ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 125 Nr kol. 1391

Eugeniusz NOWICKI
Instytut Cybernetyki Technicznej Politechniki Wrocławskiej

PROBLEM PRZEPŁYWOWY Z MASZYNAMI RÓWNOLEGŁYMI I PRZEZBROJENIAMI. ALGORYTM TABU SEARCH

Streszczenie. W pracy przedstawia się szybki aproksymacyjny algorytm oparty na technice tabu dla problemu przepływowego z maszynami równoległymi i przezbrojeniami z kryterium minimalizującym moment wykonania wszystkich zadań. Specjalna metoda implementacji istotnie zwiększa prędkość algorytmu.

THE FLOW SHOP WITH PARALLEL MACHINES AND SETUP TIMES. A TABU SEARCH ALGORITHM

Summary. A fast approximation algorithm for a problem of finding the minimum makespan in a flow shop with parallel machines and setup times is presented. The algorithm is based on a tabu search technique. A special method of implementation increases the speed of the algorithm significantly.

1. Wprowadzenie

W niniejszej pracy bada się problem przepływowy z równoległymi maszynami w poszczególnych stanowiskach obsługi, rozszerzony o dwa dodatkowe ograniczenia, wynikające z praktyki (patrz np. [9], [8]): (*i*) - niezerowe czasy przezbrojeń oraz (*ii*) - ograniczona liczba zadań przebywających w każdej chwili w systemie produkcyjnym. Kolejność przechodzenia zadań przez stanowiska jest zadana i *identyczna* dla wszystkich zadań; niektóre stanowiska - z wyjątkiem pierwszego i ostatniego - mogą być omijane. W stanowisku pierwszym (stanowisko załadowcze) tzw. część bazowa danego zadania (wyrobu) łacznie z dodatkowymi elementami (niezbędnymi w procesie produkcyjnym) jest montowana na paletę. Paleta jest transportowana pomiędzy kolejnymi stanowiskami aż do stanowiska ostatniego (stanowisko wyładowcze), w którym następuje demontaż zadania z palety; czasy transportu są określone. Zwolniona paleta może być natychmiast użyta do montowania na niej części bazowej kolejnego zadania. W przypadku gdy maszyna z danego stanowiska, na której ma być wykonywane zadanie umieszczone na palecie, jest zajęta lub jest

przezbrajana, paleta oczekuje przed tym stanowiskiem, aż maszyna stanie się gotowa. Ograniczone możliwości sytemu transportowego implikują, że tylko ograniczona liczba palet (i w konsekwencji zadań) może przebywać jednocześnie w systemie. Należy minimalizować moment czasowy, w którym wszystkie zadania zostały wykonane i zdemontowane z palet.

Praktyczne znaczenie powyższego problemu (np. przy sterowaniu elastycznymi liniami produkcyjnymi) oraz jego oczywista NP-trudność powodują, iż do rozwiązania proponuje się zwykle algorytmy heurystyczne typu konstrukcyjnego. Ich krótki czas pracy nie zawsze jednak rekompensuje nie najlepszą jakość produkowanych rozwiązań. Dlatego też aktualnym kierunkiem badań jest konstruowanie algorytmów typu popraw, a w szczególności algorytmów opartych na technice tabu search, które dostarczają znacznie lepszych rozwiązań w czasie akceptowalnym dla praktyków. Algorytmy o takich własnościach dla wielu podobnych problemów (klasyczny problem przepływowy z ograniczeniem liczby zadań przebywających w systemie lub ograniczoną pojemnością buforów, problem przepływowy z maszynami równoległymi bez/i z czasami transportu itp.) przedstawiono np. w [3]–[7].

Proponowany tutaj algorytm stanowi istotne uogólnienie algorytmów prezentowanych w powyższych pracach. Odpowiedni model grafowy umożliwił wyspecyfikowanie szeregu własności, na bazie których zaproponowano sąsiedztwo danego uszeregowania. Wykorzystanie oryginalnych metod szybkiego wyznaczania wartości funkcji celu pozwoliło znacznie zredukować czasochłonność wyboru uszeregowania z sąsiedztwa. Przeprowadzone testy komputerowe pokazują, że algorytm istotnie poprawia najlepsze uszeregowanie - wybrane wśród uszeregowań otrzymanych przez zastosowanie kilkunastu algorytmów konstrukcyjnych dla danego przykładu - w czasie rzędu kilku minut na IBM PC.

2. Sformułowanie problemu

W rozważanym systemie produkcyjnym znajduje się zbiór $\mathcal{M} = \{1, \ldots, m\}$ maszyn podzielonych na ls podzbiorów $\mathcal{M}_k = \{\omega_k + 1, \omega_k + 2, \ldots, \omega_k + m_k\}$, $k = 1, \ldots ls$, utożsamianych ze stanowiskami oraz lp palet, gdzie $\omega_k = \sum_{h=1}^{k-1} m_h$, $\omega_1 = 0$ oraz $\sum_{k=1}^{ls} m_k = m$. Stanowisko 1 oraz stanowisko ls są odpowiednio: stanowiskiem załadowczym oraz stanowiskiem wyładowczym. Przyjmujemy, że $m_1 = m_m = 1$, tzn. $\mathcal{M}_1 = \{1\}$ oraz $\mathcal{M}_{ls} = \{m\}$.

W systemie należy zrealizować r zadań ze zbioru $\mathcal{J} = \{1, 2, \ldots, r\}$. Zadanie i, $i \in \mathcal{J}$, składa się $o_i \geq 3$ operacji indeskowanych 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 $1 \leq \mu(j) \leq ls$; operacja j może być wykonana na jednej dowolnie wybranej maszynie $l \in \mathcal{M}_{\mu(j)}$ z tego stanowiska w czasie $p_{l,j} \geq 0$. Zakładamy, że $1 = \mu(\tau_i + 1) < \mu(\tau_i + 2) < \ldots < \mu(\tau_i + o_i) = ls$ dla każdego zadania $i \in \mathcal{J}$, tzn. kolejne operacje zadania i są wykonywane na stanowiskch, które stanowią pewien podciąg ciągu $1, 2, \ldots, ls$ oraz pierwsza operacja tego zadania jest wykonywana na stanowisku 1, a ostatnia na stanowisku ls. Pomiędzy wykonywaniem dwóch kolejnych operacji j', j'' na maszynie $l \in \mathcal{M}$ wymagany jest czas przezbrojenia $s_l(j', j'') \geq 0$; jeżeli operacja j jest wykonywana jako pierwsza na maszynie l, to czas przezbrojenia maszyny l jest równy $s_l(0, j) \geq 0$. Przyjmujemy, że na maszynie 1 oraz m czas przezbrojenia jest równy zero. Dodatkowo zadany jest czas transportu $t_{k',k''} \geq 0$ pomiędzy każdą parą stanowisk $1 \leq k' < k'' \leq ls$.

Na działanie omawianego systemu produkcyjnego nakładamy następujące typowe ograniczenia: (i) każda maszyna może wykonywać w danej chwili nie więcej niż jedną operację, (ii) nie można jednocześnie wykonywać więcej niż jednej operacji danego zadania w danej chwili, (iii) wykonywanie operacji na maszynie nie może być przerywane oraz dodatkowo (iv) w systemie nie może się znajdować więcej niż lp zadań. 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 na 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 $\max_{j \in \mathcal{O}} (S(j) + p_{m(j),j})$.

Dalej będziemy zakładać, że czasy przezbrojeń spełniają warunek "trójkąta", tzn. dla każdego $j' \in \mathcal{O} \cup \{0\}$, $j, j'' \in \mathcal{O}$ oraz $l \in \mathcal{M}$ zachodzi $s_l(j', j'') \leq s_l(j', j) + s_l(j, j'')$. Warunek trójkąta jest spełniony w większości sytuacji praktycznych. W szczególności jest on prawdziwy dla następujących dwóch - szczególnie często występujących w praktyce - przypadków. Przypadek pierwszy dotyczy *sekwencyjnie niezależnych* czasów przezbrojeń, tzn. czasów przezbrojeń, dla których czas przezbrojenia pomiędzy parą sąsiednich operacji j', j'' na maszynie l zależy tylko od operacji j'' i maszyny l. Przypadek drugi odnosi się do sytuacji *rodzinowo-sekwencyjnie niezależnych* czasów przezbrojeń, w której: (i) operacje (zadania) są podzielone na pewną liczbę rozłącznych podzbiorów (rodzin), (ii) niezerowy

czas przezbrojenia występuje tylko pomiędzy kolejno wykonywanymi operacjami j', j'' z dwóch różnych rodzin i zależy od maszyny oraz rodziny, do której należy operacja j''.

3. Model grafowy i bloki operacji

Jednym z podstawowych elementów przy konstrukcji algorytmów dla problemów harmonogramowania zadań produkcyjnych jest odpowiedni wybór modelu matematycznego, a w szczególności zmiennych decyzyjnych. Prezentowany tutaj model jest uogólnieniem modeli zapronowanych w [3]–[7], wykorzystujących pewien skierowany graf w konwencji "wierzchołek - operacja" i przyjmujący jako zmienną niezależną kolejność wykonywania operacji na poszczególnych maszynach. Model ten w odróżnieniu od innych modeli, np. bazujących na grafach dysjunktywnych, jest znacznie prostszy, dostarczając jednocześnie pewnych własności mających zastosowanie w projektowanych algorytmach.

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 operacje, które zostały wybrane do wykonywania na maszynie l. Oznaczmy przez n_l liczność tego zbioru. Wtedy $\sum_{l \in \mathcal{M}} n_l = n$. Z definicji wynika, że $\mathcal{O}_1 = \bigcup_{i=1}^{r} \{\tau_i + 1\}$ oraz $\mathcal{O}_m = \bigcup_{i=1}^{r} \{\tau_i + o_i\}$, tzn. zbiory $\mathcal{O}_1$, $\mathcal{O}_m$ nie zależą od podziału i zawierają zawsze wszystkie pierwsze operacje zadań oraz wszystkie ostatnie operacje zadań. 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 - określa kolejność wykonywania wszystkich operacji ze zbioru $\mathcal{O}$ na poszczególnych maszynach i jest traktowany jako zmienna decyzyjna. Ostatecznie przez Π oznaczmy zbiór permutacji π wynikających z wszystkich możliwych podziałów zbioru $\mathcal{O}$ oraz różnych kolejności wykonywania operacji na poszczególnych maszynach. Zauważmy, że przy takim sformułowaniu zmiennej decyzyjnej nie ma potrzeby rozgraniczania pomiędzy podziałem zbioru $\mathcal{O}$ na poszczególne podzbiory $\mathcal{O}_l$ a kolejnościami π_l. Podanie permutacji $\pi \in \Pi$ definiuje oba te elementy; w szczególności dla danej permutacji $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$ podział zbioru $\mathcal{O}$ ma postać: $\mathcal{O}_l = \{\pi_l(1), \ldots, \pi_l(n_l)\}$, $l \in \mathcal{M}$.

Dla permutacji π, $\pi \in \Pi$ niech $G(\pi) = (\mathcal{O} \cup \{0\}, E^T \cup E^K(\pi) \cup E^P(\pi))$ będzie skierowanym grafem ze zbiorem wierzchołków $\mathcal{O} \cup \{0\}$ oraz zbiorem łuków $E^T \cup E^K(\pi) \cup E^P(\pi)$. Wierzchołki $\mathcal{O}$ reprezentują operacje a łuki ograniczenia; wierzchołek 0 odpowiada pewnej operacji fikcyjnej. Zbiór $E^T = \bigcup_{k \in \mathcal{J}} \bigcup_{j=\tau_k+2}^{\tau_k+o_k} \{(j-1, j)\}$ określa wymaganą

kolejność wynikającą z marszruty technologicznej poszczególnych zadań, $E^K(\pi) = \bigcup_{l\in\mathcal{M}} \bigcup_{i=1}^{n_l}\{(\pi_l(i-1), \pi_l(i))\}$ - wybraną kolejność wykonywania operacji na maszynach ($\pi_l(0) = 0$), zaś $E^P(\pi) = \bigcup_{i=lp+1}^{n_l}\{(\pi_m(i-lp), \pi_1(i))\}$ modeluje ograniczoną liczbę zadań (palet) w systemie; montowanie zadania, odpowiadającego operacji $\pi_1(i)$, na paletę w stanowisku 1 może się rozpocząć dopiero po zdemontowaniu z palety w stanowisku ls zadania odpowiadającego operacji $\pi_m(i-lp)$. Każdy wierzchołek $j = \pi_l(i)$, $1 \le i \le n_l$, $l \in \mathcal{M}$ 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 $t_{\mu(j-1),\mu(j)}$; łuki ze zbioru $E^P(\pi)$ nie są obciążone. Zauważmy, że łuki $E^P(\pi)$ mogą powodować cykl w grafie $G(\pi)$. W przypadku gdy graf ten jest acykliczny, przez $r^\pi(j)$ ($q^\pi(j)$) oznaczmy wartość najdłuższej ścieżki dochodzącej do wierzchołka j (wychodzącej z wierzchołka j), łącznie z tym wierzchołkiem. Wtedy dla każdego wierzchołka $j = \pi_l(i)$ leżącego na najdłuższej ścieżce (ścieżka krytyczna) zachodzi $r^\pi(j) + q^\pi(j) - p_{l,j} = C_{\max}(\pi)$, gdzie $C_{\max}(\pi)$ jest długością tej ścieżki.

Przedyskutujemy teraz związek pomiędzy wprowadzonym grafem a uszeregowaniem dopuszczalnym. Niech $U(\pi)$ oznacza zbiór wszystkich uszeregowań dopuszczalnych i zgodnych z permutacją $\pi \in \Pi$, tzn. takich, że na maszynie l wykonywane są kolejno operacje $\pi_l(1), \ldots, \pi_l(n_l)$, $l \in \mathcal{M}$. Prawdziwa jest następująca własność, którą, podobnie jak następne, podajemy bez dowodu.

Własność 1. *Dla każdej permutacji $\pi \in \Pi$ zachodzi:*
(i) Zbiór $U(\pi)$ nie jest pusty wtedy i tylko wtedy, gdy graf $G(\pi)$ jest acykliczny.
(ii) Jeżeli graf $G(\pi)$ jest acykliczny, to uszeregowanie $(m^\pi(j), S^\pi(j))$, $j \in \mathcal{O}$ takie, że $m^\pi(j) = l$, $j \in \{\pi_l(1), \ldots, \pi_l(n_l)\}$, $l \in \mathcal{M}$ oraz $S^\pi(j) = r^\pi(j) - p_{m^\pi(j),j}$, $j \in \mathcal{O}$, jest uszeregowaniem dopuszczalnym o najmniejszej wartości funkcji celu wśród uszeregowań ze zbioru $U(\pi)$; wartość ta jest równa $C_{\max}(\pi)$.

Kolejna własność podaje proste do sprawdzenia warunki konieczne i wystarczające na acykliczność grafu $G(\pi)$.

Własność 2. *Dla permutacji $\pi \in \Pi$ graf $G(\pi)$ jest acykliczny wtedy i tylko wtedy, gdy dla każdego zadania $i \in \mathcal{J}$*

$$\pi_1^{-1}(\tau_i + 1) - \pi_m^{-1}(\tau_i + o_i) < lp, \tag{1}$$

gdzie π_1^{-1}, π_m^{-1} są permutacjami odwrotnymi do permutacji π_1, π_m.

Wykorzystując powyższe własności, badany problem formułujemy jako znalezienie takiej permutacji $\pi^* \in \Pi$, która spełnia (1) oraz odpowiadający jej graf $G(\pi^*)$ ma najmniejszą długość ścieżki krytycznej.

Ścieżkę krytyczną w grafie $G(\pi)$ można opisać jako ciąg $u = (u_1, u_2, \ldots, u_{lu})$ wierzchołków tego grafu, gdzie $u_i \in \mathcal{O}$, $1 \leq i \leq lu$, a lu jest liczbą wierzchołków ścieżki. Ścieżkę u przedstawiamy w postaci $u = (B_1, B_2, \ldots, B_{lb})$, gdzie $B_h = (\pi_{l_h}(e_h), \pi_{l_h}(e_h + 1), \ldots, \pi_{l_h}(f_h))$, $1 \leq h \leq lb$ jest ciągiem zawierającym operacje wykonywane na tej samej maszynie (oznaczamy ją przez l_h), poczynając od operacji z pozycji e_h, $1 \leq e_h \leq n_{l_h}$, a kończąc na operacji z pozycji f_h, $e_h \leq f_h \leq n_{l_h}$, w permutacji π_{l_h}. Dodatkowo przyjmujemy, że operacje z dwóch kolejnych ciągów B_{h-1}, B_h są wykonywane na różnych maszynach, tzn. $l_{h-1} \neq l_h$, $h = 2, \ldots, lb$. Dalej ciąg B_h będziemy nazywać h-tym *blokiem* operacji na ścieżce krytycznej. Ogólnie liczba elementów bloku B_h jest istotnie mniejsza niż n_{l_h}; w skrajnym przypadku blok B_h może zawierać tylko jeden element.

4. Ruch i sąsiedztwo

Konstrukcja algorytmu TS, bazującego na metodzie tabu search, wymaga sprecyzowania szeregu elementów, do których należą: ruch, sąsiedztwo, lista tabu, strategia przeglądania sąsiedztwa, poziom aspiracji, pamięć długoterminowa, rozwiązanie początkowe oraz warunki stopu [2]. Dalej omówimy te elementy rozpoczynając od definicji ruchu.

Dla $\pi \in \Pi$ będziemy rozważać ruchy $v = (a, x, b, y)$, polegające na pobraniu operacji $j = \pi_a(x)$ z pozycji $1 \leq x \leq n_a$ w permutacji π_a na maszynie $a \in \mathcal{M}_{\mu(j)}$ i włożeniu jej w permutację π_b z maszyny $b \in \mathcal{M}_{\mu(j)}$ na pozycję $1 \leq y \leq n_b + 1$ (lub na pozycję $1 \leq y \leq n_a$, $y \neq x$, jeżeli $b = a$); w trakcie wykonywania ruchu odpowiednie operacje są przesuwane o jedną pozycję w lewo lub w prawo [3], [4]. Otrzymaną w wyniku ruchu v nową permutację oznaczamy przez π_v.

Szczególnie istotnym elementem metody tabu search jest określenie zbioru wykonywanych ruchów $V(\pi)$ z danej permutacji $\pi \in \Pi$ i w konsekwencji jej sąsiedztwa $\mathcal{N}(\pi) = \{\pi_v : v \in V(\pi)\}$. Podstawowa idea konstrukcji $V(\pi)$ będzie polegała na eliminowaniu ze zbioru $V^o(\pi)$ (zawierającego wszystkie możliwe ruchy dla danej premutacji π) ruchów v, o których "możemy stwierdzić" bez wyliczania wartości $C_{\max}(\pi_v)$, że

$$C_{\max}(\pi_v) \geq C_{\max}(\pi). \tag{2}$$

Główną rolę pełnią tutaj tzw. eliminacyjne własności ścieżki krytycznej, zaprezentowane w poniższych dwóch własnościach dotyczących odpowiednio ruchów na maszynach $\mathcal{M} \setminus \{1, m\}$ oraz $\{1, m\}$. W pierwszej z nich dla ruchu $v = (l_h, x, l_h, y)$, $x \neq y$, $e_h < x < f_h$, $e_h < y < f_h$, polegającego na przesuwaniu operacji $j = \pi_{l_h}(x)$ wewnątrz pewnego bloku B_h, wykorzystujemy wielkość $\Delta^\pi(v) = -\Delta_1^\pi(v) + \Delta_2^\pi(v)$, gdzie

$$\Delta_1^\pi(v) = s_{l_h}(j1, j) + s_{l_h}(j, j2) - s_{l_h}(j1, j2), \quad \Delta_2^\pi(v) = s_{l_h}(j3, j) + s_{l_h}(j, j4) - s_{l_h}(j3, j4) \quad (3)$$

Przez $j1$ oraz $j2$ ($j3$, $j4$) oznaczyliśmy odpowiednio bezpośredniego poprzednika oraz bezpośredniego następnika operacji j z $E^K(\pi)$ w grafie $G(\pi)$ (w grafie $G(\pi_v)$). Wielkość $\Delta^\pi(v)$ 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. Z nierówności trójkąta wynika, że $\Delta_1^\pi(v) \geq 0$ oraz $\Delta_2^\pi(v) \geq 0$. Co więcej, jeżeli czasy przezbrojeń są sekwencyjnie niezależne, to $\Delta^\pi(v) = 0$.

Własność 3. *Niech $v = (a, x, b, y) \in V^o(\pi)$, $a \notin \{1, m\}$ oraz $j = \pi_a(x)$. Jeżeli zachodzi jeden z poniższych dwóch warunków:*
(i) operacja j nie należy do ścieżki krytycznej u,
(ii) operacja j należy do pewnego bloku B_h, $b = a = l_h$ oraz (I) $x = e_h$, $1 \leq y < e_h$ lub (II) $e_h < x < f_h$, $e_h < y < f_h$, $\Delta^\pi(v) \geq 0$ lub (III) $x = f_h$, $f_h < y \leq n_{l_h}$,
to prawdziwa jest nierówność (2).

Warunek (*ii*) z Własności 3 opisuje następujące sytuacje: (*I*) przesunięcie pierwszej operacji bloku B_h w lewo, (*II*) przesuwanie operacji z wnętrza bloku B_h na pozycje z wnętrza tego bloku, dla których nie maleją sumaryczne czasy przezbrojeń oraz (*III*) przesunięcie ostatniej operacji bloku B_h w prawo. Warto tu zauważyć, że w przypadku, gdy blok B_h ma tylko jeden element ($B_h = \{j\}$), to dla każdego ruchu przesuwającego operację j w obrębie maszyny l_h nierówność (2) jest prawdziwa. Nie musi zachodzić to jednak, jeżeli przesuwamy operację j na inną maszynę ze zbioru $\mathcal{M}_{\mu(j)}$.

Własność 4. *Niech $v = (a, x, a, y) \in V^o(\pi)$, $a \in \{1, m\}$ oraz $j = \pi_a(x)$. Jeżeli zachodzi warunek* (1) *dla permutacji π_v oraz jeden z trzech poniższych warunków:*
(i) operacja j oraz żadna z operacji $\pi_a(i)$, $\min(x, y) < i < \max(x, y)$ nie należy do ścieżki krytycznej u,
(ii) operacja j należy do pewnego bloku B_h, $a = l_h = 1$ oraz (I) $e_h \leq x < f_h$, $e_h \leq y < f_h$ lub (II) $x = f_h$, $f_h < y \leq n_{l_h}$,

(iii) operacja j należy do pewnego bloku B_h, $a = l_h = m$ oraz (I) $x = e_h$, $1 \leq y < e_h$ lub (II) $e_h < x \leq f_h$, $e_h < y \leq f_h$,
to prawdziwa jest nierówność (2).

Warunek (i) z własności 4 dotyczy ruchów polegających na przesuwaniu niekrytycznej operacji j z pozycji x na taką pozycję y, żeby w trakcie ruchu nie "przeskakiwać" żadnej operacji krytycznej. Z kolei warunek (ii) dotyczy pewnego bloku B_h z maszyny 1 i opisuje sytuacje: (I) przesuwanie operacji bloku B_h bez operacji ostatniej na wszystkie jego pozycje z wyjątkiem pozycji ostatniej, (II) przesunięcie ostatniej operacji bloku B_h w prawo. Warunek (iii) jest symetryczny do warunku (ii).

Biorąc powyższe pod uwagę, jako zbiór ruchów, które należy wykonać z permutacji π, proponujemy przyjąć $V(\pi) = V^1(\pi) \cup V^2(\pi)$, gdzie $V^1(\pi)$ jest zbiorem ruchów $v \in V^o$, które nie spełniają założeń własności 3 a $V^2(\pi)$ - zbiorem ruchów $v \in V^o$, które nie spełniają założeń własności 4 oraz zachodzi warunek (1) dla permutacji π_v. Z takiego sposobu postępowania nie należy oczywiście wnioskować, że dla wszystkich pozostałych ruchów z $V^o(\pi)$ nierówność (2) nie zachodzi. Przykładowo, na bazie wielkości $r^\pi(j)$, $q^\pi(j)$ można zbudować inne warunki, które implikują (2) dla danego ruchu v. Badania numeryczne pokazały jednak, że dalsza eliminacja tego typu ruchów zmniejsza skuteczność techniki tabu search. Co więcej, badania te pokazały także, że mimo iż liczba ruchów określonych przez własność 3(ii) w stosunku do ruchów określonych przez własność 3(i) nie jest stosunkowo duża, to ich eliminacja jest istotna dla algorytmu. Wykonywanie tych ruchów może powodować, że algorytm ma tendencję do szczegółowej penetracji tylko jednego obszaru rozwiązań, co z reguły nie jest korzystną cechą.

5. Efektywne wyznaczanie najlepszego ruchu

Przedstawimy teraz metodę "szybkiego" wyznaczania $C_{\max}(\pi_v)$, $v \in V(\pi)$, która jest istotnym uogólnieniem odpowiedniej procedury z [7]. W tym celu zauważmy, że $V(\pi)$ można przedstawić w postaci $V(\pi) = \bigcup_{j \in A_1 \cup A_2} V_j(\pi)$, gdzie $V_j(\pi)$ jest zbiorem ruchów z $V(\pi)$, polegającym na przesuwaniu operacji j, A_1 - zbiorem operacji z bloków B_h na maszynach $\mathcal{M} \setminus \{1, m\}$, zaś $A_2 = \mathcal{O}_1 \cup \mathcal{O}_m$. Zachodzi $|V_j(\pi)| \approx \sum_{l \in \mathcal{M}_{\mu(j)}} n_l$. Ponieważ $\sum_{l \in \mathcal{M}_{\mu(j)}} n_l \approx r$ oraz wyznaczenie $C_{\max}(\pi_v)$ dla danego ruchu v wymaga $O(n)$ czasu, to dla przeglądnięcia całego zbioru $V_j(\pi)$ potrzeba aż $O(nr)$ czasu. Prezentowana metoda pozwala wykonać to w $O(n)$. Praktycznie oznacza to r-krotną redukcję czasu obliczeń, co

przy liczbie zadań np. $r = 100$ jest istotną wielkością.

Metodę omówimy tylko dla operacji ze zbioru A_1. Główna jej idea polega na spostrzeżeniu, że ruch $v = (a, x, b, y) \in V_j(\pi)$, przesuwający operację $j = \pi_a(x)$, można podzielić na dwie fazy: (i) pobranie operacji j z pozycji x w permutacji π_a oraz (ii) włożenie operacji j na pozycję y w permutacji π_b. Czynności w fazie pierwszej nie zależą od pozycji i maszyny (permutacji), na którą operacja j jest wkładana. Stąd najpierw konstruujemy pewien graf pomocniczy $\hat{G}(\pi)$ z grafu $G(\pi)$, "odpowiadający" sytuacji po wykonaniu fazy pierwszej ruchu v. Następnie wyliczamy w nim wartość najdłuższej ścieżki dochodzącej i wychodzącej z wierzchołka t, $t \in \mathcal{O}$; wartości te odpowiadają wielkościom $r^\pi(t)$, $q^\pi(t)$ zdefiniowanym w rodziale 3 dla grafu $G(\pi)$ i są oznaczane dalej przez $\hat{r}(t)$, $\hat{q}(t)$. Bazując na nich, wyznaczamy $C_{\max}(\pi_v)$ dla każdego ruchu $v \in V_j(\pi)$. Graf pomocniczy otrzymywany jest z grafu $G(\pi)$ przez wykonanie następujących czynności: (*i*) usunięcie łuku $(j1, j) \in E^K(\pi)$ i łuku $(j, j2) \in E^K(\pi)$, (*ii*) dodanie łuku $(j1, j2)$ o wadze $s_a(j1, j2)$, (*iii*) zmianę wagi węzła j z $p_{a,j}$ na $\min_{l \in \mathcal{M}_{\mu(j)}} p_{l,j}$ oraz (*iv*) wyzerowanie wagi łuku $(j-1, j) \in E^T$ oraz wyzerowanie wagi łuku $(j, j+1) \in E^T$. Jeżeli operacja j nie ma poprzednika lub następnika kolejnościowego $j1$ lub $j2$ w grafie $G(\pi)$ lub $G(\pi_v)$, to odpowiednia czynność nie jest wykonywana. Zachodzi następująca własność:

Własność 5. *Niech $\pi \in \Pi$ oraz $j \in A_1$. Dla każdego ruchu $v = (a, x, b, y) \in V_j(\pi)$,*

$$C_{\max}(\pi_v) = \max\{\hat{C},\ r^{\pi_v}(j) + q^{\pi_v}(j) - p_{b,j}\}, \tag{4}$$

gdzie $\hat{C}$ jest długością ścieżki krytycznej w pomocniczym grafie $\hat{G}(\pi)$ oraz

$$r^{\pi_v}(j) = \max\{r^\pi(j-1) + t_{\mu(j-1),\mu(j)},\ \hat{r}(j3) + s_b(j3, j)\} + p_{b,j} \tag{5}$$

$$q^{\pi_v}(j) = \max\{q^\pi(j+1) + t_{\mu(j),\mu(j+1)},\ \hat{q}(j4) + s_b(j, j4)\} + p_{b,j} \tag{6}$$

Wykorzystując powyższą własność można zaproponować efektywną procedurę wyznaczania $C_{\max}(\pi_v)$ dla wszystkich $v \in V_j(\pi)$. W pierwszym kroku należy utworzyć graf $\hat{G}(\pi)$, wyliczyć $\hat{C}$ oraz wyznaczyć $\hat{r}(t)$, $\hat{q}(t)$ dla wszystkich operacji t z permutacji π_l, $l \in \mathcal{M}_{\mu(j)}$; można to zrobić w czasie $O(n)$. W kroku drugim dla każdego $v = (a, x, b, y) \in V_j(\pi)$ wyznaczamy $C_{\max}(\pi_v)$ z (4) - w czasie $O(1)$ - korzystając z (5) oraz (6). Ponieważ $|V_j(\pi)| \approx r$, to krok ten potrzebuje $O(r)$ czasu. Ostatecznie cała procedura wymaga $O(n)$ czasu. Wielkości $r^\pi(t)$, $q^\pi(t)$ wyliczamy w czasie $O(n)$ w trakcie wyznaczania ścieżki krytycznej u.

6. Pozostałe elementy algorytmu tabu search

Pozostałe elementy algorytmu TS są podobne jak w algorytmie z [4] dla problemu gniazdowego z maszynami równoległymi i dlatego zostaną omówione skrótowo.

Elementami listy tabu T o ustalonej długości $MaxT$ (parametr algorytmu) są odpowiednie pary operacji, stanowiące atrybuty danej permutacji π i wykonywanego z niej ruchu $v = (a, x, b, y)$ przez operację $j = \pi_a(x)$. Jeżeli $b \neq a$ na listę T zapisujemy dwie pary $(j1, j)$ oraz $(j, j2)$. W przypadku gdy $b = a$, zapisujemy tylko jedną parę; jeżeli $x < y$, to $(j, j2)$, gdy zaś $x > y$, to $(j1, j)$. Przy zapisywaniu danej pary najstarsza z nich jest usuwana; początkowo T jest całkowicie zapełniona przez pary $(0, 0)$. Ruch $v = (a, x, b, y)$ dla permutacji π jest zabroniony, jeżeli po jego wykonaniu w $\beta = (\pi_v)_b$ pojawi się para niekoniecznie sąsiednich operacji $(\beta(i'), \beta(i'')) \in T$, której nie było w π_b, gdzie $i' < i''$.

W danej iteracji algorytmu TS dla permutacji wejściowej $\pi \in \Pi$ (zwanej permutacją bazową tej iteracji) wyznacza się zbiór ruchów $V(\pi)$ oraz zbiór ich reprezentantów $W(\pi) = \{v[V_j(\pi)] : j \in A_1 \cup A_2\}$, gdzie ruch $v[V_j(\pi)]$ taki, że $v[V_j(\pi)] \in V_j(\pi)$ oraz $C_{\max}(\pi_{v[V_j(\pi)]}) = \min_{v \in V_j(\pi)} C_{\max}(\pi_v)$ jest reprezentantem zbioru $V_j(\pi)$. Następnie znajduje się ruchy $R(\pi) = \{v \in W(\pi) : (v \text{ nie jest zabroniony}) \vee (C_{\max}(\pi_v) < C^{TS})\}$, gdzie C^{TS} jest najmniejszą dotychczas znalezioną wartością funkcji celu. Ostatecznie w zbiorze $R(\pi)$ określa się ruch v' taki, że $C_{\max}(\pi_{v'}) = \min_{v \in R(\pi)} C_{\max}(\pi_v)$. Dalej następuje modyfikacja listy T oraz podstawienie $C^{TS} := C_{\max}(\pi_{v'})$, $\pi^{TS} := \pi_{v'}$, jeżeli $C_{\max}(\pi_{v'}) < C^{TS}$. Permutacja $\pi := \pi_{v'}$ jest traktowana jako bazowa dla kolejnej iteracji algorytmu.

Do budowy pamięci długoterminowej wykorzystuje się metodę skoku powrotnego (ang. back jump), zaproponowaną we wcześniejszych pracach autora i aktualnie uważaną za najbardziej perspektywiczne podejście przy konstrukcji algorytmów typu tabu search, patrz np. praca zbiorowa pod redakcją Aarts'a i Lenstry [1]. Ogólna idea metody polega na zapamiętywaniu określonej liczby najlepszych rozwiązań bazowych wraz z dodatkowymi elementami na liście L o ustalonej długości. Po wykonaniu $MaxIter$ iteracji (parametr algorytmu) bez poprawy wartości C^{TS} nie kończymy procesu poszukiwań, ale wykonujemy skok powrotny do bezpośrednio ostatnio zapamiętanego rozwiązania bazowego. Następnie, wychodząc z niego, po innej trajektorii niż poprzednio kontynuujemy poszukiwania.

W algorytmie wykorzystuje się dwa warunki stopu. Pierwszy polega na zatrzymaniu poszukiwań, kiedy wykonaliśmy już zadaną liczbę iteracji, a drugi — gdy lista L jest pusta.

7. Wyniki badań testowych

Algorytm TS zakodowano w Delphi 2 i uruchamiano na Pentiun Pro (200MHz). Bazą do konstrukcji testów było 120 szczególnie trudnych przykładów z [10] dla klasycznego problemu przepływowego z $r = 20 - -500$ oraz $ls - 2 = 5 - -20$. Dla każdego z nich wylosowano: liczbę maszyn w poszczególnych stanowiskach $m_l = 1 - -5$, czasy wykonywania $p_{l,j}$ (przez niewielkie zaburzenie czasów pierwotnych), liczbę palet $lp = ls/2 - -2ls$, czasy transportu oraz czasy przezbrojeń rodzinowo-sekwencyjnie niezależne i zależne. Jeden przykład bazowy dostarczał 20 przykładów; łącznie otrzymano ich 2400.

Ograniczone ramy pracy uniemożliwiają nawet pobieżne przedstawienie wyników testów. Pozostajemy więc tylko na stwierdzeniu, że algorytm TS poprawia najlepsze z rozwiązań (w sensie wartości funkcji celu) dostarczonych przez 14 algorytmów konstrukcyjnych z różnymi regułami priorytetowymi o około 10 – –20%, w czasie rzędu kilku minut dla przykładów o największych rozmiarach. Prawie 90% tej poprawy osiąga się już po wykonaniu początkowych 100 – –500 iteracji algorytmu.

LITERATURA

1. Aarts E., Lenstra J.K.: Local search in Combinatorial Optimization, John Wiley and Sons Ltd, Chichester, England, 1997.
2. Glover F., Laguna M.: Tabu Search. Kluwer Academic Publishers, Massachusetts USA, 1997.
3. Nowicki E.: Harmonogramowanie elastycznej linii produkcyjnej z równoległymi maszynami. Algorytm tabu search. Materiały III Krajowej Konferencji Badań Operacyjnych i Systemowych BOS'95, Warszawa, Gdynia 1995, IBS PAN, tom 1, pp. 164-174.
4. Nowicki E.: Zastosowanie techniki tabu search do harmonogramowania elastycznych gniazd produkcyjnych, Zeszyty Naukowe Politechniki Śląskiej, s. Automatyka,z. 118, Gliwice 1996, s. 153-154.
5. Nowicki E.: Problem przepływowy z ograniczoną liczbą palet. Algorytm tabu search. Elektrotechnika 1, z. 1, 1997, pp. 321-328.
6. Nowicki E.: The flow shop with buffers. A local search approach. European Journal of Operational Research 1998 (w druku).
7. Nowicki E., Smutnicki C.: The flow shop with paralled machines. A tabu search approach. European Journal of Operational Research 1998 (w druku).
8. Paulli J.: A hierarchical approach for the FMS scheduling problem. European Journal of Operational Research 86, 1995, pp. 32-42.
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: Dr hab.inż. Konrad Wala, prof. AGH

Abstract

The paper deals with the criterion of the makespan minimisation for the flow shop problem with parallel machines and setup times. The problem is being complicated by the disability of the material handling system to handle more than a fixed number of jobs at the time. Since the problem is NP-hard, an approximation algorithm is proposed based on a non trivial generalisation of the path elimination properties known for the classic flow shop problem. This algorithm is able to achieve excellent results for instances up to 500 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.