Wojciech BOŻEJKO, Mariusz UCHROŃSKI
Politechnika Wrocławska

Paweł RAJBA, Mieczysław WODECKI
Uniwersytet Wrocławski

MULTIRUCHY W GENEROWANIU OTOCZEŃ DLA PROBLEMU SZEREGOWANIA Z RÓWNOLEGŁYMI MASZYNAMI

Streszczenie. W pracy jest rozpatrywany ogólny problem gniazdowy z równoległymi maszynami (ang. flexible job shop problem). Należy on do klasy problemów silnie NP-trudnych. Przedstawiamy nowy typ otoczeń generowanych przez multiruchy (złożenia ruchów). Otoczenia te są z powodzeniem stosowane w algorytmach opartych na metodzie lokalnych poszukiwań.

MULTIMOVES IN NEIGHBORHOOD DETERMINATION FOR SCHEDULING PROBLEM WITH PARALLEL MACHINES

Summary. Flexible job shop problem is considered here. It belongs to the class of strongly NP-hard problems. We present the new neighborhoods type determined by multimoves (moves concatenation). These neighborhood can be successfully applied in local search methods.

1. Wprowadzenie

Ogólny problem gniazdowy z równoległymi maszynami polega na przydzieleniu zadań do maszyn tak, aby zminimalizować czas ich wykonania. W problemie tym dany jest zbiór zadań oraz zbiór maszyn różnych typów. Maszyny tego samego typu, tj. o tych samych własnościach funkcjonalnych różniące się jednak wydajnością tworzą gniazda. Każde zadanie należy wykonać na jednej z maszyn ustalonego typu. Dane są także czasy wykonywania zadań. Należy przydzielić zadania do maszyn oraz ustalić ich kolejność tak, aby zminimalizować czas zakończenia wykonania zadań. Problem ten jest silnie NP-trudny.

Algorytm optymalny, w którym rozwiązania były reprezentowane przez grafy dysjunktywne, przedstawili Adrabiński i Wodecki [1] oraz Pinedo [13]. Efektywnie tymi algorytmami można rozwiązywać małe przykłady (maksymalnie do 20 zadań i 10 maszyn). Opublikowano także wiele algorytmów przybliżonych, wykorzystując różne metaheurystyki: przeszukiwanie z tabu (Hurik [10] oraz Dauèrze-Pèrés i Pauli [6]), algorytmu genetycznego (Ho i Tat [9], Pezella i in. [12]) oraz algorytm hybrydowy (Gao i in. [7]). Przedstawiony przez Bożejkę, Uchrońskiego i Wodeckiego [4] algorytm równoległy składa się z dwóch modułów:

1) przydziału operacji do maszyn,
2) rozwiązywania klasycznego problemu gniazdowego.

W każdym z modułów stosowane są niezależne procedury oparte na metodzie poszukiwania z tabu. Jednym z zasadniczych elementów algorytmów opartych na metodzie lokalnych poszukiwań jest otoczenie. Sposób jego generowania i przeszukiwania ma decydujący wpływ na efektywność algorytmu. Tradycyjne otoczenia są generowane przez pojedyncze ruchy. W pracy przedstawiamy nowe otoczenie generowane przez złożenia różnych typów ruchów. Można je traktować jako „złożenie" otoczeń stosowanych w algorytmie zamieszczonym w pracy [4].

2. Sformułowanie problemu

Ogólny problem gniazdowy z równoległymi maszynami (oznaczany przez FJSP) można sformułować następująco:

PROBLEM: Dany jest zbiór zadań $J = \{1,2,...,n\}$, które należy wykonać na maszynach ze zbioru $M = \{1,2,...,m\}$. Istnieje rozbicie zbioru maszyn na typy, tj. podzbiory maszyn o tych samych własnościach funkcjonalnych. Zadanie jest ciągiem pewnych operacji. Każdą operację należy wykonać na odpowiedniego typu maszynie w ustalonym czasie. Problem polega na przydzieleniu zadań do maszyn odpowiedniego typu oraz na wyznaczeniu kolejności wykonywania operacji na maszynach, aby zminimalizować czas wykonania wszystkich zadań. Muszą być przy tym spełnione ograniczenia:

(i) każde zadanie może być wykonywane jednocześnie tylko na jednej maszynie odpowiedniego typu,
(ii) dowolna maszyna może wykonywać jednocześnie co najwyżej jedno zadanie,
(iii) wykonywanie zadania nie może być przerwane,
(iv) zachowany musi być porządek technologiczny wykonywania operacji.

Niech $O = \{1,2,...,o\}$ będzie zbiorem wszystkich operacji. Zbiór ten można rozbić na ciągi odpowiadające zadaniom, przy czym zadanie $j \in J$ jest ciągiem o_j operacji, które będą kolejno wykonywane na odpowiednich maszynach (tj. w ciągu technologicznym). Operacje te są indeksowane liczbami $(l_{j-1}+1,...,l_{j-1}+o_j)$, gdzie $l_j = \sum_{i=1}^{j} o_i$ jest liczbą operacji pierwszych j zadań, przy czym $l_0 = 0$, a $o = \sum_{i=1}^{n} o_i$.

Zbiór maszyn $M = \{1,2,...,m\}$ można rozbić na q podzbiorów maszyn tego samego typu (*gniazd*), przy czym i-ty $(i = 1,2,...,q)$ typ M^i zawiera m_i maszyn, które są indeksowane liczbami $(t_{i-1}+1,...,t_{i-1}+m_i)$, gdzie $t_i = \sum_{j=1}^{i} m_j$ jest liczbą maszyn w pierwszych i typach, $i = 1,2,...,q$, przy czym $t_0 = 0$, a $m = \sum_{j=1}^{q} m_j$.

Operację $v \in O$ należy wykonać w gnieździe $\mu(v)$, tj. na jednej z maszyn zbioru $M^{\mu(v)}$ w czasie $p_{v,r}$, gdzie $r \in M^{\mu(v)}$.

Niech

$$O^k = \{v \in O : \mu(v) = k\}$$

będzie zbiorem operacji wykonywanych w k-tym gnieździe. Ciąg zbiorów operacji

$$Q = [Q^1, Q^2, ..., Q^m]$$

takich, że dla każdego $k = 1,2,...,q$

$$O^k = \bigcup_{i=t_{k-1}+1}^{t_{k-1}+m_k} Q^i \text{ oraz } Q^i \cap Q^j = \emptyset, i \neq j, i, j = 1,2,...,m,$$

nazywamy *przydziałem operacji ze zbioru O do maszyn ze zbioru M* (w skrócie **przydziałem operacji do maszyn**). Zbiór Q^i zawiera więc operacje przydzielone do maszyny $i \in M$.

Ciąg $[Q^{t_{k-1}+1}, Q^{t_{k-1}+2}, ..., Q^{t_{k-1}+m_k}]$ jest **przydziałem** maszynom operacji w i-tym gnieździe (w skrócie *przydziałem* w i-tym gnieździe). W szczególnym przypadku maszyna może nie wykonywać żadnej operacji i wówczas w przydziale operacji w gnieździe zbiór operacji do wykonywania przez tę maszynę jest pusty.

Jeżeli dokonano przydziału operacji do maszyn, wówczas wyznaczenie optymalnego terminu wykonywania operacji (w tym i kolejności wykonywania operacji na maszynach) sprowadza się do rozwiązania klasycznego problemu szeregowania, tzw. problemu gniazdowego (Nowicki i Smutnicki [9], Grabowski i Wodecki [6]).

Niech $Q = [Q^1, Q^2, ..., Q^m]$ będzie pewnym przydziałem operacji do maszyn. Przez

$$\pi(Q) = (\pi_1, \pi_2, ..., \pi_m)$$

oznaczamy konkatenację (złożenie) m ciągów, gdzie π_i jest permutacją elementów zbioru Q^i, tj. kolejnością wykonywania operacji na maszynie $i \in M$.

Dowolne rozwiązanie dopuszczalne problemu FJSP jest parą $(Q, \pi(Q))$, gdzie Q jest przydziałem operacji do maszyn, a $\pi(Q)$ konkatenacją (ciągiem) permutacji wyznaczających kolejność wykonywania operacji, spełniającą ograniczenia (i-iv), przydzielonych każdej z maszyn. Przez Φ^* oznaczamy zbiór rozwiązań dopuszczalnych dla problemu FJSP.

3. Reprezentacja grafowa rozwiązania

Dowolne rozwiązanie dopuszczalne $\Theta = (Q, \pi(Q)) \in \Phi^*$ (gdzie Q jest przydziałem operacji do maszyn, a $\pi(Q)$ jest kolejnością wykonywania operacji na każdej maszynie) problemu FJSP można przedstawić w postaci grafu skierowanego z obciążonymi wierzchołkami (sieci) $G(\Theta) = (V, R \cup E(\Theta))$, gdzie V jest zbiorem wierzchołków, a $R \cup E(\Theta)$ zbiorem łuków, przy czym:

1. $V = O \cup \{s, c\}$, gdzie s i c są dodatkowymi operacjami reprezentującymi odpowiednio „start" i „zakończenie". Wierzchołek $v \in V \setminus \{s, c\}$ ma dwie cechy:
   - $\lambda(v)$ – numer maszyny, na której należy wykonać operację $v \in O$,
   - $p_{v,\lambda(v)}$ – wagę wierzchołka równą czasowi wykonywania operacji $v \in O$ na maszynie $\lambda(v)$.

   Wagi dodanych wierzchołków $p_s = p_c = 0$.

2. $R = \bigcup_{j=1}^{n} \left[\bigcup_{i=1}^{o_j - 1} \left\{ \left(l_{j-1} + i, l_{j-1} + i + 1 \right) \right\} \cup \left\{ \left(s, l_{j-1} + 1 \right) \right\} \cup \left\{ \left(l_{j-1} + o_j, c \right) \right\} \right]$.

   Zbiór R zawiera łuki łączące kolejne operacje tego samego zadania oraz łuki z wierzchołka s do pierwszej operacji każdego zadania i łuki od ostatniej operacji każdego zadania do wierzchołka c.

3. $E(\Theta) = \bigcup_{k=1}^{m} \bigcup_{i=1}^{|Q^k|-1} \left\{ \left(\pi_k(i), \pi_k(i+1) \right) \right\}$.

   Łatwo zauważyć, że łuki ze zbioru $E(\Theta)$ łączą operacje wykonywane na tej samej maszynie (π_k jest permutacją operacji wykonywanych na maszynie M_k, tj. operacji ze zbioru Q^k).

Łuki ze zbioru R wyznaczają kolejność wykonywania operacji w zadaniach (porządek technologiczny), a łuki ze zbioru $E(\Theta)$ kolejność wykonywania operacji na każdej z maszyn.

Twierdzenie 1 [4]. Para $\Theta = (Q, \pi(Q))$ jest rozwiązaniem dopuszczalnym dla problemu FJSP wtedy i tylko wtedy, gdy graf $G(\Theta)$ nie zawiera cykli.

Ciąg wierzchołków $(v_1, v_2, ..., v_k)$ grafu $G(\Theta)$ taki, że $(v_i, v_{i+1}) \in R \cup E(\Theta)$ dla $i = 1, 2, ..., k-1$, nazywamy *drogą* z wierzchołka v_1 do v_k. Przez $C(v,u)$ oznaczmy najdłuższą drogę (*drogę krytyczną*) w grafie $G(\Theta)$ z wierzchołka v do u ($v, u \in V$).

Łatwo zauważyć, że czas wykonywania wszystkich operacji $C_{\max}(\Theta)$ zgodnie z przydziałem operacji Q i kolejnością (uszeregowaniem) $\pi(Q)$ jest równy długości drogi krytycznej $C(s,c)$ w grafie $G(\Theta)$. Rozwiązanie problemu gniazdowego z równoległymi maszynami sprowadza się więc do wyznaczenia takiego rozwiązania dopuszczalnego $\Theta = (Q, \pi(Q))$, dla którego odpowiadający mu graf $G(\Theta)$ ma najkrótszą drogę krytyczną, tj. minimalizuje długość tej drogi $L(s,c)$.

4. Metoda rozwiązania

Ze względu na liczbę operacji, możliwych przydziałów operacji do maszyn jest wykładnicza ilość. Każdy dopuszczalny przydział generuje pewien silnie NP-trudny problem (job shop), którego rozwiązanie sprowadza się do wyznaczenia optymalnych kolejności wykonywania operacji na maszynach. Wobec tego optymalne rozwiązanie problemu FJSP wymaga rozwiązania wykładniczej liczby silnie NP-trudnych problemów. Dlatego też w praktyce są stosowane algorytmy przybliżone, w tym oparte na metodzie lokalnych poszukiwań (np. symulowane wyżarzanie, przeszukiwanie z tabu). Zasadniczym ich elementem są otoczenia, podzbiory zbioru rozwiązań generowane z bieżącego rozwiązania, przez przekształcenia zwane ruchami. W problemie FJSP z ustalonego rozwiązanie można wygenerować inne rozwiązanie między innymi przez wykonanie ruchu polegającego na:

- przeniesieniu (transfer) operacji z pewnej maszyny na inną (w tym samym gnieździe), lub
- zmianie kolejności wykonywania operacji na dowolnej maszynie.

W dalszej części opiszemy dokładnie oba te ruchy.

Niech $\Theta = (Q, \pi(Q))$ będzie rozwiązaniem dopuszczalnym problemu FJSP, gdzie $Q = [Q^1, Q^2, ..., Q^m]$ jest przydziałem operacji do maszyn, a

$$\pi(Q) = (\pi_1, \pi_2, ..., \pi_m)$$

konkatenacją m permutacji. Permutacja π_i wyznacza kolejność operacji ze zbioru Q^i, które należy wykonać na maszynie $M_i \in M$. Przez $C_{\max}(\Theta)$ oznaczamy wartość funkcji celu dla rozwiązania Θ, tj. termin zakończenia wykonywania wszystkich operacji.

4.1. Ruchy typu transfer

Przez $t_j^i(k,l)$ oznaczamy ruch typu *transfer* (w skrócie *t-ruch*) polegający na przeniesieniu operacji znajdującej się na pozycji k w permutacji π_i (tj. operacji $\pi_i(k)$) na pozycję l w permutacji π_j (przesuwając wcześniej operacje znajdujące się na pozycjach $l, l+1$, itd. o jedną pozycję w prawo). Wykonanie ruchu $t_j^i(k,l)$ generuje z $\Theta = (Q, \pi)$ nowe rozwiązanie $\Theta' = (Q', \pi')$ takie, że

$$\pi'_v = \pi_v, \quad v \neq i, j, \quad v = 1, 2, ..., m,$$

$$\pi'_i = (\pi_i(1), \pi_i(2), ..., \pi_i(k-1), \pi_i(k+1), ..., \pi_i(|Q^i|-1),$$

$$\pi'_j = (\pi_j(1), \pi_j(2), ..., \pi_j(l-1), \underline{\pi_i(k)}, \pi_j(l), ..., \pi_i(|Q^j|+1).$$

Wykonanie tego ruchu powoduje przeniesienie operacji $\pi_i(k)$ ze zbioru Q^i (tj. z maszyny M_i) do zbioru Q^j (tj. na maszynę M_j). Wobec tego

$$Q'^v = Q^v, \quad v \neq i, j, \quad v = 1, 2, ..., m$$

$$Q'^i = Q^i \setminus \{\pi_i(k)\}, \quad Q'^j = Q^j \cup \{\pi_i(k)\}.$$

Ruch $t_j^i(k,l)$ taki, że $i = j$ oraz $k = l$ nazywamy *t-ruchem neutralnym*.

Złożoność obliczeniowa *t*-ruchu wynosi $O(n)$. Wykonanie *t-ruchu* powoduje przeniesienie operacji z maszyny na inną, tj. nowy przydział operacji do maszyn w pewnym gnieździe. Wobec tego, z dowolnego rozwiązania (przydziału operacji do maszyn), wykonując *t-ruchy* można otrzymać dowolny inny przydział operacji do maszyn, tj. rozbicie zbiorów operacji na maszyny w poszczególnych gniazdach.

Jeżeli τ jest *t-ruchem*, to przez $\tau(\Theta)$ oznaczamy rozwiązanie wygenerowane z Θ przez wykonanie ruchu τ. Może się jednak tak zdarzyć, że rozwiązanie $\tau(\Theta)$ nie jest dopuszczalne.

Niech Θ będzie pewnym rozwiązaniem dopuszczalnym. Zbiór

$$T_j^i(\Theta) = \{t_j^i(k,l) : k \in Q^i \text{ and } l \in Q^j$$

zawiera wszystkie *t-ruchy* przenoszące operacje z maszyny M_i na maszynę M_j, a

$$T(\Theta) = \bigcup_{i,j \in M} T_j^i(\Theta)$$

wszystkie *t-ruchów* dla rozwiązania Θ. Moc tego zbioru jest z góry ograniczona przez $O(qm^2o^2)$.

4.2 Ruchy typu wstaw (insert)

Dla ustalenia uwagi załóżmy, że permutacja $\pi = (\pi(1), \pi(2), ..., \pi(t))$ wyznacza kolejność wykonywania operacji na pewnej maszynie.

Ruch typu wstaw i_l^k (ang. *insert move*), przestawia element $\pi(k)$ (z pozycji k w permutacji π) na pozycję l, generując permutację $i_l^k(\pi) = \pi_l^k$. Jeżeli $l \geq k$, to wówczas:

$$\pi_l^k(i) = \begin{cases} \pi(i), & \text{jeśli } i < k \vee i > l, \\ \pi(i+1), & \text{jeśli } k \leq i < l, \\ \pi(k), & \text{jeśli } i = l. \end{cases}$$

Podobnie jest w przypadku gdy $k > l$.

Ruch typu wstaw będzie w skrócie nazywany *i-ruchem*. Jego złożoność obliczeniowa jest $O(1)$. Wszystkich takich ruchów jest $t(t-1)$, gdzie t jest liczba operacji.

Dla ustalonego rozwiązania dopuszczalnego Θ, niech $I_k(\Theta)$ będzie zbiorem wszystkich *i-ruchów* dla maszyny $M_k \in M$, a

$$I(\Theta) = \bigcup_{k=1}^{m} I_k(\Theta)$$

zbiorem wszystkich *i-ruchów* na wszystkich maszynach. Łatwo zauważyć, że nie wszystkie ruchy z tego zbioru generują rozwiązania dopuszczalne. Niedopuszczalność, tj. cykl w grafie rozwiązania może powstać, gdy dwie różne operacje pewnego zadania należy „wykonać" w tym samym gnieździe.

4.3. Otoczenie generowane przez multiruchy

Ruch typu transfer powoduje „przerzucenie" (jak silne uderzenie w golfie) operacji na inną maszynę. Z kolei ruchy typu wstaw jedynie "delikatnie" modyfikują kolejność wykonywania operacji na maszynach. Inspiracją do składania tych ruchów były wyniki badań zamieszczone w pracy Bożejko i Wodecki [3].

Niech Θ będzie pewnym rozwiązaniem dopuszczalnym, a $T(\Theta)$ zbiorem wszystkich *t-ruchów*. Rozpatrujemy ruch $t_j^i(k,l) \in T(\Theta)$. Przenosi on operację z pozycji k-tej na maszynie M_i na pozycję l-tą na maszynie M_j. Ruch ten generuje rozwiązanie $\Theta' = t_j^i(k,l)(\Theta)$. Zbiór $I(\Theta')$ zawiera wszystkie *i-ruchy* związane z rozwiązaniem Θ', a $I_j(\Theta')$ *i-ruchy* określone na operacjach wykonywanych na maszynie M_j.

Niech $i_t^s \in I_j(\Theta')$ będzie pewnym i-*ruchem*. Wykonanie tego ruchu generuje z Θ' nowe rozwiązanie $\Theta'' = i_t^s(\Theta')$. Przekształcenie generujące z Θ rozwiązanie Θ'' nazywamy *it-multiruchem*. Jest ono złożeniem *t-ruchu* $t_j^i(k,l)$ i *i-ruchu* i_t^s. W skrócie będziemy ten multiruch oznaczali przez $i_t^s \circ t_j^i(k,l)$. Wobec tego $\Theta'' = i_t^s \circ t_j^i(k,l)(\Theta)$.

Przez $I \circ T(\Theta)$ oznaczamy zbiór wszystkich *it-multiruchów* określonych dla rozwiązania Θ. *Otoczeniem golfowym* rozwiązania Θ jest zbiór

$$N(\Theta) = \{\lambda(\Theta): \lambda \in I \circ T(\Theta)\}. \quad (1)$$

Moc tego otoczenia wynosi $O(qm^2o^4)$. Jeżeli ograniczymy się jedynie do *t-ruchów* neutralnych, wówczas (1) zawiera ruchy stosowane w najlepszych algorytmach przybliżonych rozwiązywania problemu gniazdowego zamieszczonych w pracach Nowickiego i Smutnickiego [11] oraz Grabowskiego i Wodeckiego [8].

Eksperymenty obliczeniowe otoczenia golfowego wykonano korzystając z algorytmu TSOSGW ([8]), rozwiązywania problemu gniazdowego, opartego na metodzie przeszukiwania z tabu. Zasadnicza modyfikacja polegała na uwzględnieniu otoczenia (1). Obliczenia wykonano na komputerze osobistym z procesorem Pentium 1,2 GHz. Liczba iteracji algorytmu (warunek zatrzymania) wynosi $100n$. Otrzymane wyniki, dla części przykładów z pracy [2] zamieszczono w Tabeli 1. Średni czas obliczeń jednego przykładu jest niewielki, bowiem nie przekracza 7 sek. Ponieważ obliczenia algorytmu M^2h [4] wykonano na 128 procesorowej karcie graficznej Tesla C870, więc istnieje duża trudność z rzetelnym porównaniem czasów obliczeń.

Tabela 1

Wartości rozwiązań dla przykładów testowych Barnesa i Chambersa [2]

Problem	nxm	o	(LB,UB)	M^2h [4]	TSOSGW[8]
mt10c1	10x11	100	(655,927)	927	927
mt10xx	10x12	100	(655,929)	918	918
mt10xxx	10x13	100	(655,936)	918	918
setb4c9	15x11	150	(857,924)	914	913
setb4cc	15x12	150	(857,909)	907	908
setb4xxx	15x13	150	(846,925)	925	925
seti5c12	15x16	225	(1027,1185)	1174	1174
seti5cc	15x17	225	(955,1136)	1136	1136
seti5xxx	15x18	225	(955,1213)	1197	1197

W pracy Bożejki, Uchrońskiego i Wodeckiego [5] udowodniono między innymi tzw. kryteria eliminacyjne umożliwiające eliminowanie *i* oraz *t-ruchów* generujących rozwiązania niedopuszczalne. Ponieważ liczba *t-ruchów* może być bardzo duża (jest rzędu $O(qm^2o^2)$) więc niektóre z nich pomijamy, a także szacujemy wartości funkcji celu rozwiązań generowanych przez *t-ruchy*. Powoduje to znaczne (o około 50%) przyśpieszenie działania algorytmu przy niewielkim pogorszeniu wyników.

***

Praca częściowo finansowana z projektu badawczego MNiSW Nr N N514 232237.

LITERATURA

1. Adrabiński A, Wodecki M.: An algorithm for solving the machine sequencing problem with parallel machines, Zast. Mat. XVI 3,1979, p. 513-541.

2. Barnes J.W., Chambers J.B.: Flexible job shop scheduling by tabu search. Graduate program in operations research and industrial engeering, The University of Texas at Austin, Technical Report Series: ORP96-09, 1966.
3. Bożejko W., Wodecki M.: On the theoretical properties of swap multimoves, Operations Research Letters 35/2, 2007, p. 227-231.
4. Bożejko W., Uchroński M., Wodecki M.: Parallel Meta^{2}heuristics for the Flexible Job Shop Problem, LNAI, 6114, 2010, p. 395-402.
5. Bożejko W., Uchroński M., Wodecki M.: Parallel hybrid metaheuristics for the flexible job shop problem, Com. & Ind. Engiee. (doi:10.1016/j.cie.2010.05.004).
6. Dauzère-Pérès S., Pauli J.: An integrated approach for modeling and solving the general multiprocessor job shop scheduling problem using tabu search, Annals of Operations Research 70(3), 1997, p. 281–306.
7. Gao J., Sun L., Gen M.: A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems, Computers & Operations Research 35, 2008, p. 2892–2907.
8. Grabowski J., Wodecki M.: A very fast tabu search algorithm for the job shop problem. In: Rego C., Alideaee B. (eds.), Adaptive memory and evolution; tabu search and scatter search, Dordrecht, Kluwer, 2005, p. 117–144.
9. Ho N.B., Tay J.C.: An efficient cultural algorithm for solving the Flexible Job-Shop Problem, IEEE, Inter. Conf. on Rob. and Aut., 2004, p. 1759–1766.
10. Hurink E., Jurisch B., Thole M.: Tabu search for the job shop scheduling problem with multi-purpose machine, Oper. Res. Spektrum 15, 1994, p. 205–215.
11. Nowicki E., Smutnicki C., A fast tabu search algorithm for the job shop problem, Management Scence 421, 1996, p. 797-813.
12. Pezzella F., Morganti G., Ciaschetti G.: A genetic algorithm for the Flexible Job-schop Scheduling Problem, Comp. & Oper. Res. 35, 2008, p. 3202–3212.
13. Pinedo M.: Scheduling: theory, algorithms and systems, Englewood cliffs, NJ: Prentice-Hall, 2002.