ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1990

Seria: AUTOMATYKA z.102 Nr kol.1084

Sławomir Jagiełło
Eugeniusz Toczyłowski
Instytut Automatyki Politechniki Warszawskiej

Szczegółowe minimalnokosztowe harmonogramowanie produkcji w gnieździe produkcyjnym z maszynami równoległymi [1]

Streszczenie. W pracy przedstawiono przybliżony algorytm dla zagadnienia harmonogramowania zadań produkcyjnych w ogólnym gnieździe produkcyjnym z maszynami równoległymi. Algorytm jest oparty o złożony schemat iteracyjny, w którym ciąg kolejnych rozwiązań jest ograniczany za pomocą opracowanego mechanizmu marginalnych kosztów oraz ograniczonego algorytmu symulowanego wyżarzania. Za kryterium jakości harmonogramu przyjęto minimalizację ogólnego kosztu będącego sumą kosztów zamrożenia robót w toku, kosztów magazynowania wyrobów ukończonych przed terminem pożądanym oraz sumy kar za nieterminowość zadań opóźnionych.

1. Wprowadzenie

W pracy jest rozważane zagadnienie harmonogramowania zadań produkcyjnych w ogólnym gnieździe produkcyjnym złożonym z wielu maszyn, niededykowanych, realizujących szeroki asortyment operacji przy założonym uzbrojeniu maszyn. Operacje wymagające tego samego uzbrojenia tworzą *typy* operacji. Do każdego typu operacji mogą należeć operacje o różnych czasach wykonywania. Operacje danego typu są wykonywane na określonym podzbiorze maszyn, przy czym podzbiory te nie są rozłączne. Dopuszcza się możliwość występowania niezerowego czasu przezbrajania maszyn przy zmianie typu wykonywanej operacji. Za kryterium jakości harmonogramu przyjmuje się minimalizację ogólnego kosztu będącego sumą kosztów zamrożenia robót w toku, kosztów magazynowania wyrobów ukończonych przed terminem pożądanym oraz sumy kar za nieterminowość zadań opóźnionych.

Rozważany typ zadań harmonogramowania należy do najbardziej ogólnych, najtrudniejszych modeli harmonogramowania. Algorytmy dokładne są zbyt pracochłonne dla realistycznych rozmiarów zadań. Z kolei algorytmy heurystyczne oparte na regułach szeregowania priorytetowego mogą być bardzo niedokładne. W pracy [10] (p.6.3) zaproponowano iteracyjny algorytm przybliżony dla minimalnokosztowego problemu szeregowania zadań bez alokacji. W tej pracy przedstawiono algorytm rozwinięty i uogólniony na przypadek problemu szeregowania z alokacją operacji. W algorytmie tym wprowadzono kilka metod kolejnego ograniczania przestrzeni poszukiwań rozwiązania. Dla ustalonej alokacji i uszeregowania operacji w celu wyznaczenia najlepszego harmonogramu jest rozwiązywane zadanie programowania liniowego określającego czasy rozpoczęcia wszystkich operacji. Zadanie to ma wygodną, specjalną strukturę, opracowano zatem efektywny algorytm jego rozwiązywania. Ogólny algorytm wyboru kierunku poszukiwań oparty został o metodę symulowanego wyżarzania.

W rozdziale 2 sformułowano rozważany w pracy problem harmonogramowania, natomiast w rozdz. 3 przedstawiono szczegółowo algorytm jego rozwiązania. W rozdziale 4 porównano jego efektywność z kilkoma algorytmami heurystycznymi opartymi na regułach priorytetowych, uznanych w literaturze jako dobre i bardzo dobre.

[1] Praca częściowo finansowana w ramach problemu R.P.I. 02 w temacie 5.3

2. Sformułowanie zadania

Dany jest zbiór maszyn $L = \{1, \ldots, l^*\}$, które mogą różnić się wydajnością. Niech $J = \{1, \ldots, n^*\}$ będzie zbiorem zadań do realizacji. Zadanie produkcyjne polega na wykonaniu ustalonego ciągu operacji $1, \ldots, m-1, m$. Dla każdej operacji m ze zbioru wszystkich operacji $M = \{1, \ldots, m^*\}$ ustalona jest wartość pracy w_m, niezbędnej do jej realizacji. Czas p_m wykonania operacji m określony jest jako iloraz pracy w_m niezbędnej do jej realizacji oraz wydajności v_{lm} maszyny, do której została przydzielona. Dalej zakładamy, że maszyny są produkcyjnie jednorodne, czyli $v_{lm} = v_l, \forall m$. Każda operacja m jest związana jednoznacznie z pewnym zadaniem produkcyjnym, które będziemy oznaczać przez j_m. Operacje są niepodzielne, a każda maszyna może wykonywać w danej chwili tylko jedną operację. Przyjmujemy, że po zakończeniu operacji m, rozpoczęcie następnej operacji n na tej samej maszynie może wymagać niezerowego czasu przygotowawczego s_{mn} - czasu przezbrojenia.

Dla każdego zadania $j \in J$ ustalony jest najwcześniejszy termin wprowadzenia do systemu a_j oraz pożądany termin ukończenia d_j. Niedotrzymanie terminu d_j wiąże się z dodatkowymi kosztami karnymi w przypadku jego przekroczenia lub z kosztami magazynowania w przypadku przedwczesnego wykonania zadania.

Z każdą operacją $m \in M$ i maszyną $l \in L$ związana jest funkcja $A|M \to L, l = A(m)$ zwana przydziałem operacji m do maszyny l. Celem harmonogramowania jest ustalenie przydziału wszystkich operacji do maszyn oraz wyznaczenie dla każdej maszyny kolejności i czasów wykonywania operacji tak, aby wskaźnik jakości harmonogramu, za który przyjmujemy łączny koszt realizacji zadań, był możliwie najmniejszy.

Wskaźnik jakości harmonogramu. Do kosztów realizacji zadań produkcyjnych, obok kosztów magazynowania oraz kar wynikających z ich nieterminowej realizacji, włączamy bezpośrednie koszty produkcji. Zakładamy, że wykonanie pojedynczej operacji $m \in M$ powoduje powstanie kosztów bezpośrednich (energia, zużycie materiałów, siła robocza itp.), zwiększających wartość obrabianego elementu. Niech q_m oznacza wartość produkcji dodanej, związanej z wykonaniem operacji m. Zostaje ona zamrożona w przetwarzanym elemencie do czasu całkowitego zakończenia jego obróbki – czasu ukończenia zadania. Jeżeli przez t_m oznaczymy czas ukończenia operacji m zadania $j \in J$, a przez f_j czas ukończenia ostatniej operacji zadania j, to koszt związany z zamrożeniem wartości dodanej q_m przy operacji m wyniesie $F_m = q_m(f_j - t_m)$

Niech $M_e \subset M$ będzie zbiorem operacji końcowych wszystkich zadań. Koszt magazynowania w jednostce czasu j-tego wyrobu gotowego oznaczymy przez h_{j_m}, a karę za przekroczenie o jednostkę czasu pożądanego terminu jego ukończenia przez r_{j_m}. Niech Q_m oznacza sumaryczną wartość elementu po wykonaniu operacji m. Zakładamy, że początkowa wartość elementu przed wprowadzeniem go do systemu wynosi $Q_0 = 0$. Ogólny koszt zamrożenia robót w toku F_1, koszt magazynowania wyrobów gotowych F_2 oraz kary za przekroczenie pożądanych terminów ukończenia zadań F_3 przedstawimy za pomocą wzorów:

$$F_1 = \sum_{m \in M_e} Q_m t_m - \sum_{m \in M} q_m t_m \tag{1}$$

$$F_2 = \sum_{m \in M_e} (Q_m + h_{j_m})(d_{j_m} - t_m)^+ \tag{2}$$

$$F_3 = \sum_{m \in M_e} r_{j_m}(t_m - d_{j_m})^+ \tag{3}$$

Ogólny koszt realizacji zadań, uwzględniający powyższe trzy składniki zapiszemy w postaci:

$$F = \sum_{m \in M_e} (h_{j_m} t_m^- + \bar{r}_{j_m} t_m^+) - \sum_{m \in M} q_m t_m + \sum_{m \in M_e} Q_m d_{j_m} \tag{4}$$

przy czym $t_m^- = \max(0, d_{j_m} - t_m), t_m^+ = \max(0, t_m - d_{j_m}), \bar{r}_{j_m} = r_{j_m} + Q_m$.

W szczególności powyższy wzór osiąga minimum, gdy nie występują koszty magazynowania wyrobów gotowych ($t_m^- = 0$) oraz kary za przekroczenie pożądanych terminów ukończenia zadań ($t_m^+ = 0$), a czasy przejścia zadań przez system są jak najmniejsze. Algorytm harmonogramowania minimalizujący powyższy wskaźnik jakości preferuje produkcję "akurat na czas".

3 Algorytm harmonogramowania

Algorytm rozwiązania przedstawionego wyżej problemu zbudowany został w oparciu o złożony, iteracyjny schemat poszukiwań możliwie najlepszego rozwiązania przybliżonego, wykorzystujący metodę symulowanego wyżarzania.

Na początku k-tej iteracji algorytmu znany jest przydział operacji do maszyn A^k, uszeregowanie operacji $S^k = (S_1, S_2, \ldots, S_{l^*})$ oraz harmonogram szczegółowy T^k. W sąsiedztwie aktualnego rozwiązania (A^k, S^k) liczba możliwych przydziałów operacji do maszyn oraz liczba możliwych uszeregowań operacji są bardzo duże. Dlatego też sąsiedztwo to znacznie zredukujemy, wybierając do poszukiwań jedynie rozwiązania najbardziej obiecujące, tj. takie, które dają dużą szansę zmniejszenia wskaźnika jakości. Zbiór par najbardziej obiecujących alokacji i uszeregowań operacji (A, S), wygenerowanych z aktualnego punktu (A, S) nazywać będziemy *ograniczonym* sąsiedztwem tego punktu i oznaczamy przez $Z(A, S)$.

W celu zwiększenia efektywności poszukiwań sąsiedztwo $Z(A, S)$ zredukujemy dalej do zbioru $Z_\lambda(A, S)$ zawierającego tylko λ najbardziej obiecujących elementów, gdzie λ jest parametrem metody. Zbiór $Z_\lambda(A, S)$ nazywamy dalej *zredukowanym* sąsiedztwem punktu (A, S). Aby umożliwić selekcję najbardziej obiecujących punktów sąsiedztwa $Z(A, S)$ bez konieczności kosztownych obliczeń funkcji celu (wymagających rozwiązywania zadania programowania liniowego), opracowano metodę oszacowania ogólnego kosztu związanego z każdym punktem sąsiedztwa.

A oto pełny zapis k-tej iteracji algorytmu:

1. wygenerowanie ograniczonego sąsiedztwa $Z(A^k, S^k)$ aktualnego punktu (A^k, S^k),
2. wyznaczenie zredukowanego sąsiedztwa $Z_\lambda(A, S)$ zawierającego λ najbardziej obiecujących rozwiązań (po uprzednim oszacowaniu wartości funkcji celu dla każdego elementu sąsiedztwa $Z(A^k, S^k)$),
3. wybór nowej alokacji i szeregowania operacji (A^{k+1}, S^{k+1}),
4. obliczenie szczegółowego harmonogramu, w którym dla ustalonej alokacji i uszeregowania (A^{k+1}, S^{k+1}) wyznacza się optymalne czasy ukończenia operacji T^{k+1},
5. akceptacja lub odrzucenie obranego rozwiązania za pomocą reguł metody symulowanego wyżarzania.

Powyższe kroki algorytmu są realizowane iteracyjnie, aż do spełnienia kryterium stopu.

3.1 Generowanie ograniczonego sąsiedztwa

Uszeregowanie $S_l \in M \times M$ reprezentuje relację poprzedzania operacji wykonywanych na maszynie l. Zachodzi $(m, n) \in S_l$, jeżeli $A(m) = A(n) = l$ oraz operacja m bezpośrednio poprzedza operację n. Niech zbiór $R \subset M \times M$ reprezentuje relacje poprzedzania operacji należących do tych samych zadań. Zachodzi $(m, n) \in R$, jeżeli m, n są operacjami tego samego zadania oraz m jest bezpośrednim poprzednikiem n.

Dla ustalonego harmonogramu parę operacji $(m, n) \in S_l$ nazywać będziemy parą operacji *przyległych*, jeżeli $t_n - t_m = p_n + s_{mn}$. Operację m nazwiemy operacją *swobodną*, jeżeli nie tworzy pary operacji przyległych ani ze swym poprzednikiem, ani ze swym następnikiem na maszynie $l = A(m)$. Dla ustalonego

τ operację n uważać będziemy za operację *związaną bezpośrednio* z operacją m, jeżeli $(m,n) \in S_l$ lub $(m,n) \in R$ oraz przy zmianie czasu ukończenia t_m operacji m o τ operacje m, n stają się przyległe:

$$\begin{array}{ll} t_n - (t_m + \tau) < p_n + s_{mn}, & (m,n) \in S_l \\ t_n - (t_m + \tau) < p_n, & (m,n) \in R \setminus S_l \end{array} \tag{5}$$

Zbiór operacji związanych bezpośrednio z operacją m przy jej przesunięciu o τ oznaczymy przez $\eta(m,\tau)$. Łatwo zauważyć, że $|\eta(m,\tau)| \leq 2$ oraz jeżeli $n \in \eta(m,\tau)$, to $m \in \eta(n,-\tau)$.

Definicja 1 *Zbiorem operacji* związanych *z operacją m przy przesunięciu τ nazywamy zbiór*

$$\gamma(m,\tau) = \bigcup_{i=0}^{j} \eta(r_i, \tau_{r_i}) \tag{6}$$

przy czym $r_0 = m, \tau_{r_0} = \tau$ oraz τ_{r_i} jest przesunięciem operacji $r_i \in \eta(r_{i-1}, \tau_{r_{i-1}})$, wynikającym z przesunięcia τ, tzn. $\tau_i = t_{r_i} - t_{r_{i-1}} + \tau_{r_{i-1}} - t$, gdzie

$$t = \begin{cases} p_{r_i} + s_{r_{i-1} r_i} & gdy \tau \geq 0 \\ p_{r_{i-1}} + s_{r_i r_{i-1}} & gdy \tau < 0 \end{cases} \tag{7}$$

oraz dla j zachodzi $\eta(r_j, \tau_{r_j}) = 0$

Dla wygody wprowadzimy także pojęcie operacji *pustej*. Będzie ona umieszczana w momentach nie wykorzystania maszyny, a czas jej wykonania równy będzie czasowi przestoju maszyny. Zbiór wszystkich operacji pustych oznaczać będziemy przez $\mathcal{N}$.

W przypadku zmiany alokacji za obiecujące uznamy takie nowe przyporządkowanie operacji do maszyn A^{k+1}, które prowadzi do zmniejszenia obciążenia maszyn nadmiernie eksploatowanych. Niech $b(l)$ będzie obciążeniem maszyny l. Pierwszą fazą generowania nowej alokacji będzie uporządkowanie maszyn $l \in L$ według obciążenia $b(l)$: $l_1, \ldots, l_i, \ldots, l_{l^*}$, przy czym $b(l_i) \leq b(l_{i+1})$. Niech $N_i = \{n | n \in \mathcal{N}, A(n) = l_i, p_n \geq p\}$ będzie zbiorem operacji pustych na maszynie l_i, których czas wykonania jest nie mniejszy od zadanego parametru p.

Definicja 2 *Sąsiedztwem H_n operacji $n \in M \cup \mathcal{N}$ nazywać będziemy zbiór operacji $m \in M$ spełniających warunki:*

1. $A(m) = l_j \neq A(n) = l_i;\ j \in \{i+1, \ldots, l^*\}$
2. $t_m \in [(t_n - p_n - \delta), t_n + \delta]$ *(przy czym $\delta \geq 0$ parametr algorytmu)*
3. $n \notin \gamma(m,\tau),\ \tau = t_n - t_m$

Warunek 1 definicji 2 sprawia, że do otoczenia operacji n, przyporządkowanej do maszyny l_i mogą należeć jedynie operacje wykonywane na bardziej obciążonych maszynach. Warunek 2 definicji zawęża zbiór do tych operacji, których czasy ukończenia znajdują się w odpowiednim przedziale czasowym oraz warunek 3 do operacji pozwalających na ich zamianę z operacją n.

Nową alokację otrzymamy poprzez zamianę operacji pustej $n \in N_j, j \in L$ z pewną operacją $m \in H_n$. Jeżeli początkowo $A(n) = l_i, A(m) = l_j, n \in N_j, m \in H_n$, a czasy wykonania operacji wynoszą odpowiednio p_m, p_n, to po zamianie operacji m, n otrzymamy: $A(n) = l_j, A(m) = l_i$ oraz nowe czasy wykonania $\bar{p}_m = w_m / v_{l_j}$ i $\bar{p}_n$ – nowy czas przestoju maszyny j. Mogą przy tym ulec zmianie czasy t_m, t_n zakończenia operacji m, n oraz czasy ukończenia wszystkich operacji związanych z nimi. Czas wykonania operacji pustej $n \in \mathcal{N}$ określiliśmy wcześniej jako czas przestoju maszyny. Skorzystamy teraz z faktu, że czas ten możemy modyfikować. Czas wykonania p_n operacji pustej $n \in N_j$ zredukujemy, jeżeli zwiększenie czasu ukończenia jego bezpośredniego poprzednika lub zmniejszenie czasu ukończenia bezpośredniego następnika na maszynie wpłynie na poprawę wskaźnika jakości harmonogramu. Zysk

redukcji o τ czasu wykonania operacji pustej $n \in N_j$ przy jej zamianie z operacją $m \in M$ oznaczymy przez $U_z(m)$.

Rozpatrzmy teraz zmianę uszeregowania operacji. Zmiany położenia operacji swobodnych nie prowadzą do zmniejszenia ogólnego kosztu produkcji. Weźmy pod uwagę parę operacji przyległych m, n. Możemy przypuszczać , że przy braku ograniczenia na kolejność ich wykonania można dobrać czasy ukończenia t_m, t_n przy odmiennej od pierwotnie zakładanej kolejności ich wykonania, które mogą zapewnić poprawę wskaźnika jakości harmonogramu. Zmiana kolejności wykonania każdej pary operacji przyległych prowadzi więc do nowego, obiecującego uszeregowania S^{k+1}.

3.2 Wyznaczanie zredukowanego sąsiedztwa

Szacowanie ogólnego kosztu produkcji. Zmieniając przyporządkowanie operacji do maszyn lub kolejność wykonania dwóch operacji przyległych, ulegają zmianie czasy ukończenia wszystkich operacji z nimi związanych. Każda zmiana czasu ukończenia pojedynczej operacji wpływa na ogólny koszt produkcji. Koszt zmiany położenia pojedynczej operacji o τ jednostek czasowych (τ dodatnie lub ujemne) wyznaczymy ze wzoru:

$$u_m(\tau) = \begin{cases} -\tau q_m & \text{gdy} \quad m \in M \setminus M_e \\ -\tau q_m + \tau \bar{r}_{j_m}, & t_m \geq d_{j_m},\ m \in M_e \\ -\tau q_m - \tau h_{j_m} & t_m < d_{j_m},\ m \in M_e \end{cases} \tag{8}$$

Marginalnym kosztem przesunięcia o τ operacji $m \in M$ nazwiemy koszt równy sumie kosztów zmian położenia wszystkich operacji związanych $n \in \gamma(m, \tau)$. Koszt ten oznaczać będziemy przez $U_m(\tau)$.

$$U_m(\tau) = \sum_{n \in \gamma(m,\tau)} u_n(\tau_n) + u_m(\tau) \tag{9}$$

przy czym τ_n jest przesunięciem operacji n wynikającym z przesunięcia τ (por. def.1). Koszt zmiany kolejności wykonania pary operacji przyległych m, n wyznaczymy ze wzoru:

$$E_{mn} = U_m(\tau_m) + U_n(\tau_n) \tag{10}$$

przy czym τ_m, τ_n są minimalnymi czasami przesunięcia operacji m, n przy zmianie relacji poprzedzania, $\tau_m = p_n + s_{nm} + s, \tau_n = -(p_m + s_{mn}) + s$, gdzie $s = \max(0, s_{vn} - s_{vm})$, v - operacja wykonywana bezpośrednio przed operacją m w uszeregowaniu S^k oraz bezpośrednio przed operacją n w uszeregowaniu S^{k+1}. Koszt E_{mn} jest oszacowaniem od góry rzeczywistego kosztu zmiany kolejności wykonania operacji m, n.

Zmiana alokacji operacji $m \in M$ może wywołać zmianę czasu t_m jej ukończenia oraz zmiany czasów ukończenia operacji $n \in \gamma(m, \tau)$. Koszt E_m, związany ze zmianą alokacji operacji m wyznaczymy podobnie jak koszt zmiany uszeregowania operacji. Włączymy do niego ewentualny zysk $U_z(m), U_z(m) \leq 0$, związany z korzystnymi zmianami czasów ukończenia byłego, bezpośredniego poprzednika i następnika operacji m.

$$E_m = U_m(\tau) + U_z(m) \tag{11}$$

przy czym ewentualne przesunięcie w przód operacji m, $\tau^+ = \max(0, \bar{t}_m - t_m)$ oraz ewentualne przesunięcie w tył $\tau^- = \min(\bar{t}_m - \bar{p}_m - t_m + p_m, 0)$, gdzie $\bar{t}_m, \bar{p}_m$ są odpowiednio nowym czasem ukończenia oraz czasem wykonania operacji m na nowej maszynie. Koszt E_m jest oszacowaniem od góry rzeczywistego kosztu zmiany alokacji operacji m.

Redukcja sąsiedztwa. Korzystając z wyznaczonych oszacowań kosztów zmiany uszeregowania oraz alokacji operacji elementy sąsiedztwa $Z(A^k, S^k)$, porządkujemy według niemalejących kosztów. W rezultacie wyboru λ najlepszych elementów uzyskujemy zbiór $Z_\lambda(A^k, S^k)$.

3.3 Wybór nowej alokacji i uszeregowania operacji

Wyboru nowej alokacji i uszeregowania operacji dokonujemy poprzez losowanie jednego elementu (A^{k+1}, S^{k+1}) sąsiedztwa $Z_\lambda(A^k, S^k)$. Każdemu elementowi $(A, S) \in Z(A^k, S^k)$ przypiszemy liczbę, będącą prawdopodobieństwem jego wylosowania. Określimy ją wzorem:

$$p(A,S) = \alpha c^{F(A^k,S^k)-\tilde{F}(A,S)} \tag{12}$$

przy czym $(A, S) \in Z_\lambda(A^k, S^k)$ punkt z ograniczonego sąsiedztwa, α współczynnik normalizujący, $c \geq 1$ parametr algorytmu, $F(A^k, S^k)$ wartość funkcji celu w k iteracji oraz $\tilde{F}(A, S)$ szacowana wartość funkcji celu dla punktu (A, S).

Im większa wartość parametru c, tym bardziej preferowane są rozwiązania, które wydają się najbardziej poprawiać aktualne rozwiązanie (A^k, S^k). Wylosowany element (A^{k+1}, S^{k+1}) może zostać zaakceptowany lub odrzucony w dalszym etapie algorytmu.

3.4 Harmonogramowanie szczegółowe

Zadanie harmonogramowania szczegółowego w k-tym kroku algorytmu polega na wyznaczeniu czasów ukończenia operacji tak, aby ogólny koszt harmonogramu dla alokacji A^{k+1} oraz uszeregowania S^{k+1} osiągnął minimum. Zadanie to sprowadza się do rozwiązania zadania liniowego postaci:

$$\min F(A^{k+1}, S^{k+1}) = \sum_{m \in M_e} (h_{j_m} t_m^- + \bar{r}_{j_m} t_m^+) - \sum_{m \in M} q_m t_m \tag{13}$$

przy ograniczeniach

$$\begin{array}{ll} t_n - t_m \geq p_n & (m,n) \in R \\ t_n - t_m \geq p_n + s_{mn} & (m,n) \in S_l, l = 1, \ldots, l^* \\ t_m, t_m^-, t_m^+ \geq 0 & m \in M \setminus M_0 \\ t_m \geq a_{j_m} & m \in M_0 \\ t_m = d_{j_m} + t_m^+ - t_m^- & m \in M_e \end{array} \tag{14}$$

przy czym M_0 jest jest zbiorem operacji początkowych zadań.

Zadanie to ma wygodną strukturę (zadanie dualne jest zadaniem sieciowym). Dla takich problemów można wykorzystać znane techniki programowania sieciowego. Należy jednak zdawać sobie sprawę z tego, że przy dużej liczbie operacji potrzeba będzie znacznego czasu na jego rozwiązanie. Zadanie to jest jednak rozwiązywane tylko dla jednego, wybranego elementu sąsiedztwa $Z_\lambda(A^k, S^k)$. Celowe jest dalsze rozwijanie efektywnych metod dokładnych i przybliżonych dla tego zadania.

3.5 Akceptacja lub odrzucenie rozwiązania

Algorytm akcepacji lub odrzucenia wybranej alokacji A^{k+1} oraz uszeregowania operacji S^{k+1} zbudowany został w oparciu o metodę symulowanego wyżarzania (por. [9]). Polega on na:

1. wyznaczeniu prawdopodobieństwa

$$p_k = \min\{exp[(F(A^k, S^k) - F(A^{k+1}, S^{k+1}))/t^k], 1\} \tag{15}$$

2. zaakceptowaniu z prawdopodobieństwem p_k alokacji A^{k+1} i uszeregowania S^{k+1} lub wykonaniu z prawdopodobieństwem komplementarnym $1 - p_k$ podstawienia $A^{k+1} = A^k, S^{k+1} = S^k$

3. zmodyfikowaniu parametru temperatury t_k

4. podstawieniu $k = k + 1$ i przejściu do następnej fazy algorytmu.

Ze względu na zbyt dużą liczbę iteracji wymaganych do dostatecznego obniżenia temperatury, przyjmowano, że algorytm zostaje zatrzymany po zadanej maksymalnej liczbie iteracji lub po zadanym czasie obliczeń.

3.6 Punkt startowy

W celu wyznaczenia początkowej alokacji A^0 przyjmijmy pewne pożądane wielkości obciążenia $b(l), l \in L$ poszczególnych maszyn. Pogrupujmy operacje w zbiory operacji podobnych $M^\nu, \nu \in \mathcal{M}$, wymagających tych samych zasobów, mających takie same czasy wykonania. Początkowe przyporządkowanie operacji do maszyn wyznaczamy tak, aby maksymalna (dodatnia lub ujemna) odchyłka od pożądanych wielkości obciążenia maszyn była jak najmniejsza. Jeżeli przez $o_{\nu l}$ oznaczymy zmienną decyzyjną, określającą ilość operacji typu ν przyporządkowanych do maszyny l, to zadanie wyznaczenia początkowej alokacji będzie można zapisać w postaci

$$\max u \tag{16}$$

$$\sum_{\nu \in \mathcal{M}} o_{\nu l} p_{\nu l} + u \leq b(l), \quad l \in L \tag{17}$$

$$\sum_{l \in L} o_{\nu l} = |M^\nu|, \quad \nu \in \mathcal{M} \tag{18}$$

Początkowego uszeregowania operacji można dokonać za pomocą dowolnej reguły priorytetów. W badanej w pracy wersji algorytmu początkowe uszeregowanie wyznaczano przypadkowo.

4. Wyniki numeryczne

Przedstawiony algorytm został zaimplementowany w języku FORTRAN oraz uruchomiony na komputerze IBM PC. Przetestowano go na serii przykładów, różniących się danymi dotyczącymi liczby maszyn, operacji oraz zadań (patrz tab.1).

symbol przykładu	liczba maszyn	liczba zadań	liczba typów zadań	liczba operacji	liczba typów operacji
EX_{010}	1	7	2	21	6
EX_{011}	1	20	4	35	7
EX_{021}	2	15	1	30	2
EX_{022}	2	30	4	35	4
EX_{023}	2	10	2	15	3
EX_{024}	2	16	2	48	6
EX_{030}	3	8	3	15	6
EX_{031}	3	15	4	30	8
EX_{032}	3	16	2	48	6
EX_{040}	4	33	5	80	10
EX_{050}	5	15	2	30	4

Tablica 1: Wykaz parametrów testowanych przykładów. Parameters of tested problems.

Algorytm testowano z uwzględnieniem przezbrojeń i bez przezbrojeń oraz z różną wartością parametru λ, określającego liczność zredukowanego sąsiedztwa aktualnego rozwiązania. Przy testowaniu problemów z przezbrojeniami algorytm wykazywał właściwość tworzenia na poszczególnych maszynach bloków przyległych operacji podobnych, pomiędzy którymi nie występują czasy przezbrojeń.

W opracowanym algorytmie, oznaczanym dalej jako HIS najlepsze rezultaty osiągnięto przy niezbyt dużym λ rzędu $0.2m^*$, gdzie m^* liczba wszystkich operacji. Parametr ten należy dobierać tak, aby błąd przy wyborze najbardziej obiecujących elementów sąsiedztwa, wynikający z szacowania wartości funkcji celu był jak najmniejszy. Efektywność algorytmu HIS porównano z kilkoma algorytmami heurystycznymi opartymi na regułach priorytetowych, uznanych w literaturze jako dobre i bardzo dobre. Algorytmy te to:

- CR (Critical ratio) - wyznaczany jest wskaźnik krytyczności jako iloraz różnicy pożądanego terminu ukończenia a aktualnego czasu oraz czasu realizacji zadania. Do systemu jako pierwsze wprowadzane jest zadanie o najmniejszym wskaźniku krytyczności.
- SPT (Shortest proc.time) - zadania są wprowadzane do systemu wg czasów realizacji.
- STO (Slack time per oper.)- wskaźnik, wg którego zadania wprowadzane są do systemu, obliczany jest jako stosunek czasu, jaki pozostał do pożądanego terminu ukończenia oraz liczby niezrealizowanych operacji zadania.
- STR (Slack time remaining)- brany jest pod uwagę czas pozostający do pożądanego terminu ukończenia zadania oraz czas realizacji zadania. Ich różnica jest wskaźnikiem , wg którego zadania są wprowadzane do systemu.

Uzyskane wyniki przedstwione są w tab.2. Użyte tam symbole oznaczają: F - ogólny koszt harmonogramu, F_1 - koszt zamrożenia robót w toku, F_2 - koszt magazynowania gotowych wyrobów, F_3 - kary za przekroczenie pożądanych terminów ukończenia zadań.

Z porównania algorytmów widać, że algorytm HIS okazał się najlepszy dla 10 zadań na ogólną ilość 11. W kilku przypadkach wartość funkcji celu była nawet kilkakrotnie lepsza niż w algorytmach heurystycznych. W jednym tylko przypadku (EX_{011}) algorytm HIS dał gorsze rezultaty od algorytmów STO i STR. Wynika to ze sposobu definiowania sąsiedztwa aktualnego rozwiązania, który nie uwzględnia zmian alokacji oraz uszeregowania jednocześnie dla całej grupy operacji przyległych tego samego typu.

5. Zakończenie

Aktualnie prowadzone są prace w celu poprawienia efektywności algorytmu. Rozszerza się pojęcie sąsiedztwa danego rozwiązania o zmiany położenia całych *bloków* operacji podobnych, między którymi nie występują przezbrojenia. Powinno to zapewnić uzyskanie lepszych rezultatów szczególnie w przykładach zawierających dużą liczbą operacji niewielu typów. Czas działania algorytmu w stosunku do algorytmów heurystycznych jest wysoki. Dla każdego z testowanych przykładów algorytm wykonywał kilkaset iteracji, a na każdą z nich potrzebował na komputerze IBM/AT (12MHz) od kilku sekund do kilkunastu minut. Czas jednej iteracji zależy głównie od liczby operacji. W przypadku problemów zawierających duża liczbę zadań realizowanych w dłuższym horyzoncie czasu, podział horyzontu harmonogramowania na okresy oraz konstrukcja odpowiedniego zadania nadrzędnego przydziału operacji do poszczególnych okresów może pozwolić na zwiększenie rozmiarów rozważanych problemów oraz zwiększenie szybkości algorytmu.

Literatura

[1] Balas E., 1979, "Disjunctive Programming" *Annals of Discrete Mathematics* 5,3-51.

[2] Fox M.S., S. Smith, 1984, "ISIS - a Knowledge-Based System for Factory Scheduling" *Expert Systems* Vol.1, No. 1, 25 - 49.

[3] French S., 1982, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. Horwood, Chichester.

[4] Gere W.S., 1966, "Heuristics in Job Shop Scheduling" *Management Science*, 13, 164-180.

[5] Grabowski J. *Uogólnione zagadnienia optymalizacji kolejnościowej w dyskretnych systemach produkcyjnych* Monografie ICT PWr, Monografie 9, Wyd. Pol. Wroc., Wrocław, 1979

[6] Jagiełło S. *Harmonogramowanie zadań produkcyjnych w ogólnym gnieździe produkcyjnym przy kryterium kosztowym* Praca Dyplomowa, Instytut Automatyki PW, 1989

symbol		HIS	CRT	SPT	STR	STO
EX_{010}	F	**3162.0**	**8358.0**	**8126.0**	**7624.0**	**7633.0**
	F_1	1552.0	5030.0	3682.0	2836.0	5009.0
	F_2	40.0	3328.0	3584.0	4788.0	2624.0
	F_3	1570.0	0.0	860.0	0.0	0.0
EX_{011}	F	**45933.0**	**47769.0**	**72909.0**	**31494.0**	**40013.0**
	F_1	2573.0	6706.0	5715.0	3821.0	8274.0
	F_2	5012.0	1980.0	7633.0	4875.0	450.0
	F_3	38348.0	39083.0	59561.0	22796.0	31289.0
EX_{021}	F	**3300.0**	**32125.0**	**31965.0**	**32125.0**	**32125.0**
	F_1	970.0	765.0	1275.0	765.0	765.0
	F_2	1980.0	31360.0	30690.0	31360.0	31360.0
	F_3	350.0	0.0	0.0	0.0	0.0
EX_{022}	F	**6215.0**	**14895.0**	**15515.0**	**16065.0**	**16065.0**
	F_1	280.0	290.0	190.0	290.0	290.0
	F_2	3285.0	14605.0	14045.0	15775.0	15775.0
	F_3	2650.0	0.0	1280.0	0.0	0.0
EX_{023}	F	**555.0**	**1275.0**	**3195.0**	**1275.0**	**1275.0**
	F_1	250.0	380.0	210.0	380.0	380.0
	F_2	305.0	845.0	1885.0	845.0	845.0
	F_3	0.0	50.0	1100.0	50.0	50.0
EX_{024}	F	**11153.0**	**11384.0**	**42203.0**	**10104.0**	**9763.0**
	F_1	7293.0	6452.0	24695.0	5787.5	6596.5
	F_2	1050.0	1936.0	1982.5	1716.0	154.0
	F_3	2810.0	2995.5	15525.5	2600.5	3012.5
EX_{030}	F	**233.0**	**6068.0**	**6740.0**	**6284.0**	**6068.0**
	F_1	133.0	627.0	609.0	306.0	627.0
	F_2	100.0	5441.0	5884.0	5978.0	5441.0
	F_3	0.0	0.0	247.0	0.0	0.0
EX_{031}	F	**4377.0**	**6875.0**	**8797.0**	**7151.0**	**6578.0**
	F_1	1234.0	2651.0	1818.0	986.0	2333.0
	F_2	2893.0	4224.0	5819.0	6165.0	4245.0
	F_3	250.0	0.0	1160.0	0.0	0.0
EX_{032}	F	**8213.0**	**13358.0**	**27644.0**	**10484.0**	**10644.0**
	F_1	3929.0	8010.0	14316.0	5774.0	7357.0
	F_2	660.0	4306.0	4863.0	4713.0	3147.0
	F_3	3624.0	1042.5	8465.0	0.0	140.0
EX_{040}	F	**12979.0**	**21610.0**	**55819.0**	**22213.0**	**23551.0**
	F_1	1737.0	4869.0	3219.0	4921.0	5401.0
	F_2	2702.0	66.0	11820.0	722.0	0.0
	F_3	8540.0	16675.0	40780.0	16570.0	18150.0
EX_{050}	F	**1305.0**	**1595.0**	**2980.0**	**2087.0**	**1595.0**
	F_1	390.0	590.0	315.0	317.0	590.0
	F_2	790.0	455.0	815.0	920.0	455.0
	F_3	125.0	550.0	1850.0	850.0	550.0

Tablica 2: Porównanie efektów działania algorytmu HIS z innymi algorytmami. Comparizon of HIS algorithm with the priority-based algorithms.

[7] Kennington J.F., R.V. Helgason, *Algorithms for Network Programming*, (John Willey & Sons, New York, 1981).

[8] Morton T.E., S.R.Lawrence, S. Rajagopalan, S. Kekre, 'SCHED-STAR: A Price Based Schop Scheduling Module' Working Paper, July 1987, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh.

[9] Syski W., Toczyłowski E., Strategie poszukiwań z wykorzystaniem symulowanego wyżarzania, materiały tej konferencji.

[10] Toczyłowski E., *Niektóre strukturalne metody optymalizacji do sterowania w dyskretnych systemach wytwarzania*, WNT, Warszawa 1989.

Recenzent: Doc.dr h.inż. F.Marecki

Wpłynęło do Redakcji do 1990-04-30.

Detailed Minimum Cost Scheduling for General Job-Shops with Parallel Machines

Summary We present the algorithm for scheduling in a general job-shop cell which consists of several non-dedicated machines. The method is based on a composite iterative scheme, in which the search directions are limited with the help of the marginal cost computations and a restricted simulated annealing algorithm. The primary objective of scheduling is to minimize total production costs, subject to resource and precedence constraints. The total costs include direct production costs, set-up costs, earliness penalties, tardiness penalties and holding costs of the work-in-process.

СОСТАВЛЕНИЕ ПОДРОБНЫХ МИНИМАЛЬНОСТОИМОСТНЫХ ПРОИЗВОДСТВЕННЫХ ГРАФИКОВ ДЛЯ ПРОИЗВОДСТВЕННОГО ГНЕЗДА С ПАРАЛЛЕЛЬНЫМИ МАШИНАМИ

Р е з ю м е

В работе представлен приближенный алгоритм для вопроса составления графиков производственных задач для общего производственного гнезда с параллельными машинами. Алгоритм разработан с использованием сложной итерационной схемы, для которой последовательность очерёдных решений ограничена с использованием разработанного механизма маргинальных расходов, а также ограниченного алгоритма симулированного прокаливания. В качестве критерия качества графика принята минимизация общих расходов, являющихся суммой расходов замораживания выполняемых работ, расходов хранения изделий реализованных досрочно, а также суммы штрафов за несвоевременное с опозданием выполнение задач.