ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: AUTOMATYKA z.114 Nr kol. 1250

Eugeniusz NOWICKI
Politechnika Wrocławska

ZASTOSOWANIE TECHNIKI TABU SEARCH DO PROBLEMÓW SZEREGOWANIA W OTWARTYCH SYSTEMACH OBSŁUGI

Streszczenie: W pracy przedstawia się szybki aproksymacyjny algorytm oparty na technice tabu search, dla problemu szeregowania zadań w otwartym systemie obsługi (open shop problem) z kryterium będącym czasem wykonania wszystkich zadań. Wyniki badań numerycznych pokazują, że proponowany algorytm nie tylko znajduje uszeregowania o mniejszej wartości funkcji celu niż inne najlepsze aproksymacyjne algorytmy, ale także pracuje w krótszym czasie.

APPLICATION OF TABU SEARCH TECHNIQUE FOR THE OPEN SHOP PROBLEM

Summary: A fast approximation algorithm for the problem of finding a minimum makespan in an open-shop is presented. The algorithm is based on a tabu search technique. Computational experiments show that the algorithm not only finds shorter makespans than the best approximation approaches but also runs in shorter time.

ANWENDUNG DER TABU SEARCH METHODE FÜR DAS OPEN SHOP PROBLEM

Zusammenfassung: Mit dem Ziel der Minimierung der Gesamtbearbeitungszeit wird heuristischer Algorithmus basiert auf der Tabu Search Methode für das Open Shop Problem betrachtet. Der Algorithmus war an den aus der Literatur bekannten Benchmark Beispielen getestet. Die Resultate der Testrechnungen demonstrieren eindrucksvoll die Qualität unseres Algorithmus.

1. Wstęp

Ogólnie algorytmy aproksymacyjne dla problemów szeregowania zadań na maszynach można podzielić na dwie klasy: *algorytmy konstrukcyjne* i *algorytmy popraw*. Do pierwszej klasy należą algorytmy, które na bazie pewnych reguł budują uszeregowanie i po jego otrzymaniu kończą swoje działanie. Algorytmy popraw startują już z pewnego uszeregowania (z reguły otrzymanego przez algorytmy konstrukcyjne) i następnie w kolejnych

etapach próbują je poprawić w celu zmniejszenia wartości funkcji celu.

Główne techniki stosowane dotychczas przy konstrukcji algorytmów popraw polegały na lokalnym przeszukiwaniu otoczeń i akceptowaniu do dalszych poszukiwań tylko uszeregowań o mniejszej - niż aktualnie najmniejsza - wartości funkcji celu. Algorytmy te różniły się między sobą takimi elementami jak: definicja otoczenia, strategia jego przeglądania, warunki stopu itp. Ich wspólną niekorzystną cechą było zatrzymywanie się w obszarach minimów lokalnych. Jedyna możliwość przezwyciężenia tej wady polegała na wielokrotnym uruchamianiu danego algorytmu - startując z różnych rozwiązań początkowych. Postępowanie takie niestety też nie dawało satysfakcjonujących wyników. Dlatego, aktualne kierunki badań dotyczą tylko takich technik, które nie posiadają wspomnianej wady.

W badaniach tych, bazujących przede wszystkim na metodach sztucznej inteligencji, można wyróżnić (między innymi) następujące techniki : (i) technika genetyczna, (ii) technika symulowanego wyżarzania (simulated annealing) oraz (iii) technika tabu search. Sygnalizowane w literaturze implementacje dwóch pierwszych technik dla wybranych problemów szeregowania zadań na maszynach pokazały, że znalezienie satysfakcjonujących rozwiązań następuje kosztem bardzo dużych nakładów obliczeniowych. Zaproponowana przez Glowera [2], w połowie lat osiemdziesiątych, technika tabu search zachowując zalety podejść (i)-(ii) pozwala znacznie zredukować nakłady obliczeniowe do wielkości zadowalających praktyków. Aktualnie jest ona intensywnie implementowana dla wielu problemów szeregowania. Przykładowo, ostatnio przedstawiono bardzo szybkie algorytmy dla problemów z maszynami równoległymi [5], problemów przepływowych [9] i gniazdowych [10].

Niniejsza praca jest jedną z pierwszych prób (oprócz pracy [11]), implementacji techniki tabu search dla problemu szeregowania n zadań na m maszynach w otwartym systemie obsługi (open shop problem) z kryterium będącym czasem wykonania wszystkich zadań. Bazując na tzw. blokach maszynowych i blokach zadaniowych, wykazuje się pewne własności eliminujące z poszukiwań niektóre uszeregowania. Następnie, wykorzystując te własności, podaje się oryginalną definicję otoczenia danego rozwiązania. Otoczenie to jest podstawą do konstrukcji algorytmu typu tabu search. Otrzymany algorytm został wstępnie przetestowany na szeregu problemach testowych wziętych z literatury. Wyniki badań numerycznych wskazują, że algorytm charakteryzuje się krótkim czasem obliczeń, produkując jednocześnie rozwiązania bliskie lub równe optymalnym.

2. Sformułowanie problemu i podstawowe definicje

Rozważany w tej pracy problem szeregowania w otwartym systemie obsługi można sformułować następująco: Dany jest zbiór zadań $J = \{1, 2, \ldots, n\}$, zbiór maszyn $M = \{1, 2, \ldots, m\}$ oraz zbiór operacji $O = \{1, 2, \ldots, o\}$, gdzie $o = n \cdot m$. Zadanie j, $j \in J$, składa się z m operacji indeksowanych przez $l_j + 1, l_j + 2, \ldots, l_j + m$, gdzie $l_j = (j-1)m$. Operacja $l_j + k$ jest wykonywana na maszynie k bez przerwań w czasie $p_i \geq 0$, $i = l_j + k$,

$j \in J$, $k \in M$. Zakładamy, że w każdej chwili czasowej: (i) co najwyżej jedna operacja każdego zadania może być wykonywana oraz (ii) każda maszyna może wykonywać co najwyżej jedną operację. Uszeregowanie dopuszczalne jest definiowane przez momenty rozpoczęcia wykonywania operacji $S_i \geq 0$, $i \in O$ spełniające wyżej sformułowane ograniczenia. Problem polega na znalezieniu uszeregowania dopuszczalnego minimalizującego moment wykonania wszystkich operacji $\max_{i \in O}(S_i + p_i)$.

Podstawowa różnica pomiędzy sformułowanym powyżej problemem (notowanym za Grahamem jako $O \mid\mid C_{max}$) a problemem gniazdowym $J \mid\mid C_{max}$ polega na tym, że kolejność wykonywania operacji danego zadania nie jest ustalona i podlega wyborowi. W konsekwencji dla określenia dopuszczalnego uszeregowania należy podać kolejność wykonywania operacji na maszynach oraz kolejność wykonywania operacji każdego zadania. Dalej, dla wygody rozważań, podamy inne sformułowanie problemu $O \mid\mid C_{max}$ wykorzystujące bezpośrednio te kolejności w tzw. modelu grafowym.

Zauważmy, że zbiór operacji O można podzielić na podzbiory $M_k = \{i : i = l_j + k, j \in J\}$, $k \in M$ oraz na podzbiory $J_j = \{i : i = l_j + k, k \in M\}$, $j \in J$. Podzbiór M_k określa operacje wykonywane na maszynie k, zaś J_j operacje wchodzące w skład zadania j. Zachodzi $O = \bigcup_{k=1}^{m} M_k$ oraz $O = \bigcup_{j=1}^{n} J_j$. Niech permutacja $\pi_k = (\pi_k(1), \pi_k(2), \ldots, \pi_k(n))$ określa kolejność wykonywania operacji ze zbioru M_k, a Π_k zbiór wszystkich takich permutacji, $k \in M$. Podobnie niech permutacja $\gamma_j = (\gamma_j(1), \gamma_j(2), \ldots, \gamma_j(m))$ określa kolejność wykonywania operacji ze zbioru J_j, a Γ_j zbiór wszystkich takich permutacji, $j \in J$. Wtedy kolejność wykonywania operacji na maszynach jest zdefiniowana przez m-tkę $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$, gdzie $\pi \in \Pi = \Pi_1 \times \Pi_2 \ldots \times \Pi_m$, zaś kolejność wykonywania operacji poszczególnych zadań przez n-tkę $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$, gdzie $\gamma \in \Gamma = (\Gamma_1, \Gamma_2, \ldots, \Gamma_n)$. Dla kolejności $\pi \in \Pi$ oraz $\gamma \in \Gamma$ określamy skierowany graf $G(\pi, \gamma) = (O, R(\gamma) \cup E(\pi))$ ze zbiorem wierzchołków O i zbiorem łuków $R(\gamma) \cup E(\pi)$. Każdy wierzchołek $i \in O$ ma wagę p_i. Łuki ze zbioru $E(\pi) = \bigcup_{k=1}^{m} \bigcup_{j=1}^{n-1} \{(\pi_k(j), \pi_k(j+1))\}$ precyzują kolejność wykonywania operacji na maszynach, zaś łuki $R(\gamma) = \bigcup_{j=1}^{n} \bigcup_{k=1}^{m-1} \{(\gamma_j(k), \gamma_j(k+1))\}$ kolejność wykonywania operacji poszczególnych zadań.

Dalej przedyskutujemy relację pomiędzy dopuszczalnością uszeregowania S_i, $i \in O$ a postacią wprowadzonego grafu. Będziemy mówić, że kolejność $(\pi, \gamma) \in \Pi \times \Gamma$ jest dopuszczalna, jeżeli istnieje uszeregowanie dopuszczalne S_i, $i \in O$, w którym kolejność wykonywania operacji jest określona przez (π, γ). Łatwo zauważyć, że kolejność (π, γ) jest dopuszczalna wtedy i tylko wtedy, gdy graf $G(\pi, \gamma)$ nie zawiera cykli. Co więcej, dla każdej dopuszczalnej kolejności $(\pi, \gamma) \in \Pi \times \Gamma$ można skonstruować wiele uszeregowań dopuszczalnych. Najlepsze z nich (w sensie rozważanego kryterium C_{max}) jest określone przez $S_i = \tau_i$, $i \in O$, gdzie τ_i jest długością najdłuższej ścieżki dochodzącej do wierzchołka i (bez niego) w grafie $G(\pi, \gamma)$; jeżeli wierzchołek i nie ma poprzedników, to $\tau_i = 0$. Niech $C_{max}(\pi, \gamma)$ oznacza długość najdłuższej ścieżki (ścieżka krytyczna) w grafie $G(\pi, \gamma)$. Teraz problem $O \mid\mid C_{max}$ można sformułować następująco:

Znaleźć kolejność $(\pi^*, \gamma^*) \in \Pi \times \Gamma$, taką że

$$C_{max}(\pi^*, \gamma^*) = \min_{(\pi,\gamma)\in\Pi\times\Gamma} C_{max}(\pi, \gamma). \tag{1}$$

W (1) nie uwzględniliśmy warunku na "niecykliczność" grafu $G(\pi, \gamma)$, ponieważ w grafie cyklicznym długość ścieżki krytycznej jest równa nieskończoności.

Niech $u = (u_1, u_2, \ldots, u_w)$ oznacza pewną ścieżkę krytyczną w $G(\pi, \gamma)$, gdzie $u_i \in O$, $i = 1, \ldots, w$, zaś w jest liczbą jej wierzchołków. Zachodzi $C_{max}(\pi, \gamma) = \sum_{i=1}^{w} p_{u_i}$.

Podamy teraz definicje tzw. *maszynowego bloku* operacji i *zadaniowego bloku* operacji na ścieżce krytycznej u. Definicje te uogólniają pojęcie bloku operacji wykorzystywane w problemie gniazdowym dla konstrukcji algorytmów dokładnych i algorytmów popraw. Niech $m(i)$ oraz $j(i)$ oznacza odpowiednio maszynę oraz zadanie związane z operacją i, $i \in O$. Podciąg ścieżki krytycznej $BM = (u_a, u_{a+1}, \ldots, u_b)$, gdzie $1 \leq a < b \leq w$, będziemy nazywać maszynowym blokiem operacji, jeżeli: (i) wszystkie operacje tego bloku są wykonywane na tej samej maszynie, tzn. $m(u_a) = m(u_{a+1}) = \ldots = m(u_b) \stackrel{\text{def}}{=} m(BM)$, ($ii$) operacja u_{a-1} (jeżeli $a > 1$) jest wykonywana na maszynie $m(u_{a-1}) \neq m(BM)$ oraz (iii) operacja u_{b+1} (jeżeli $a < w$) jest wykonywana na maszynie $m(u_{b+1}) \neq m(BM)$. Podobnie podciąg ścieżki krytycznej $BJ = (u_c, u_{c+1}, \ldots, u_d)$, gdzie $1 \leq c < d \leq w$, będziemy nazywać zadaniowym blokiem operacji, jeżeli: (i) wszystkie operacje tego bloku są związane z tym samym zadaniem, tzn. $j(u_c) = j(u_{c+1}) = \ldots = j(u_d) \stackrel{\text{def}}{=} j(BJ)$, ($ii$) operacja u_{c-1} (jeżeli $c > 1$) jest związana z zadaniem $j(u_{c-1}) \neq j(BJ)$ oraz (iii) operacja u_{d+1} (jeżeli $d < w$) jest związana z zadaniem $j(u_{d+1}) \neq j(BJ)$. Niech lm oraz lj oznacza odpowiednio liczbę bloków maszynowych oraz liczbę bloków zadaniowych na ścieżce u. Wtedy na ścieżce u występują maszynowe bloki operacji $BM_k = (u_{a_k}, u_{a_k+1}, \ldots, u_{b_k})$, $k = 1, 2, \ldots, lm$, gdzie $b_k < a_{k+1}$, $k = 1, 2, \ldots, lm - 1$ oraz zadaniowe bloki operacji $BJ_k = (u_{c_k}, u_{c_k+1}, \ldots, u_{d_k})$, $k = 1, 2, \ldots, lj$, gdzie $d_k < c_{k+1}$, $k = 1, 2, \ldots, lj - 1$. Każda operacja z u jest w jednym z trzech stanów: (s1) należy do bloku maszynowego i bloku zadaniowego, (s2) należy tylko do bloku maszynowego oraz (s3) należy tylko do bloku zadaniowego.

Sformułowane powyżej pojęcia zilustrujemy następującym przykładem: Rozważmy problem $O \,||\, C_{max}$, w którym występują cztery zadania ($n = 4$), cztery maszyny ($m = 4$) oraz szesnaście operacji ($o = 4 \cdot 4 = 16$). Zgodnie z przyjętą konwencją numeracji zadań zadanie 1 składa się z operacji 1,2,3,4, zadanie 2 z 5,6,7,8, zadanie 3 z 9,10,11,12, zaś zadanie 4 z 13,14,15,16. Niech $\pi = (\pi_1, \pi_2, \pi_3, \pi_4)$, gdzie $\pi_1 = (1, 5, 13, 9)$, $\pi_2 = (2, 10, 6, 14)$, $\pi_3 = (7, 3, 11, 15)$, $\pi_4 = (4, 8, 12, 16)$ oraz $\gamma = (\gamma_1, \gamma_2, \gamma_3, \gamma_4)$, gdzie $\gamma_1 = (1, 2, 3, 4)$, $\gamma_2 = (7, 8, 5, 6)$, $\gamma_3 = (10, 12, 11, 9)$, $\gamma_4 = (16, 15, 14, 13)$. Zauważmy, że kolejność (π, γ) jest kolejnością dopuszczalną. Dla tak określonej kolejności przyjmijmy, że ścieżka krytyczna u w grafie $G(\pi, \gamma)$ ma postać $u = (1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9)$. Wtedy $w = 11$ i na ścieżce u możemy wyróżnić dwa maszynowe bloki operacji ($lm = 2$) oraz dwa zadaniowe bloki operacji ($lj = 2$). Występuja one w następującej kolejności:

$BJ_1 = (1,2,3,4)$, $BM_1 = (4,8,12,16)$, $BJ_2 = (16,15,14,13)$, $BM_2 = (13,9)$, gdzie $c_1 = 1, d_1 = 4$, $a_1 = 4, b_1 = 7$, $c_2 = 7, d_2 = 10$, $a_2 = 10, b_2 = 11$. Operacje 4,16,13 są w stanie (s1), operacje 8,12,9 w stanie (s2), zaś 1,2,3,15,14 w stanie (s3).

3. Eliminacyjne własności bloków

W problemie gniazdowym ($J \mid\mid C_{max}$) kolejność γ w grafie $G(\pi,\gamma)$ jest ustalona i nie podlega zmianie. Bloki maszynowe na ścieżce u pozwalają wyspecyfikować permutacje $\beta \in \Pi$, dla których $C_{max}(\beta,\gamma)$ jest niemniejsze niż $C_{max}(\pi,\gamma)$. Permutacje te otrzymywane są przez zmianę kolejności wykonywania operacji należących do wnętrza bloku BM_k (tzn. operacji $u_{a_k+1}, u_{a_k+2}, \ldots, u_{b_k-1}$), $k = 1,2,\ldots,lm$. Eliminacja takich permutacji w procesie poszukiwań pozwala znacznie przyspieszyć znajdowanie satysfakcjonującego uszeregowania. Podobne eliminacyjne własności bloków maszynowych i bloków zadaniowych można sformułować dla rozważanego przez nas problemu. Niech para $(\pi,\gamma) \in \Pi \times \Gamma$ określa pewną kolejność dopuszczalną. Oznaczmy przez $\pi' \in \Pi$ permutację, która została otrzymana z permutacji π przez zmianę kolejności wykonywania operacji należących do wnętrza bloków BM_k, $k = 1,2,\ldots,lm$. Podobnie przez $\gamma' \in \Gamma$ oznaczmy permutację, która została otrzymana z permutacji γ przez zmianę kolejności wykonywania operacji należących do wnętrza bloków BJ_k, $k = 1,2,\ldots,lj$. Zbiór wszystkich takich par (π',γ') oznaczmy przez $\Omega(\pi,\gamma)$. Zachodzi następująca własność.

Własność 1. *Niech para $(\pi,\gamma) \in \Pi \times \Gamma$ oraz graf $G(\pi,\gamma)$ nie zawiera cykli. Dla każdej pary $(\pi',\gamma') \in \Omega(\pi,\gamma)$,*

$$C_{max}(\pi',\gamma') \geq C_{max}(\pi,\gamma). \tag{2}$$

Sformułowana powyżej własność umożliwia wyeliminowanie w procesie poszukiwań istotnej liczby uszeregowań, o których z góry wiemy (bez wyliczania C_{max}), że nie są "perspektywiczne", tzn. nie dostarczają uszeregowań o mniejszej wartości funkcji celu niż $C_{max}(\pi,\gamma)$. Łatwo zauważyć, że zbiór $\Omega(\pi,\gamma)$ zawiera

$$\prod_{k=1}^{lm}(|BM_k - 2|)! \cdot \prod_{k=1}^{lj}(|BJ_k - 2|)!$$

kolejności. Graf $G(\pi',\gamma')$, dla kolejności $(\pi',\gamma') \in \Omega(\pi,\gamma)$ może posiadać cykle. Rozważmy dalej zbiór $\Phi(\pi,\gamma)$ zawierający tylko takie kolejności, które otrzymuje się z (π,γ) przez wymianę położenia jednej pary sąsiednich operacji na ścieżce krytycznej u.

Własność 2. *Niech para $(\pi,\gamma) \in \Pi \times \Gamma$. Jeżeli graf $G(\pi,\gamma)$ nie zawiera cykli, to dla każdej pary $(\pi',\gamma') \in \Phi(\pi,\gamma)$ graf $G(\pi',\gamma')$ też nie zawiera cykli.*

Zauważmy, że $\Phi(\pi,\gamma) \not\subset \Omega(\pi,\gamma)$. Rzeczywiście, zbiór $\Phi(\pi,\gamma)$ zawiera także kolejności otrzymane przez wymianę położenia pary operacji (u_s, u_{s+1}), gdzie u_s jest pierwszą (lub

u_{s+1} jest ostatnią) operacją pewnego bloku, jeżeli ten blok ma co najmniej trzy elementy lub u_s jest pierwszą, a u_{s+1} ostatnią operacją tego bloku, jeżeli ma dwa elementy.

Zdefiniowane zbiory zilustrujemy na przykładzie z rozdziału 2. Zbiór $\Omega(\pi,\gamma)$ zawiera 8 elementów i ma postać $\Omega(\pi,\gamma) = \{(\pi',\gamma') \in \Pi \times \Gamma : \gamma'_1 \in \{(1,2,3,4),(1,3,2,4)\},\ \gamma'_4 \in \{(16,15,14,13),(16,14,15,13)\},\ \pi'_4 \in \{(4,8,12,16),(4,12,8,16)\},\ \gamma'_k = \gamma_k,\ k = 2,3;\ \pi'_k = \pi_k,\ k = 1,2,3\}$. Z kolei zbiór $\Phi(\pi,\gamma) = \bigcup_{k=1}^{4} \Phi_k(\pi,\gamma)$, gdzie $\Phi_1(\pi,\gamma) = \{(\pi,\gamma') : \gamma'_1 \in \{(2,1,3,4),(1,3,2,4),(1,2,4,3)\},\ \gamma'_k = \gamma_k,\ k = 2,3,4\}$, $\Phi_2(\pi,\gamma) = \{(\pi',\gamma) : \pi'_4 \in \{(8,4,12,16),(4,12,8,16),(4,8,16,12)\},\ \pi'_k = \pi_k,\ k = 1,2,3\}$, $\Phi_3(\pi,\gamma) = \{(\pi,\gamma') : \gamma'_4 \in \{(15,16,14,13),(16,14,15,13),(16,15,13,14)\},\ \gamma'_k = \gamma_k,\ k = 1,2,3\}$, oraz $\Phi_4(\pi',\gamma) = \{(\pi',\gamma) : \pi'_1 = (1,5,9,13),\ \pi'_k = \pi_k,\ k = 2,3,4\}$. Zbiory $\Phi_k(\pi,\gamma)$, $k = 1,2,3$ zawierają po trzy elementy a $\Phi_4(\pi,\gamma)$ jeden. Łącznie zbiór $\Phi(\pi,\gamma)$ ma $w - 1 = 10$ elementów.

4. Podstawowe elementy techniki tabu search

Technika tabu search została wyczerpująco przedstawiona w literaturze zagranicznej w [2], [3], [4], a w literaturze krajowej w [6], [7], [8]. Dlatego też tutaj ograniczymy nasze rozważania do kilku najistotniejszych elementów tej techniki takich jak: *ruch, otoczenie, lista tabu, poziom aspiracji* oraz *strategia przeglądania otoczenia*. Przez ruch będziemy rozumieć funkcję, która odwzorowuje jedno uszeregowanie w drugie uszeregowanie. Dla każdego uszeregowania określamy zbiór ruchów, który generuje otoczenie tego uszeregowania. Algorytmy oparte na technicę tabu search startują z pewnego uszeregowania początkowego. W każdym kroku iteracyjnym wszystkie ruchy skojarzone z danym uszeregowaniem (uszeregowanie początkowe) są przeglądane w celu znalezienia ruchu i odpowiadającego mu uszeregowania o najmniejszej wartości funkcji celu. Następnie, *niezależnie* od relacji między wartością funkcji celu dla znalezionego uszeregowania a wartością dla uszeregowania początkowego, ruch zostaje wykonany. Oznacza to, że znalezione uszeregowanie traktujemy jako uszeregowanie początkowe do następnego kroku. W celu zabezpieczenia się przed wpadnięciem w cykl, tzn. generowaniem cyklicznie pewnego ciągu uszeregowań, wprowadzamy listę ruchów zabronionych (lista tabu) modyfikowaną w każdej iteracji. Duże restrykcje wprowadzane przez listę tabu są zmniejszane przez określenie poziomu aspiracji. Polega to na tym, że ruchy zabronione, które produkują uszeregowania o wartości funkcji celu mniejszej niż poziom aspiracji, też można wykonywać.

Przedstawimy teraz krótko implementacje sformułowanych powyżej podstawowych elementów techniki tabu search, dla analizowanego w tej pracy problemu $O \mid\mid C_{max}$.

Typowym ruchem rozważanym w problemie gniazdowym jest wymiana położenia sąsiedniej pary operacji, które należą do ścieżki krytycznej u i są wykonywane na tej samej maszynie (tzn. należą do pewnego bloku maszynowego), [12]. Ruch ten jest w pełni opisany przez podanie dla kolejności (π,γ) pary operacji $v = (x,y)$, takiej że $x = u_k$, $y = u_{k+1}$ dla pewnego $1 \le k < w$, oraz $m(x) = m(y)$. Przez analogię, dla problemu

$O \mid\mid C_{max}$ w pracy [11] rozważono ruchy $v = (x, y)$ polegające na wymianie położenia dowolnej sąsiedniej pary operacji z u. Zbiór wszystkich takich ruchów dla kolejności (π, γ) określmy przez $V'(\pi, \gamma)$. Zauważmy, że $\mid V'(\pi, \gamma) \mid = w - 1$ oraz każda para operacji $v \in V'(\pi, \gamma)$ należy do pewnego bloku maszynowego lub zadaniowego. Przez $(\pi_v, \gamma_v) \in \Pi \times \Gamma$ oznaczmy kolejność otrzymaną z kolejności (π, γ) przez wykonanie ruchu v. Wtedy otoczenie kolejności (π, γ) można określić następująco:

$$\mathcal{N}'(\pi, \gamma) = \{(\pi_v, \gamma_v) : v \in V'(\pi, \gamma)\} \tag{3}$$

Taka definicja zbioru ruchów wymaga w każdej iteracji algorytmu wyliczenia $w - 1$ razy wartości funkcji celu, co implikuje duże nakłady obliczeniowe - szczególnie gdy w jest porównywalne z n. Dlatego proponujemy znaczne zmniejszenie liczności zbioru ruchów bazując na eliminacyjnych własnościach bloków (podobnie jak w [9], [10] dla problemów przepływowych i gniazdowych). Niech $V(\pi, \gamma) = \bigcup_{k=1}^{lm} VM_k(\pi, \gamma) \cup \bigcup_{k=1}^{lj} VJ_k(\pi, \gamma)$, gdzie

$$VM_k(\pi, \gamma) = \{(u_{a_k}, u_{a_k+1}), (u_{b_k-1}, u_{b_k})\}, \quad k = 1, 2, \ldots, lm, \tag{4}$$

$$VJ_k(\pi, \gamma) = \{(u_{c_k}, u_{c_k+1}), (u_{d_k-1}, u_{d_k})\}, \quad k = 1, 2, \ldots, lj. \tag{5}$$

Zbiór $V(\pi, \gamma)$ zawiera ruchy otrzymane przez wymianę położenia tylko dwóch początkowych i dwóch końcowych operacji każdego bloku. Zbiór ten generuje otoczenie

$$\mathcal{N}(\pi, \gamma) = \{(\pi_v, \gamma_v) : v \in V(\pi, \gamma)\}. \tag{6}$$

Przedyskutujemy teraz własności otoczeń $\mathcal{N}'(\pi, \gamma)$, $\mathcal{N}(\pi, \gamma)$. Zauważmy, że $\mathcal{N}(\pi, \gamma) \subset \mathcal{N}'(\pi, \gamma) = \Phi(\pi, \gamma)$, oraz $(\mathcal{N}'(\pi, \gamma) \setminus \mathcal{N}(\pi, \gamma)) \subset \Omega(\pi, \gamma)$. Stąd i z Własności 1,2 mamy:

Własność 3. *Niech para $(\pi, \gamma) \in \Pi \times \Gamma$ oraz graf $G(\pi, \gamma)$ nie zawiera cykli.*
(i) Dla każdej pary $(\pi', \gamma') \in \mathcal{N}(\pi, \gamma)$, graf $G(\pi', \gamma')$ jest acykliczny.
(ii) Dla każdej pary $(\pi', \gamma') \in (\mathcal{N}'(\pi, \gamma) \setminus \mathcal{N}(\pi, \gamma))$, $C_{max}(\pi', \gamma') \geq C_{max}(\pi, \gamma)$.

Otoczenie $\mathcal{N}(\pi, \gamma)$ ma co najwyżej $2(lm + lj)$ elementów (w przykładzie z rozdziału 2, $\mid \mathcal{N}(\pi, \gamma) \mid = 7 < w - 1 = 10$). W badaniach testowych okazało się, że przedstawiona redukcja otoczenia, polegająca na wyrzuceniu "nieperspektywicznych" kolejności, znacznie zmniejsza czas potrzebny na znalezienie satysfakcjonującego uszeregowania.

Kiedy tylko ruch $v = (x, y)$ jest wykonywany, ruch odwrotny (y, x) jest wprowadzany na listę T zabronionych ruchów. Precyzyjniej wykonujemy operację $T := T \oplus v$, gdzie $\oplus$ definiujemy następująco: $T_i := T_{i+1}$, $i = 1, 2, \ldots, maxt - 1$, $T_{maxt} := (y, x)$ oraz $maxt$ jest długością listy T. Początkowo wszystkie elementy listy są zerowane. W danej iteracji algorytmu ruch v uważamy za zabroniony, jeżeli znajduje się na liście T. Jako poziom aspiracji przyjmujemy najmniejszą znalezioną dotychczas wartość funkcji celu C^*.

Strategia przeglądania otoczenia w danej iteracji polega na określeniu dwóch podzbiorów A oraz B zbioru ruchów $V(\pi, \gamma)$, gdzie (π, γ) jest uszeregowaniem początkowym

dla tej iteracji. Zbiór $A = \{v \in V(\pi,\gamma) : v \notin T\}$ zawiera ruchy niezabronione, zaś $B = \{v \in V(\pi,\gamma) : v \in T,\ C_{max}(\pi_v,\gamma_v) < C^*\}$ - "interesujące" nas (przez poziom aspiracji) ruchy zabronione. Następnie jeżeli zbiór $A \cup B$ jest niepusty, to wybieramy ruch z tego zbioru, który generuje uszeregowanie o najmniejszej wartości funkcji celu. W przeciwnym wypadku najstarsze ruchy zabronione z listy tabu T są usuwane, aż zbiór A stanie się niepusty i wybieramy ruch wg. procedury opisanej powyżej.

5. Algorytm i wyniki badań testowych

Podamy teraz szczegółowy opis proponowanego przez nas algorytmu *TS*. Uszeregowanie początkowe (π^p,γ^p) wygenerowano wg reguły: wolnej maszynie przydziel operację i o największym p_i spośród operacji, które mogą być aktualnie wykonywane (operacja i może być aktualnie wykonywana, jeżeli $j(i) \neq j(k)$ dla wszystkich operacji k już wykonywanych), [1]. W *TS* wprowadzono dwa warunki stopu: (i) wykonanie *maxiter* iteracji bez zmniejszenia wartości funkcji celu oraz (ii) globalne wykonanie *maxglobiter* iteracji. Liczniki *iter*1 oraz *iter*2 zliczają wykonane iteracje odpowiednio dla (i) oraz (ii).

Algorytm *TS*

Krok 0. Połóż $iter1 := 0$, $iter2 := 0$, $\pi := \pi^p$, $\gamma := \gamma^p$, $C := C_{max}(\pi,\gamma)$, $\pi^* := \pi^p$, $\gamma^* := \gamma^p$, $C^* := C$ oraz $T_i := (0,0)$ dla $i = 1,2,\ldots,maxt$.

Krok 1. Połóż $iter1 := iter1 + 1$, $iter2 := iter2 + 1$. Wyznacz zbiór ruchów $V(\pi,\gamma)$, zbiór ruchów niezabronionych $A = \{v \in V(\pi,\gamma) : v \notin T\}$ oraz zbiór interesujących ruchów zabronionych $B = \{v \in V(\pi,\gamma) : v \in T,\ C_{max}(\pi_v,\gamma_v) < C^*\}$.

Krok 2. Jeżeli $A \cup B \neq \emptyset$, to wyznacz ruch $v' \in A \cup B$ taki, że $C_{max}(\pi_{v'},\gamma_{v'}) = \min_{v \in A \cup B} C_{max}(\pi_v,\gamma_v)$ i idź do *Kroku 3.* W przeciwnym wypadku zmodyfikuj listę tabu kładąc $T := T \oplus (0,0)$, wyznacz ponownie zbiór A i idź do *Kroku 2.*

Krok 3. Zmodyfikuj listę tabu kładąc $T := T \oplus v'$ i przejdź do nowej kolejności $\pi := \pi_{v'}$, $\gamma := \gamma_{v'}$, $C := C_{max}(\pi_{v'},\gamma_{v'})$.

Krok 4. Jeżeli $C < C^*$, to połóż $\pi^* := \pi, \gamma^* := \gamma$, $C^* := C$ oraz $iter1 := 0$.

Krok 5. Jeżeli $iter1 = maxiter$ lub $iter2 = maxglobiter$, to *STOP*. W przeciwnym wypadku idź do *Kroku 1.*

Algorytm *TS* ma trzy parametry sterujące *maxt*, *maxiter* oraz *maxglobiter*, których wartości mają wpływ na jakość produkowanego rozwiązana i czas jego otrzymania. Przyjęto $maxt = 8$, $maxiter = 5000$ oraz $maxglobiter = 50000$.

Wstępne badania testowe algorytmu przeprowadzono na przykładach zaczerpniętych z pracy [11]. Są to przykłady, w których liczba zadań jest równa liczbie maszyn i zmienia

się od 4 do 20. Dla każdego $n = 4, 5, 7, 10, 15, 20$ wygenerowano w [11] odpowiednio 50000, 45000, 1000, 300, 40, 25 przykładów i wybrano 10 szczególnie trudnych instancji. W konsekwencji otrzymano $6 \cdot 10$ przykładów. Algorytm *TA* z pracy [11], który jest aktualnie najlepszym algorytmem popraw, rozwiązywał je wykonując od $5 \cdot 10^5$ do $9 \cdot 10^6$ iteracji. Czas jednej iteracji wynosił $23 \cdot 10^{-6} n^2$ sekund na Silicon Graphics (10 Mips).

Zaproponowany przez nas algorytm *TS* został zakodowany w Pascalu i był uruchamiany na minikomputerze IBM PC DX486 o tej samej prędkości 10 Mips. Po wykonaniu $5 \cdot 10^4$ iteracji wyprodukowano uszeregowania, z których 14 ma mniejszą wartość funkcji celu niż otrzymana przez *TA*, 43 równą i tylko 3 większą. Po zwiększeniu parametru *maxglobiter* do wartości 60000 także dla tych trzech przykładów otrzymano uszeregowanie o mniejszej wartości funkcji celu. Czas wykonania jednej iteracji jest około 4 razy krótszy niż w [11]. Reasumując, algorytm *TS* znajduje uszeregowania lepsze lub takie same jak *TA* w czasie od 40 do 180 razy krótszym na komputerze o porównywalnej mocy obliczeniowej. Aktualnie algorytm *TS* jest testowany na przykładach o wymiarach $n = 50, 80, 100$. Otrzymywane wyniki są bardzo zachęcające.

6. Uwagi końcowe

W pracy podano bardzo szybki algorytm aproksymacyjny typu tabu search dla otwartego systemy obsługi. Aktualnie jest on modyfikowany przez wprowadzenie koncepcji tzw. powrotów. Koncepcja ta polega na tym, że po wykonaniu *maxiter* iteracji bez poprawy wartości funkcji celu algorytm nie kończy działania, ale wraca do najlepszego znalezionego uszeregowania i dalej kontynuuje poszukiwania ze zmienioną listą tabu.

Inny kierunek prac polega na adaptacji wprowadzonego otoczenia dla problemów, w których w jednym stanowisku obsługi znajduje się kilka (zamiast jednej) maszyn. Problemy te są szczególnie istotne przy sterowaniu w elastycznych systemach produkcyjnych.

LITERATURA

[1] Chen B., Strusevich, V.A.: Worst-case analysis of heuristics for open shop with parallel machines. European Journal of Operation Research 70, 1993, 379-390.

[2] Glover F.: Future path for integer programming and links to artificial intelligence. Computers and operation research 13, 1986, 533-549.

[3] Glower F.: Tabu search. Part I. ORSA Journal on Computing 1, 1989, 190-206.

[4] Glover F., Laguna M.: Tabu search. (A Chapter in Modern Heuristic Techniques for Combinatorial Problems). Report Department of Computer Science and Institute of Cognitive Science, University of Colorado at Boulder, Jul 24, 1993.

[5] Hübscher R., Glover F.: Applying tabu search with influential diversification to

multiprocessor scheduling. Report Department of Computer Science and Institute of Cognitive Science, University of Colorado at Boulder, October 11, 1992.

[6] Kadłuczka P., Wala K.: Tabu search heuristic for generalized graph partitioning problem. Zeszyty Naukowe AGH seria: Automatyka z. 64, Kraków, 1993, 473-487.

[7] Korcyl A., Sawik T.: Algorytmy typu tabu search dla wyznaczania partii produkcyjnej i obciążenia maszyn w elastycznym systemie produkcyjnym. Zeszyty Naukowe Politechniki Śląskiej z. 109 , 1992, 147-157.

[8] Nowicki E.: Zastosowanie techniki przeszukiwania z zabronionymi ruchami do problemów przepływowych z ustalonymi i zmiennymi czasami wykonywania zadań. Zeszyty Naukowe AGH seria: Automatyka z. 64, Kraków, 1993, 323-335.

[9] Nowicki E., Smutnicki C.: A fast tabu search algorithm for the job shop problem. Raport serii: Preprinty ICT 8/1993 Wydawnictwo Politechniki Wrocławskiej (w redakcji Management Science).

[10] Nowicki E., Smutnicki C.: A fast tabu search algorithm for the flow shop problem. Raport serii: Preprinty ICT 8/1994 Wydawnictwo Politechniki Wrocławskiej (w redakcji European Journal of Operational Research).

[11] Taillard E.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 1993, 278-285.

[12] Taillard E.: Parallel Taboo Search Technique for the Jobshop Scheduling Problem. Working Paper ORWP 89/11 (revised version October 1992), Departement de Mathematiques, Ecole Polytechnique Federale De Lausanne, Lausanne, Switzerland, 1989.

Recenzent: Prof.dr hab.inż. Tadeusz Sawik

Wpłynęło do Redakcji do 30.04.1994 r.

Abstract

A fast and easily implemented approximation algorithm for the problem of finding a minimum makespan in an open-shop is presented. The algorithm is based on a tabu search technique with a specific neighborhood. A substantially small neighborhood is defined using so-called blocks of operations on a critical path. The essence of the algorithm consists in basing on properties which allow us to search over the most interesting part of a neighborhood (using the makespan). We show experimentally (having tested on commonly used benchmarks up to 400 operations) that the algorithm can find shorter makespans than the best approximation approaches and also runs in shorter time. Moreover, it was successfuly run on random benchmarks up to 10.000 operations on a Personal Computer AT 486DX. The algorithm is easily tuned (primal values of tuning parameters are recommended).