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

Eugeniusz NOWICKI
Politechnika Wrocławska

ZASTOSOWANIE TECHNIKI TABU DO HARMONOGRAMOWANIA ELASTYCZNYCH GNIAZD PRODUKCYJNYCH Z CZASAMI TRANSPORTU I PRZEZBROJEŃ

Streszczenie. W pracy przedstawia się szybki aproksymacyjny algorytm, minimalizujący moment wykonania wszystkich zadań, dla problemu gniazdowego z maszynami równoległymi, czasami transportu i czasami przezbrojeń. Algorytm jest oparty na technice Tabu ze specyficzną definicją sąsiedztwa. Specjalna metoda implementacji istotnie zwiększa prędkość algorytmu.

APPLICATION OF TABU SEARCH TECHNIQUE TO SCHEDULING A FLEXIBLE JOB SHOP WITH TRANSPORT AND SETUP TIMES

Summary. A fast approximation algorithm for a problem of finding the minimum makespan in a job shop with parallel machines, transport times and setup times is presented. The algorithm is based on a Tabu search technique with a specific neighborhood definition. A special method of implementation increases the speed of the algorithm significantly.

1. Wprowadzenie

Praca dotyczy problemu szeregowania zadań produkcyjnych w elastycznych gniazdach produkcyjnych z kryterium minimalizującym moment wykonania wszystkich zadań. Przyjmuje się, że w każdym gnieździe produkcyjnym znajduje się określona liczba, niekoniecznie identycznych maszyn. Zadanie produkcyjne jest reprezentowane przez pewien ustalony ciąg operacji. Dla każdej operacji określone jest gniazdo produkcyjne, w którym może być ona tylko wykonywana. Realizacja operacji wymaga zaangażowania jednej (dowolnej) maszyny z tego gniazda przez zadany czas (ogólnie różny dla różnej maszyny). Zakłada się, że pomiędzy dwoma kolejno wykonywanymi operacjami na danej maszynie wymagany jest czas przezbrojenia, który zależy od tych operacji i maszyny. Dodatkowo uwzględnia się niezerowe czasy transportu zadań pomiędzy gniazdami produkcyjnymi.

Elastyczność gniazd produkcyjnych jest tu rozumiana przede wszystkim w kontekście możliwości wyboru różnych marszrut technologicznych dla poszczególnych zadań produkcyjnych. Z kolei przyjęty model transportu odpowiada sytuacji "czekaj" i "jedź" (ang. stop and go) polegającej na tym, że zadanie jest związane z wózkiem transportowym od początku do końca jego procesu produkcyjnego; w trakcie wykonywania operacji danego zadania wózek czeka na jej wykonanie, by następnie przetransportować zadanie do kolejnego gniazda produkcyjnego.

Aktualnie do rozwiązania badanego problemu poleca się w literaturze stosowanie przede wszystkim algorytmów przybliżonych typu konstrukcyjnego, patrz np. [9]. Algorytmy te na podstawie pewnych reguł priorytetowych (dyspozytorskich) przydzielają kolejno pojedyncze operacje do maszyn i jednocześnie szeregują je na nich. Czas ich pracy jest bardzo krótki, ale jakość produkowanych przez nie harmonogramów nie jest zadowalająca.

W tej pracy proponujemy zastosowanie algorytmu "popraw" opartego na metaheurystyce Tabu, który jest uogólnieniem algorytmu przedstawionego w [4] dla uproszczonej wersji rozważanego problemu z zerowymi czasami transportu i zerowymi czasami przezbrojeń; podobnego typu algorytmy dla tak uproszczonej wersji problemu podano także w [1], [3]. Specyficzna konstrukcja sąsiedztwa dla danego uszeregowania oraz wykorzystanie odpowiednich własności do „szybkiego" jego przeszukania pozwoliło znacznie zredukować czasochłonność wyboru najlepszego rozwiązania z sąsiedztwa (podobnie jak w pracach [5], [6] oraz [7] dla problemów przepływowych). Fakt ten w połączeniu z odpowiednim doborem ruchu, jego atrybutów, efektywną metodą określania statusu ruchu oraz metodą tzw. skoku powrotnego powoduje, że prezentowany algorytm dostarcza istotnie lepszych rozwiązań niż najlepsze rozwiązania otrzymane przez jednoczesne zastosowanie kilkunastu powszechnie używanych algorytmów konstrukcyjnych dla tego typu problemów, w maksymalnym czasie rzędu kilku minut na IBM PC. Rozwiązywane w tym czasie zagadnienia mają rozmiar akceptowalny w praktyce (kilka tysięcy operacji, kilkaset zadań, kilkanaście stanowisk oraz kilka maszyn w stanowisku).

2. Matematyczne sformułowanie problemu i model grafowy

Dany jest zbiór $\mathcal{M} = \{1, \ldots, m\}$ maszyn podzielonych na ls rozłącznych podzbiorów $\mathcal{M}_k = \{\omega_k + 1, \omega_k + 2, \ldots, \omega_k + m_k\}$, $k = 1, \ldots ls$, utożsamianych ze stanowiskami (gniazdami produkcyjnymi), gdzie m_k jest liczbą maszyn w stanowisku k, $\omega_k = \sum_{h=1}^{k-1} m_h$ - liczbą maszyn w stanowiskach $1, \ldots, k-1$ oraz $\sum_{k=1}^{ls} m_k = m$. Należy wykonać r zadań

ze zbioru $\mathcal{J} = \{1, 2, \ldots, r\}$. Zadanie i, $i \in \mathcal{J}$ składa się z $o_i \geqslant 1$ operacji indeksowanych przez $\tau_i + 1, \tau_i + 2, \ldots, \tau_i + o_i$ i wykonywanych w tej kolejności, gdzie $\tau_i = \sum_{h=1}^{i-1} o_h$ oraz $\tau_1 = 0$. Łącznie w systemie trzeba wykonać $n = \sum_{i \in J} o_i$ operacji $\mathcal{O} = \{1, 2, \ldots, n\}$. Dla każdej operacji $j \in \mathcal{O}$ określone jest stanowisko $1 \leqslant \mu(j) \leqslant ls$; operacja j może być wykonana na jednej dowolnie wybranej maszynie $l \in \mathcal{M}_{\mu(j)}$ z tego stanowiska w czasie $p_{l,j} \geqslant 0$. Pomiędzy wykonywaniem dwóch kolejnych operacji j', j'' na maszynie $l \in \mathcal{M}$ wymagany jest czas przezbrojenia $s_l(j', j'') \geqslant 0$ spełniający warunek "trójkąta"; jeżeli operacja j wykonuje się jako pierwsza na maszynie l, to czas przezbrojenia równa się $s_l(0, j) \geqslant 0$. Ponadto, transport zadania pomiędzy każdą parą różnych stanowisk $k', k'' \in \{1, 2, \ldots, ls\}$ wymaga $t_{k',k''} > 0$ czasu i nie zależy od transportowanego zadania; $t_{k,k} = 0$, $1 \leqslant k \leqslant ls$.

Podobnie jak w klasycznym problemie gniazdowym, nie dopuszczamy podzielności operacji i zakładamy, że w każdej chwili czasowej: (i) każda maszyna w każdym stanowisku może wykonywać co najwyżej jedną operację oraz (ii) nie można jednocześnie wykonywać więcej niż jednej operacji danego zadania (w tych samych lub różnych stanowiskach). Uszeregowanie dopuszczalne definiujemy jako zbiór par $(m(j), S(j))$, $j \in \mathcal{O}$ takich, że powyższe ograniczenia są spełnione, gdzie $m(j) \in \mathcal{M}_{\mu(j)}$ jest maszyną, która została wybrana w stanowisku $\mu(j)$ do wykonywania operacji j, zaś $S(j)$ - momentem rozpoczęcia wykonywania tej operacji na wybranej maszynie $m(j)$. Problem polega na znalezieniu dopuszczalnego uszeregowania minimalizującego moment wykonania wszystkich operacji $\max_{j \in \mathcal{O}} (S(j) + p_{m(j),j})$.

Przejdziemy teraz do sformułowania modelu grafowego. W tym celu wprowadzimy pewne dodatkowe oznaczenia, takie same jak w [5]. Niech zestaw $(\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_m)$ określa podział zbioru operacji $\mathcal{O}$ na m rozłącznych podzbiorów takich, że $l \in \mathcal{M}_{\mu(j)}$ dla każdej operacji $j \in \mathcal{O}_l$ oraz maszyny $l \in \mathcal{M}$. Zbiór $\mathcal{O}_l$ zawiera $n_l = |\mathcal{O}_l|$ operacji, które zostały wybrane do wykonywania na maszynie l; $\sum_{l \in \mathcal{M}} n_l = n$. Niech kolejność wykonywania operacji $\mathcal{O}_l$ wyznacza permutacja $\pi_l = (\pi_l(1), \ldots, \pi_l(n_l))$. Wtedy zestaw permutacji $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$ – zwany dalej też permutacją – jest zmienną decyzyjną określającą kolejność wykonywania wszystkich operacji ze zbioru $\mathcal{O}$ na poszczególnych maszynach; zbiór wszystkich takich permutacji oznaczamy przez Π.

Wykorzystując wprowadzone oznaczenia dla permutacji π, $\pi \in \Pi$ definiujemy skierowany graf

$$G(\pi) = (\mathcal{O} \cup \{0\}, E^T \cup E^K(\pi)) \tag{1}$$

ze zbiorem wierzchołków $\mathcal{O} \cup \{0\}$ reprezentującym poszczególne operacje (plus wierzchołek

0 odpowiadający operacji fikcyjnej) oraz zbiorami łuków

$$E^T = \bigcup_{k\in\mathcal{J}} \bigcup_{j=\tau_k+2}^{\tau_k+o_k} \{(j-1,j)\}, \tag{2}$$

$$E^K(\pi) = \bigcup_{l\in\mathcal{M}} \bigcup_{i=1}^{n_l} \{(\pi_l(i-1),\pi_l(i))\}, \tag{3}$$

reprezentującymi odpowiednio kolejność wynikającą z marszruty technologicznej poszczególnych operacji w zadaniach oraz kolejność π wykonywania operacji na maszynach; $\pi_l(0) = 0$ dla $l \in \mathcal{M}$. Każdy wierzchołek $j = \pi_l(i)$, $1 \leqslant i \leqslant n_l$, $l \in \mathcal{M}$ ma wagę $p_{l,j}$; wierzchołek 0 ma wagę zero. Każdy łuk $(\pi_l(i-1),\pi_l(i))$ ze zbioru $E^K(\pi)$ ma wagę $s_l(\pi_l(i-1),\pi_l(i))$, a każdy łuk $(j-1,j)$ ze zbioru E^T ma wagę $t_{\mu(j-1),\mu(j)}$.

Niech $C_{\max}(\pi)$ oznacza długość najdłuższej ścieżki (ścieżka krytyczna) w grafie $G(\pi)$ dla permutacji $\pi \in \Pi$; w przypadku gdy graf $G(\pi)$ jest cykliczny, przyjmujemy, że $C_{\max}(\pi) = \infty$. Ostatecznie rozważany problem możemy sformułować jako znalezienie takiej permutacji $\pi^* \in \Pi$, że odpowiadający jej graf $G(\pi^*)$ ma najmniejszą długość ścieżki krytycznej $C_{\max}(\pi^*)$.

Z przedstawionych rozważań wynika, że najistotniejszym elementem przy analizie badanego problemu jest graf $G(\pi)$, a w szczególności ścieżka krytyczna w tym grafie zapisywana dalej jako ciąg $u = (u_1, u_2, \ldots, u_{lu})$ wierzchołków (operacji), których liczbę oznaczamy przez lu (w celu uproszczenia nie uwzględniamy wierzchołka 0). Ścieżkę u można przedstawić, podobnie jak dla szeregu innych problemów, w postaci konkatenacji pewnej liczby (oznaczmy ją przez lb) zwartych podciągów operacji wykonywanych na tej samej maszynie, zwanych blokami. Dokładniej,

$$u = (B_1, B_2, \ldots, B_{lb}), \tag{4}$$

gdzie $B_h = (\pi_{l_h}(e_h), \pi_{l_h}(e_h+1), \ldots, \pi_{l_h}(f_h))$, $1 \leqslant h \leqslant lb$ jest ciągiem (blokiem) zawierającym operacje wykonywane na maszynie oznaczanej przez l_h poczynając od operacji z pozycji e_h, $1 \leqslant e_h \leqslant n_{l_h}$, a kończąc na operacji z pozycji f_h, $e_h \leqslant f_h \leqslant n_{l_h}$, w permutacji π_{l_h}. Dodatkowo przyjmujemy, że operacje z dwóch kolejnych bloków B_{h-1}, B_h są wykonywane na różnych maszynach, tzn. $l_{h-1} \neq l_h$, $h = 2, \ldots, lb$. Zauważmy, że wierzchołek $\pi_{l_{h-1}}(f_{h-1})$ odpowiadający ostatniej operacji z bloku B_{h-1} jest połączony z wierzchołkiem $\pi_{l_h}(e_h)$ odpowiadającym pierwszej operacji z bloku B_h, łukiem z E^T. Wierzchołki odpowiadające sąsiednim operacjom bloku są połączone łukami z $E^T \cup E^K(\pi)$.

3. Eliminacyjne własności bloków dla ruchów typu włóż

Istotnym elementem techniki Tabu, [2], [7] jest pojęcie ruchu, tzn. funkcji v, która odwzorowuje zbiór Π w Π. Zdefiniowanie dla danej permutacji $\pi \in \Pi$ ruchu v, a dokładniej określenie zbioru wykonywanych z tej permutacji ruchów $V(\pi)$, pozwala określić jej sąsiedztwo $\mathcal{N}(\pi) = \{\pi_v : v \in V(\pi)\}$, czyli podstawowy element każdej metody "popraw"; w technice Tabu wartość funkcji v w punkcie π oznacza się przez π_v zamiast przez $v(\pi)$. Wśród szeregu różnych ruchów szczególnie polecane dla problemów szeregowania są ruchy typu "włóż" (ang. insert) i typu "wymień" (ang. swap). Co więcej, te pierwsze, w odróżnieniu od drugich, pozwalają na zbudowanie efektywnych czasowo metod przeglądania sąsiedztwa, tzn. znajdowania ruchu z $V(\pi)$ prowadzącego do permutacji o najmniejszej wartości funkcji celu.

Dla $\pi \in \Pi$ ruch $v = (a, x, b, y)$ typu włóż polega na pobraniu operacji $j = \pi_a(x)$ z pozycji $1 \leqslant x \leqslant n_a$ w permutacji π_a na maszynie $a \in \mathcal{M}_{\mu(j)}$ ze stanowiska $\mu(j)$ i włożeniu jej w permutację π_b na pewnej maszynie $b \in \mathcal{M}_{\mu(j)}$ tego stanowiska na pozycję $1 \leqslant y \leqslant n_b + 1$ (lub na pozycję $1 \leqslant y \leqslant n_a$, $y \neq x$, jeżeli $b = a$); w trakcie wykonywania ruchu odpowiednie operacje są przesuwane o jedną pozycję w lewo lub w prawo. Dalej przez $V^o(\pi)$ oznaczmy zbiór wszystkich możliwych ruchów typu włóż z permutacji $\pi \in \Pi$. W pracy [7] pokazano, że zbiór ten zawiera $\sum_{k=1}^{ls} ns_k(ns_k + m_k - 3) + m_k$ ruchów, gdzie $ns_k = |\{j \in \mathcal{O} : \mu(j) = k\}|$ jest liczbą operacji, które muszą być wykonane w stanowisku k, $k = 1, \ldots, ls$.

Ogólnie, zbiór ruchów $V^o(\pi)$ można podzielić na dwa rozłączne podzbiory $VA^o(\pi) = \{v \in V^o(\pi) : C_{\max}(\pi_v) \geqslant C_{\max}(\pi)\}$ oraz $VB^o(\pi) = \{v \in V^o(\pi) : C_{\max}(\pi_v) < C_{\max}(\pi)\}$. Do pierwszego z nich należą ruchy v, które nie zmniejszają wartości funkcji celu, a do drugiego – pozostałe. Zdefiniowane w poprzednim rozdziale bloki operacji pozwalają prosto określić niektóre ruchy ze zbioru $VA^o(\pi)$. Odpowiednie warunki, które je definiują, są bardzo podobne w zapisie do warunków zamieszczonych w [5] przy analizie uogólnionych problemów przepływowych. Tym niemniej, ze względu na znaczną różnice pomiędzy ww. problemami a rozważanym tutaj problemem zostaną one dalej podane.

Dla ruchu $v = (a, x, a, y)$, $x \neq y$, $e_h < x < f_h$, $e_h < y < f_h$ polegającego na przesuwaniu operacji $j = \pi_a(x)$ wewnątrz pewnego bloku B_h $(a = l_h)$ definiujemy wielkość $\Delta^\pi(v) = -\Delta_1^\pi(v) + \Delta_2^\pi(v)$, gdzie $\Delta_1^\pi(v) = s_a(i', j) + s_a(j, i'') - s_a(i', i'')$, $\Delta_2^\pi(v) = s_a(k', j) + s_a(j, k'') - s_a(k', k'')$; w definicji przez i' oraz i'' (k', k'') oznaczyliśmy odpowiednio bezpośredniego poprzednika oraz bezpośredniego następnika operacji j

z $E^K(\pi)$ w grafie $G(\pi)$ (w grafie $G(\pi_v)$). Wielkość $\Delta^\pi(v)$, podobnie jak w [5], oznacza wzrost długości ścieżki w grafie $G(\pi_v)$ przechodzącej przez wszystkie wierzchołki ścieżki u w stosunku do $C_{\max}(\pi)$, spowodowany zmianą czasów przezbrojeń wynikających z innego umiejscowienia operacji j w B_h.

Własność 1. *Niech permutacja $\pi \in \Pi$ generuje acykliczny graf $G(\pi)$ ze ścieżką krytyczną u i blokami operacji B_h, $h = 1, \ldots, lb$. Niech ruch $v = (a, x, b, y) \in V^o(\pi)$. Jeżeli operacja $j = \pi_a(x)$ spełnia jeden z poniższych pięciu warunków:*
(i) nie należy do żadnego bloku,
(ii) należy do bloku jednoelementowego oraz $p_{b,j} \geqslant p_{a,j}$,
(iii) należy do bloku co najmniej dwuelementowego B_h, $b = a$ oraz
(a) $x = e_h$, $1 \leqslant y < e_h$ lub
(b) $e_h < x < f_h$, $e_h < y < f_h$, $\Delta^\pi(v) \geqslant 0$ lub
(c) $x = f_h$, $f_h < y \leqslant n_{l_h}$,
to ruch v należy do zbioru $VA^o(\pi)$.

Jest oczywiste, że gdybyśmy nawet dysponowali efektywną czasowo procedurą identyfikacji wszystkich ruchów ze zbioru $VA^o(\pi)$, to przy definicji zbioru ruchów $V(\pi)$ – generującego sąsiedztwo permutacji π – nie ograniczylibyśmy się tylko do zbioru $VB^o(\pi)$, ponieważ dla większości permutacji $\pi \in \Pi$ jest to zbiór pusty; w konsekwencji nasza metoda popraw bardzo szybko zatrzymałaby się w obszarze minimum lokalnego. Ruchy określone w powyższej własności stanowią jednak tylko pewną część ruchów ze zbioru $VA^o(\pi)$ i dlatego jako zbiór $V(\pi)$ proponujemy przyjąć zbiór $V^o(\pi)$ pomniejszony o te ruchy; wyniki testów numerycznych w pełni potwierdzają słuszność takiego postępowania.

Dla dalszych rozważań wygodnie jest zbiór $V(\pi)$ przedstawić w następującej postaci: $V(\pi) = \bigcup_{j \in K} V_j(\pi)$, gdzie $V_j(\pi)$ jest zbiorem ruchów z $V(\pi)$ polegającym na przesuwaniu operacji j, zaś K – zbiorem operacji ze wszystkich co najmniej dwuelementowych bloków. Zachodzi $|V_j(\pi)| \approx \sum_{l \in \mathcal{M}_{\mu(j)}} n_l = n s_{\mu(j)}$. Ponieważ sprawdzenie czy dla danego ruchu v graf $G(\pi_v)$ jest acykliczny i w przypadku odpowiedzi pozytywnej wyliczenie wartości $C_{\max}(\pi_v)$ wymaga $O(n)$ czasu, to w celu przeglądnięcia całego zbioru $V_j(\pi)$ potrzeba $O(n \cdot n s_{\mu(j)})$ czasu. W następnej sekcji przedstawimy efektywną metodę, która pozwala wykonać te czynności w czasie $O(n)$, co oznacza bardzo istotną redukcję czasu obliczeń.

4. Efektywna czasowo metoda przeglądania sąsiedztwa $\mathcal{N}(\pi)$

Proponowana metoda efektywnego przeglądnięcia sąsiedztwa $\mathcal{N}(\pi)$, a w szczególności wszystkich permutacji π_v, $v = (a, x, b, y) \in V_j(\pi)$ dla ustalonej operacji $j = \pi_a(x) \in K$ oraz permutacji $\pi \in \Pi$, wymaga zdefiniowania i wyznaczenia szeregu wielkości pomocniczych.

Przed ich zdefiniowaniem, w celu uproszczenia notacji, wprowadzimy skrócone oznaczenia na sześć operacji "związanych" z przesuwaną operacją j. Niech i' (j') oraz i'' (j'') oznaczają odpowiednio bezpośredniego kolejnościowego (technologicznego) poprzednika oraz następnika wierzchołka j w grafie $G(\pi)$. Podobnie, niech k' oraz k'' oznaczają odpowiednio bezpośredniego kolejnościowego poprzednika oraz następnika wierzchołka j w grafie $G(\pi_v)$. Dodatkowo, przez t' oraz t'' oznaczmy odpowiednio obciążenie łuku technologicznego dochodzącego oraz wychodzącego z wierzchołka j w grafie $G(\pi)$.

Definicje wielkości pomocniczych rozpoczynamy od określenia grafu G', który otrzymuje się z grafu $G(\pi)$: (*i*) po wyrzuceniu dwóch łuków kolejnościowych (i', j), (j, i'') ze zbioru $E^K(\pi)$, (*ii*) dodaniu łuku (i', i'') z wagą $s_a(i', i'')$ oraz (*iii*) obciążeniu wierzchołka j wagą zerową. Niech $r(i)$ oraz $q(i)$ oznacza odpowiednio wartość najdłuższej drogi dochodzącej do wierzchołka i oraz wychodzącej z wierzchołka i w grafie $G(\pi)$, łącznie z wagą tego wierzchołka; $i \in \mathcal{O}$. Identycznie definiujemy wartości $r'(i)$ oraz $q'(i)$ w grafie G'; dodatkowo przez C' oznaczmy wartość najdłuższej drogi w tym grafie. Dla każdej maszyny $b \in \mathcal{M}_{\mu(j)}$ definiujemy dwie pozycje c_b, d_b takie, że:

$$c_b = \max\{1 \leqslant i \leqslant n_b : \text{ istnieje ścieżka z } \pi_b(i) \text{ do } j' \text{ w grafie } G(\pi)\}, \tag{5}$$

$$d_b = \min\{1 \leqslant i \leqslant n_b : \text{ istnieje ścieżka z } j'' \text{ do } \pi_b(i) \text{ w grafie } G(\pi)\}. \tag{6}$$

Na bazie tych pozycji określamy dla operacji $j = \pi_a(x)$ zbiór ruchów W_j, zawierający ruchy (a, x, b, y) nie spełniające warunków (*iiia*)-(*iiic*) z własności 1 oraz takie, że $b \in \mathcal{M}_{\mu(j)}$ i $c_b + 1 \leqslant y < d_b$, jeżeli $b = a$ lub $c_b + 1 \leqslant y \leqslant d_b$ w przeciwnym wypadku. Ostatecznie dla ruchu $v = (a, x, b, y) \in W_j(\pi)$ definiujemy wielkość

$$F(v) = \max\{C', \hat{r}(j) + \hat{q}(j) - p_{b,j}\}, \tag{7}$$

gdzie

$$\hat{r}(j) = \max\{r(j') + t', r'(k') + s_b(k', j)\} + p_{b,j}, \tag{8}$$

$$\hat{q}(j) = \max\{q(j'') + t'', q'(k'') + s_b(j, k'')\} + p_{b,j}. \tag{9}$$

Prawdziwa jest następująca własność:

Własność 2. *Dla ustalonej permutacji $\pi \in \Pi$ oraz operacji $j = \pi_a(x) \in K$, znajdującej się na pozycji $1 \leqslant x \leqslant n_a$, na maszynie a zachodzi:*
(i) W_j jest zbiorem wszystkich ruchów $v \in V_j(\pi)$, dla których graf $G(\pi_v)$ jest acykliczny.
(ii) Dla każdego ruchu $v \in W_j$, $\hat{r}(j)$ oraz $\hat{q}(j)$ jest równe odpowiednio wartości najdłuższej drogi dochodzącej do wierzchołka j oraz wychodzącej z wierzchołka j w grafie $G(\pi_v)$, łącznie z wagą tego wierzchołka;
(iii) Dla każdego ruchu $v \in W_j$, $F(v) = C_{\max}(\pi_v)$.

Bazując na przedstawionej własności, efektywną czasowo metodę przeglądania sąsiedztwa generowanego przez zbiór ruchów $V_j(\pi)$ możemy przedstawić w postaci opisanej poniżej procedury PS(π, j). Procedura ta dla zadanej permutacji $\pi \in \Pi$ oraz operacji $j \in K$. wyznacza wśród wszystkich ruchów ze zbioru $V_j(\pi)$, dla których graf $G(\pi_v)$ jest acykliczny, ruch $w(j)$ generujący permutację $\pi_{w(j)}$ o najmniejszej wartości funkcji celu.

Procedura PS(π, j)

Krok 1. *Dla każdego wierzchołka $i \in \bigcup_{b \in \mathcal{M}_{\mu(j)}} \mathcal{O}_b$ wyznacz wartości najdłuższych dróg $r(i)$ oraz $q(i)$ w grafie $G(\pi)$.*

Krok 2. *Utwórz graf G'. Dla każdego wierzchołka $i \in \bigcup_{b \in \mathcal{M}_{\mu(j)}} \mathcal{O}_b$ wyznacz wartości najdłuższych dróg $r'(i)$, $q'(i)$ oraz wartość najdłuższej drogi C' w grafie G'.*

Krok 3. *Dla każdej maszyny $b \in \mathcal{M}_{\mu(j)}$ wyznacz dwie pozycje c_b, d_b zgodnie (5), (6). Następnie znajdź zbiór ruchów W_j.*

Krok 4. *Znajdź ruch $w(j)$ taki, że $w(j) \in W_j$ oraz $F(w(j)) = \min_{v \in W_j} F(v)$, gdzie $F(v)$ jest określone przez (7).*

Zauważmy, że każdy z kroków 1, 2 oraz 3 powyżej procedury można wykonać w czasie $O(n)$. Krok 4 wymaga $O(ns_{\mu(j)})$ czasu, ponieważ $W_j \subset V_j(\pi)$, $|V_j(\pi)| \approx ns_{\mu(j)}$ oraz do wyznaczenia wartości $F(v)$ dla jednego ruchu v potrzeba $O(1)$ czasu. Stąd cała procedura jest wykonywana w czasie $O(n)$.

Wszystkie pozostałe elementy proponowanego tutaj algorytmu "popraw" – zwanego dalej algorytmem TSA – takie jak: postać listy Tabu T, definicja ruchów zabronionych, strategia przeglądania otoczenia $\mathcal{N}(\pi)$ oparta na pojęciu reprezentantów oraz metody intensyfikacji i dywersyfikacji poszukiwań są podobne do tych, które naszkicowano w pracy [5] i dlatego nie będą przedstawiane. Dokładne ich omówienie wraz z odpowiednimi własnościami teoretycznymi można znaleźć w monografii [7].

5. Wyniki badań testowych

Algorytm TSA zakodowano w Delphi 3 i uruchamiano na Pentium II (330MHz). Aktualnie w literaturze dla badanego problemu brak jest odpowiednich przykładów testowych. Dlatego też zostały one wygenerowane z 80 szczególnie trudnych bazowych przykładów podanych w [10] dla klasycznego problemu gniazdowego. Przykłady te są podzielone na 8 grup, po dziesięć w każdej grupie, oznaczanych przez: 225/15/15, 300/20/15, 400/20/20, 450/30/15, 600/30/20, 750/50/15, 1000/50/20 oraz 2000/100/20 (pierwszy element oznacza liczbę operacji, drugi liczbę zadań, a trzeci liczbę jednomaszynowych stanowisk); każde zadanie składa się z tylu operacji, ile jest stanowisk. Procedura generacji była następująca: Dla każdego przykładu bazowego wylosowano: (*i*) liczbę maszyn w poszczególnych stanowiskach $m_l = 1 \div 5$, (*ii*) czasy wykonywania $p_{l,j}$, w przypadku gdy $m_l > 1$ (przez niewielkie zaburzenie czasów pierwotnych, które wahały się od 1 do 100), (*iii*) czasy transportu $t_{k',k''} = 1 \div 10$ oraz (*iv*) czasy przezbrojeń $s_l(j', j'') = 0 \div 20$. Jeden przykład bazowy dostarczał 20 przykładów; łącznie otrzymano 1600 przykładów testowych.

Brak w literaturze odpowiednich algorytmów typu popraw dla ogólnej postaci badanego problemu spowodował, że algorytm TSA porównywano z wieloma różnymi algorytmami konstrukcyjnymi. Głównym celem badań było określenie, o ile lepsze rozwiązania generuje TSA i ile potrzeba czasu, aby je otrzymać? Do badań porównawczych wykorzystano 9 algorytmów konstrukcyjnych różniących się regułami priorytetowymi (w tym algorytm konstrukcyjny z [3] i z [1], po odpowiednich dostosowaniach) oraz jeden algorytm bazujący na metodzie wstawień, będący modyfikacją algorytmu NEH dla problemu przepływowego. Dla każdego przykładu testowego znajdowano rozwiązanie bazowe (najlepsze rozwiązanie otrzymane z 10 rozwiązań wygenerowanych przez ww. algorytmy) oraz rozwiązanie wyprodukowane przez algorytm TSA po 5000 iteracji.

Ograniczone ramy pracy pozwalają tylko na stwierdzenie, że algorytm TSA poprawia rozwiązanie bazowe w sensie wartości funkcji celu (startując z rozwiązania dostarczonego przez jeden algorytm konstrukcyjny) o około $9 \div 18\%$ w czasie rzędu od kilku sekund do kilku minut dla przykładów o największych rozmiarach (2000 operacji, 20 stanowisk, od 1 do 5 maszyn w stanowisku).

LITERATURA

1. Dauzère-Pérès S., Pauli J.: An integrated approach for modeling and solving the general multiprocessor job-set scheduling problem using Tabu search. Annals of Operations Research 70, 1997, pp. 281-306.
2. Glover F., Laguna M.: Tabu Search. Kluwer Academic Publishers, Massachusetts USA, 1997.
3. Hurink J., Jurisch J., Thole M.: Tabu search for the job-shop scheduling problem with multi-purpose machines. OR Spektrum 15, 1994, pp. 205-215.
4. Nowicki E.: Zastosowanie techniki Tabu search do harmonogramowania elastycznych gniazd produkcyjnych. Zeszyty Naukowe Politechniki Śląsiej, s. Automatyka, z. 118, Gliwice 1996, str. 153-154.
5. Nowicki E.: Problem przepływowy z maszynami równoległymi i przezbrojeniami. Algorytm Tabu search. Zeszyty Naukowe Politechniki Śląskiej, s. Automatyka, z. 125, Gliwice 1998, str. 67-78.
6. Nowicki E.: The permutation flow shop with buffers. A local search approach. European Journal of Operational Research 116, 1999, pp. 205-219.
7. Nowicki E.: Metoda Tabu w problemach szeregowania zadań produkcyjnych. Oficyna Wydawnicza Politechniki Wrocławskiej, Ser. Monografie 27, Wrocław 1999.
8. Nowicki E., Smutnicki C.: The flow shop with paralled machines. A Tabu search approach. European Journal of Operational Research 106, 1998, pp. 226-253.
9. Sawik T.: Planowanie i sterowanie produkcji w elastycznych systemach montażowych. WNT, Warszawa 1996.
10. Taillard E.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 1993, pp. 278-285.

Recenzent: Prof. dr hab. inż. K. Wala

Abstract

The paper deals with the criterion of the makespan minimisation for the job shop problem with parallel machines, transport times and setup times. A fast and easily implemented approximation algorithm is proposed based on a non trivial generalisation of the path elimination properties known for the classic job shop problem. This algorithm is able to achieve excellent results for instances up to 2000 operations, 100 jobs and 20 machine centers due to exploiting of some structural properties of the problem combined with a local search technique controlled by a Tabu search strategy. A special advanced method of implementation improves the local search significantly and increases the speed of the algorithm.