżarzania (zawierająca w sobie elementy probabilistyki) może być lepsza od innych metod lokalnej optymalizacji.

LITERATURA

1. Aarts E.H.L., Laarhoven P.J.M.; Statistical cooling: A general approach to combinatorial optimization problems, Philips J. Res., 40 (1985), s. 193-226.
2. Bożejko W., Wodecki M.: Solving Flow Shop Problem by Parallel Simulated Annealing, LNCS, Springer-Verlag, 2328, (2002), 236-247.
3. Černý V.: Thermodynamical approach to travelling salesman problem: An efficient simulation algorithm, J. Optim. Theory Appl. 45, (1985), s. 41-51.
4. Cohn H., Fielding M.: Simulated Annealing: Searching for an optimal temperature schedule, SIAM J.Optim., 9 (1999), s. 779-802.
5. Garey M.R., Johnson D.S., Seti R.: The complexity of flow shop and job shop scheduling, Mathematics of Operations Research, 1, (1976), s. 117-129.
6. Grabowski J.: A new algorithm of solving the flow-shop problem, Operations Research in Progress, D. Reidel Publishing Company, (1982), s. 57-75.
7. Grabowski J., Pempera J.: New block properties for the permutation flow-shop problem with application in TS, Journal of Operational Research Society 52, (2001), s. 210-220.
8. Hajek B.: Cooling schedules for optimal annealing, J.Math.Oper.Res.,13 (1988), 311-329.
9. Ignall E., Schrage L.E.: Application of the branch-and-bound technique to some flow-shop scheduling problems, Operations Research, (1965), 13/3, s. 400-412.
10. Ishibuchi H., Misaki S., Tanaka H.: Modified Simulated Annealing Algorithms for the Flow Shop Sequencing Problem, European Journal of Operational Research 81, (1995), s. 388-398.
11. Johnson S.M.: Optimal two and three-stage production schedules with setup times included, Naval Research Logistic Quertely, 1 (1954), s. 61-68.
12. Kirkpatrick S., Gellat C.D., Vecchi M.P.: Optimization by simulated annealing, Science 220, (1983), s. 671-680.
13. Lageweg B.J., Lenstra J.K., Rinnooy Kan A.H.G.: A General Bouding Scheme for the Permutation Flow-Schop Problem, Opns. Res. 26, (1978), s. 53-67.
14. Lundy M., Mees A.: Convergence of an annealing algorithm, Math. Programming, 34, (1986), s. 111-124.
15. Navaz M., Enscore E.E. Jr, Ham I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, OMEGA 11/1 (1983) s. 91-95.
16. Nowicki E., Smutnicki C.: A fast tabu search algorithm for the permutation flow-shop problem, European Journal of Operational Research 91, (1996), s. 160-175.
17. Ogbu F., Smith D.: The Application of the Simulated Annealing Algorithm to the Solution of the n/m/Cmax Flowshop Problem, Computers & Operations Research, 17(3), (1990), s. 243-253.
18. Osman I., Potts C.: Simulated Annealing for Permutation Flow-Shop Scheduling, OMEGA 17(6), (1989), s. 551-557.
19. Reeves C.: Improving the Efficiency of Tabu Search for Machine Sequencing Problems, Journal of Operational Research Society 44(4), (1993), s. 375-382.

20. Reeves C.: A Genetic Algorithm for Flowshop Sequencing, Computers & Operations Research 22(1), (1995), s. 5-13.

21. Taillard E.: Some efficient heuristic methods for the flow shop sequencing problem, European Journal of Operational Research 47(1), (1990), s. 65-74.

22. Taillard E.: Benchmarks for basic scheduling problems, European Journal of Operational Research 64, (1993), s. 278-285.

Recenzent: Prof. dr hab. inż. Tadeusz Sawik

Abstract

In this paper we consider the permutation flow shop scheduling problem described as follows. There are: a set of n jobs $J=\{1,2,\ldots,n\}$, a set of m machines $M=\{1,2,\ldots,m\}$. Job $j\in J$, consists of a sequence of m operations O_{j1}, $O_{j2},\ldots$, O_{jm}. Operation O_{jk} corresponds to the processing of job j on machine k during an uninterrupted processing time p_{jk}. We want to find a schedule such that the maximum completion times is minimal. The problem is indicated by $F|m|C_{\max}$ and is strongly NP-hard. We propose neighbourhood using so called blocks of jobs on a critical path, specific accepting function and two simulated annealing algorithms (sequential and parallel). We also present computer simulations on Taillard [21] problems.