ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1992

Seria: AUTOMATYKA z. 109 Nr kol. 1175

Eugeniusz Nowicki

Politechnika Wrocławska

ALGORYTMY APROKSYMACYJNE SZEREGOWANIA ZADAŃ NA RÓWNOLEGŁYCH MASZYNACH ZE ZMIENNYMI CZASAMI WYKONYWANIA

APPROXIMATION ALGORITHMS FOR THE M-MACHINE SCHEDULING PROBLEM ON PARALLEL MACHINES WITH CONTROLLABLE PROCESSING TIMES

АППРОКСИМАЦИОННЫЕ АЛГОРИТМЫ СОСТАВЛЕНИЯ РАСПИСАНИЯ ЗАДАЧ В СИСТЕМЕ С ПАРАЛЛЕЛЬНЫМИ МАШИНАМИ И С УЧЕТОМ ИЗМЕНЕНИЯ ДЛИТЕЛЬНОСТИ ОБСЛУЖИВАНИЯ

Streszczenie: W pracy analizuje się problem szeregowania zadań na m (m≥2) identycznych, równoległych maszynach, w którym czasy wykonywania zadań są zmiennymi decyzyjnymi. Stawia się problem minimalizacji łącznego kosztu uszeregowania. Do jego rozwiązania proponuje się algorytm aproksymacyjny ze współczynnikiem najgorszego przypadku równym $(\rho+\sqrt{\rho(m-1)})/2+ 1/4 + O(1/\sqrt{m})$, gdzie ρ jest współczynnikiem najgorszego przypadku algorytmu rozwiązującego badany problem z ustalonymi czasami wykonywania.

Summary: The paper deals with a problem of scheduling on m (m≥2) identical parallel machines in which job processing times are decision variables. An approximation algorithm for minimizing the overall schedule cost with the worst-case performance ratio equal to $(\rho+\sqrt{\rho(m-1)})/2 + 1/4 + O(1/\sqrt{m})$ is provided, where ρ is the worst-case performance ratio of a procedure for solving the pure scheduling problem.

Резюме: В статье представляется система с параллельными m-машинами, в которой времена выполнения задач являются переменными подлежащими выбору. Ставится проблема минимизации суммарной стоимости расписания. Для этой проблемы формулируется аппроксимационный алгоритм с оценкой погрешности равной $(\rho+\sqrt{\rho(m-1)})/2 + 1/4 + O(1/\sqrt{m})$, где ρ - оценка погрешности алгоритма для исследуемой проблемы с фиксированными временами выполнения задач.

1. Wstęp i sformułowanie problemu

Problemy szeregowania zadań i rozdziału zasobów są jednymi z najtrudniejszych problemów optymalizacji dyskretnej. Duża liczba zmiennych decyzyjnych oraz ograniczeń powoduje, że standardowe algorytmy zawodzą. Dlatego też wyodrębnia się pewne duże klasy tych problemów i wykorzystując specyficzne własności, buduje się odpowiednie algorytmy; z reguły są to algorytmy aproksymacyjne. Oczywiście, im klasa jest ogólniejsza, tym własności są coraz słabsze i efektywność algorytmów maleje. Istotny więc staje się problem odpowiedniego kompromisu między ogólnością modelu a istnieniem wystarczająco silnych własności, pozwalających skonstruować efektywne algorytmy. Wyodrębnienie takiej klasy (lub klas) i jej dokładne zbadanie umożliwia następnie stosunkowo łatwe rozszerzanie wyników dla ogólniejszych modeli.

Przedstawiana praca jest kontynuacją szeregu prac (np. [2], [4], [6], [8], [14]–[17], w których modelem zadania jest funkcja typu koszt/czas - powszechnie stosowana w programowaniu sieciowym. W tym modelu zakłada się, że do wykonania zadania potrzebny jest, oprócz maszyny (zasób odnawialny), dodatkowy zasób nieodnawialny, podzielny w sposób ciągły i mierzony przez jego koszt. W konsekwencji czas wykonania zadania może zmieniać się, w pewnym domkniętym przedziale, w zależności od ilości przydzielonego zasobu. Zwiększenie ilości zasobu dla danego zadania powoduje zwiększenie kosztu wykonania i zmniejszenie jego czasu. W badanej klasie problemów analizuje się klasyczne problemy sze-

regowania, wyspecyfikowane w trójpolowej notacji Grahama [5], z uwzględnieniem modelu typu koszt/czas. Ze względu na obszerną listę otrzymanych rezultatów nie możemy tutaj przedstawić ich przeglądu i dlatego odsyłamy czytelnika do pracy [15].

Niniejsza praca dotyczy problemu szeregowania zadań na m (m≥2) identycznych maszynach (notowanego jako $P||C_{max}$) z uwzględnieniem modelu zadania typu koszt/czas. Wyniki początkowych badań dla tego problemu z dwiema maszynami przedstawiono w [9]. Klasyczny problem $P||C_{max}$ ma bardzo istotne zastosowania i jest jednocześnie stosunkowo łatwym problemem szeregowania. Dlatego też był on intensywnie badany i otrzymano szereg istotnych rezultatów w postaci algorytmów aproksymacyjnych lub tzw. wielomianowych schematów aproksymacyjnych (patrz np. praca przeglądowa [7]).

Badany w pracy problem szeregowania można sformułować następująco: Dany jest zbiór J = {1,2,...,n} zadań, które należy wykonać mając do dyspozycji zbiór M= {1,2,...,m} identycznych maszyn. Każde zadanie może być wykonywane na którejkolwiek z nich bez przerwań. Maszyna w danej chwili czasowej wykonuje co najwyżej jedno zadanie. Dla każdego zadania $j\in J$ określono:

(i) czas wykonywania $p_j=a_j-x_j$, $0\le x_j\le u_j$, gdzie x_j jest szukanym skróceniem normatywnego czasu wykonywania a_j, zaś u_j jest maksymalnym skróceniem; $0\le u_j\le a_j$,

(ii) jednostkowy koszt skrócenia c_j, $c_j\ge 0$.

Oznaczmy przez $x=(x_1,x_2,...,x_n)$ wektor skróceń, a przez $X=(x: 0\le x_j\le u_j, j\in J)$ - zbiór wszystkich dopuszczalnych wektorów skróceń. Przyjmijmy, że I, $I=(I_1,I_2,...,I_m)\in\mathbf{I}$ określa pewien przydział zadań do maszyn, będący podziałem J na m rozłącznych podzbiorów, zaś $\mathbf{I}$ - zbiór wszystkich takich przydziałów; zbiór $I_i\subseteq J$ oznacza zadania wykonywane na maszynie i, $i\in M$. Niech $C_{max}(x,I)$ będzie czasem wykonania wszystkich zadań, przy ustalonym przydziale $I\in\mathbf{I}$ zadań do maszyn oraz czasach wykonywania zadań $p_j=a_j-x_j$, $j\in J$, $x\in X$. Zachodzi

$$C_{max}(x,I) = \max_{i\in M} \{\sum_{j\in I}(a_j-x_j)\} \qquad (1)$$

Łączny koszt uszeregowania definiujemy jako $K(x,I) = cC_{max}(x,I) + \Sigma_{j\in J}c_jx_j$, gdzie c, c>0, jest jednostkowym kosztem pracy systemu, który bez straty ogólności może być przyjęty jako równy jeden. Ostatecznie w badanym problemie (P1) poszukujemy przydziału zadań do maszyn $I^*\in\mathbf{I}$ oraz wektora skróceń $x^*\in X$ takich, że

$$K(x^*,I^*) = \min\{K(x,I): x\in X,\ I\in\mathbf{I}\}. \qquad (2)$$

Z faktu, iż funkcja $C_{max}(x,I)$ $(\Sigma_{j\in J}c_jx_j)$ traktowana jako funkcja jednej zmiennej x_j, przy ustalonych pozostałych zmiennych jest nierosnąca (niemalejąca) wynika, że wystarczy ograniczyć badania do sytuacji $c_j<1$, $j\in J$. W przypadku gdy dla pewnego $j\in J$, $c_j\ge 1$, można położyć $u_j:=0$ oraz $c_j:=0$.

Możliwe są inne sformułowania badanego problemu, [15]. Znaleźć $I^*\in\mathbf{I}$ oraz $x^*\in X$ takie, że $\Sigma_{j\in J}c_jx_j^* \le N$ oraz

$$C_{max}(x^*,I^*) = \min\{C_{max}(x,I): x\in X,\ I\in\mathbf{I},\ \Sigma_{j\in J}c_jx_j \le N\}, \qquad (3)$$

gdzie N, N>0, jest górnym poziomem kosztów, które wynikają z uwzględnienia dodatkowego zasobu (lub jest wprost ilością tego zasobu). (P2). W sformułowaniu "dualnym" (P3) poszukujemy $I^*\in\mathbf{I}$ oraz $x^*\in X$ takich, że $C_{max}(x^*,I^*) \le \tau$ oraz

$$\Sigma_{j\in J}c_jx_j^* = \min\{\Sigma_{j\in J}c_jx_j : x\in X,\ I\in\mathbf{I},\ C_{max}(x,I) \le \tau\}, \qquad (4)$$

gdzie τ, $\tau>0$, jest dopuszczalnym czasem wykonania wszystkich zadań. Ostatecznie, można też sformułować zadanie dwukryterialne (P4) poszukiwania zbioru (lub jego aproksymacji) wszystkich punktów efektywnych (Pareto-optymalnych) w przestrzeni $(C_{max}(x,I), \Sigma_{j\in J}c_jx_j)$. Zauważmy, że każda para $(C_{max}(x^*,I^*), \Sigma_{j\in J}c_jx_j^*)$ będąca rozwiązaniem problemu (P1),(P2) lub (P3) jest punktem efektywnym. Przykładowo, zmieniając wartość c w problemie (P1) lub wartość N w problemie (P2), generujemy różne punkty efektywne. Ostatecznego wyboru (z otrzymanych rozwiązań) dokonuje użytkownik, biorąc pod uwagę np. górny poziom kosztów, dopuszczalny czas wykonania itp. Dalej będziemy rozważać tylko problemy (P1)-(P3), a w szczególności problem (P1).

Zauważmy, że problemy (P1)-(P4) są NP-trudne. Wynika to natychmiast z faktu, że klasyczny problem $P||C_{max}$ jest NP-trudny. Stąd i z potrzeby dysponowania algorytmami o sensownym czasie obliczeń skupimy się tylko nad konstrukcją algorytmów aproksymacyjnych. Dokładność algorytmu aproksymacyjnego H, dla pewnego ustalonego problemu konkretnego, będziemy oceniać za pomocą ilorazu $F(y^H)/F(y^*)$, gdzie F jest funkcją celu, a y^* i y^H odpowiednio rozwiązaniem optymalnym i rozwiązaniem produkowanym przez algorytm H. Dokładność algorytmu H dla dowolnego problemu konkretnego określa parametr ρ^H, [1], [3], $\rho^H=\inf\{\eta: F(y^H)/F(y^*) \le \eta$ dla każdego problemu konkretnego$\}$, zwany przez nas współczynnikiem najgorszego przypadku.

2. Algorytmy aproksymacyjne

Większość algorytmów aproksymacyjnych dla klasy problemów szeregowania, wyspecyfikowanych w trójpolowej notacji Grahama, z uwzględnieniem modelu zadania typu koszt/czas, bazuje na następującej obserwacji. Zmienne decyzyjne dzielą się na dwie grupy: zmienne dyskretne (permutacja, podział zbioru itp.) oraz zmienne ciągłe (wektor skróceń). Ustalając jedną z nich otrzymujemy klasyczny problem szeregowania lub zadanie programowania liniowego. Stąd ogólny schemat algorytmu polega na tym, że wybieramy początkowy wektor skróceń, rozwiązujemy odpowiedni klasyczny problem szeregowania wykorzystując istniejące algorytmy dokładne lub aproksymacyjne i następnie, bazując na znalezionej wartości zmiennej dyskretnej, rozwiązujemy zadanie programowania liniowego. Cały proces może być iterowany. Jakość otrzymanego algorytmu aproksymacyjnego zależy oczywiście w istotny sposób od wyboru początkowego wektora skróceń - tym bardziej że często iterowanie postępowania nie polepsza rozwiązania.

Bazując na wyborze początkowego wektora skróceń, zaproponowanego w pracy [10], dla maszynowego permutacyjnego problemu przepływowego ze zmiennymi czasami wykonywania zadań (dla problemu (P1)), proponujemy wybór następujący. Niech

$$x_j^0 = \max\{\min\{\frac{1+\alpha(m-1)}{\alpha m} - \frac{c_j}{\alpha}, 1\}, 0\}u_j, \quad j\in J, \tag{5}$$

gdzie

$$\alpha = 1 - \frac{\rho m}{[\rho+\sqrt{\rho(m-1)}]^2}. \tag{6}$$

oraz ρ jest współczynnikiem najgorszego przypadku algorytmu rozwiązującego problem $P||C_{max}$. W konsekwencji dla problemu (P1) otrzymujemy całą rodzinę algorytmów aproksymacyjnych H(A), w zależności od algorytmu A rozwiązującego problem $P||C_{max}$.

Algorytm H(A)

Krok 1. Wyznacz przydział $I^{H(A)} \in \mathbf{I}$ dla problemu $P||C_{max}$, z czasami wykonywania zadań $p_j^o = a_j - x_j^o$, $j \in J$, stosując pewien algorytm A o współczynniku najgorszego przypadku równym ρ^A.

Krok 2. Wylicz wektor skróceń $x^{H(A)} \in X$ minimalizujący $K(x, I^{H(A)})$, przy $x \in X$.

Powyższy algorytm można także zastosować do rozwiązania problemów (P2)-(P3). Krok 1 w obu wypadkach nie ulega zmianie. Krok 2 dla problemu (P2) ma postać: wylicz wektor skróceń $x^{H(A)} \in X$ minimalizujący $C_{max}(x, I^{H(A)})$, przy ograniczeniach $\Sigma_{j \in J} c_j x_j \leq N$. Podobnie Krok 2 dla problemu (P3) jest następujący: wylicz wektor skróceń $x^{H(A)} \in X$ minimalizujący $\Sigma_{j \in J} c_j x_j$ przy ograniczeniach $C_{max}(x, I^{H(A)}) \leq \tau$. W celu nie wprowadzania nadmiernej notacji świadomie określiliśmy tymi samymi zmiennymi rozwiązania dla różnych problemów.

W kroku 1 algorytmu H(A) proponujemy wykorzystać (jako algorytm A) wielomianowy, listowy algorytm Grahama, [5], z $\rho^G = 4/3 - 1/(3m)$ lub inne algorytmy (wyczerpujący przegląd w pracy przeglądowej [7]).

Zadanie optymalizacyjne, które należy rozwiązać w Kroku 2, można przedstawić w postaci następującego zadania programowania liniowego (dla problemu (P1)):

$$\min \{T + \Sigma_{j \in J} c_j x_j : T + \Sigma_{j \in I_i} x_j \geq \Sigma_{j \in I_i} a_j,\ i \in M,\ x_j \leq u_j,\ x_j \geq 0,\ j \in J,\ T \geq 0\},$$

gdzie $I_i = I_i^{H(A)}$, $i \in M$. Zadanie to ma n+1 zmiennych i n+m istotnych ograniczeń. Można je rozwiązać stosując standardowe procedury (np. metodę simplex). Jednakże ze względu na jego specyfikę przedstawiamy poniżej algorytm zachłanny AZM, który rozwiązuje je w czasie $O(n \max(m, \log n))$. W celu opisu głównej idei tego algorytmu wprowadzimy pojęcie maszyny *krytycznej*. Maszynę k, $k \in M$, będziemy nazywać maszyną krytyczną przy pewnym ustalonym $x \in X$, jeżeli $\Sigma_{j \in I_k} a_j - x_j = \max_{i \in M} \{\Sigma_{j \in I_i} a_j - x_j\}$. Na początku algorytmu przyjmujemy, że $x_j = 0$, $j \in J$. W każdej głównej iteracji skracamy, o jednakową wielkość Δ, czas wykonywania jednego zadania z każdej maszyny krytycznej; do skracania wybieramy zadanie o najmniejszym jednostkowym koszcie skróceń wśród zadań na danej maszynie, których czasy wykonywania można jeszcze skracać. Iteracja jest wykonywana tylko wtedy, gdy sumaryczny, jednostkowy koszt skróceń wybranych zadań jest mniejszy niż jeden (w przeciwnym wypadku skracanie jest nieopłacalne i algorytm kończy pracę). Wielkość Δ wyznaczona jest w oparciu o dwa warunki: skracamy tak długo aż jedno ze skracanych zadań zostanie skrócone całkowicie lub pojawią się nowe maszyny krytyczne. W opisie algorytmu wykorzystano oznaczenia:

T_i - $T_i = \Sigma_{j \in I_i} a_j$; dla uproszczenia opisu przyjmujemy, że $T_1 \geq T_2 \geq \ldots \geq T_m$.

r - liczba różnych wartości T_i, $i \in M$.

s_i - maksymalny numer maszyny, dla której $T_{s_{i-1}+1} = T_{s_i}$, $i = 1, \ldots, r$, $s_0 = 0$;

$$T_1 = T_2 = \ldots = T_{s_1} > T_{s_1+1} = T_{s_1+2} = \ldots = T_{s_2} > \ldots > T_{s_{r-1}+1} = T_{s_{r-1}+2} = \ldots = T_{s_r} = T_m.$$

A_i - zbiór zadań na maszynie i, które można skracać, tzn. $u_j > 0$ dla $j \in A_i$; $A_i \subseteq I_i$.

z_i - liczność zbioru A_i.

π_i - permutacja zadań ze zbioru A_i wg niemalejących jednostkowych kosztów skróceń: $A_i = \{\pi_i(1), \ldots, \pi_i(z_i)\}$, $c_{\pi_i(k)} \leq c_{\pi_i(k+1)}$, $k = 1, \ldots, z_i - 1$.

k_i - indeks aktualnie analizowanego zadania w permutacji π_i.

Δ_1 - dopuszczalne skrócenie czasu wykonywania zadań $\pi_i(k_i)$, $i=1,2,...,s_t$ wynikające z ich aktualnych maksymalnych skróceń, $\Delta_1 = \min_{1\le i\le s_t}\{u_{\pi_i(k_i)}\}$, $t=1,...,r$; u_j jest modyfikowane w trakcie działania algorytmu,

Δ_2 - dopuszczalne skrócenie czasu wykonywania zadań $\pi_i(k_i)$, $i=1,2,...,s_t$ wynikające z pojawienia się nowych maszyn krytycznych, $\Delta_2 := T_{s_t} - T_{s_{t+1}}$, $t=1,2,...,r$; $T_{s_{r+1}}=0$,

Δ - skrócenie czasu wykonywania zadań w jednej iteracji $\Delta := \min\{\Delta_1, \Delta_2\}$.

```
   procedure AZM;
 1 begin
 2   for i:=1 to m do
 3   begin
 4     T_i:=Σ_{j∈I_i} a_j; A_i:={j∈I_i; u_j>0}; z_i:=|A_i|; k_i:=min(z_i,1);
 5     if k_i = 1 then wyznacz permutacje π_i zadań ze zbioru A_i wg niemalejących
                      jednostkowych kosztów skróceń;
 6   end; (* zakładamy, że T_1≥T_2≥...≥T_m *)
 7   r:=0; s_0:=0; T_{m+1}:=0;
 8   for i:=1 to m do if T_i > T_{i+1} then begin r:=r+1; s_r:=i end; s_{r+1}:=m+1;
 9   for t:=1 to r do
10   begin
11     for i:=s_{t-1}+1 to s_t do if k_i=0 then STOP;
12     repeat
13       if Σ_{i=1}^{s_t} c_{π_i(k_i)} ≥ 1 then STOP;
14       Δ_1:= min_{1≤i≤s_t} {u_{π_i(k_i)}}; Δ_2:=T_{s_t} - T_{s_{t+1}} ; Δ:=min{Δ_1,Δ_2};
15       for i:=1 to s_t do begin x_{π_i(k_i)}:=x_{π_i(k_i)}+Δ; u_{π_i(k_i)}:=u_{π_i(k_i)}-Δ; T_i:=T_i-Δ end;
16       for i:=1 to s_t do if u_{π_i(k_i)} = 0 then begin k_i:=k_i+1; if k_i > z_i then STOP end;
17     until Δ_1 ≥ Δ_2;
18   end;
19 end;
```

Linie (2)-(8) w powyższej procedurze inicjują obliczenia. Linie (13)-(16) określają główną iterację algorytmu. Warunek stopu w linii (11) zachodzi wtedy, gdy na danej (aktualnie krytycznej) maszynie żadne zadanie nie może być skracane; tzn. początkowe maksymalne skrócenia u_j są równe zero. Z kolei warunek stopu z linii (16) zajdzie wtedy, gdy wszystkie zadania z pewnej (aktualnie krytycznej) maszyny zostały skrócone całkowicie. Złożoność obliczeniowa linii inicjujących jest O(nlogn). Główna iteracja algorytmu (linie (13)-(16)) ma złożoność O(m). Ponieważ w głównej iteracji jedno zadanie jest skracane całkowicie ($\Delta=\Delta_1$) lub pojawia się nowa maszyna krytyczna ($\Delta=\Delta_2$), to liczba tych iteracji nie przekracza n+m. Stąd złożoność całego algorytmu jest O(n max{m,logn}).

Procedura AZM może być także zastosowana do problemu (P2). Wtedy linie (13)-(15) należy zastąpić liniami (13')-(15'):

13' if N = 0 then *STOP*;

14' $\Delta_1 := \min_{1 \le i \le s_t} \{u_{\pi_i(k_i)}\}$; $\Delta_2 := T_{s_t} - T_{s_{t+1}}$; $\Delta_3 := N / \sum_{i=1}^{s_t} c_{\pi_i(k_i)}$; $\Delta := \min\{\Delta_1, \Delta_2, \Delta_3\}$;

15' for i:=1 to s_t do

begin $x_{\pi_i(k_i)} := x_{\pi_i(k_i)} + \Delta$; $u_{\pi_i(k_i)} := u_{\pi_i(k_i)} - \Delta$; $T_i := T_i - \Delta$ end; $N := N - \Delta \sum_{i=1}^{s_t} c_{\pi_i(k_i)}$;

Podobnie, w przypadku problemu (P3) linie (13)-(14) należy zastąpić liniami (13")-(14"):

13" if $T_1 \le \tau$ then *STOP*;

14" $\Delta_1 := \min_{1 \le i \le s_t} \{u_{\pi_i(k_i)}\}$; $\Delta_2 := T_{s_t} - T_{s_{t+1}}$; $\Delta_3 := T_1 - \tau$; $\Delta := \min\{\Delta_1, \Delta_2, \Delta_3\}$;

Jeżeli algorytm zakończy działanie oraz $T_1 = \max_{i \in M}\{T_i\} > \tau$, to dla danego przydziału $I^{H(A)}$, problem (P3) nie ma rozwiązania dopuszczalnego. Nie znaczy to, że problem (P3) nie ma w ogóle rozwiązania dopuszczalnego; dla innych przydziałów takie rozwiązanie może istnieć.

3. Analiza najgorszego przypadku

W tym rozdziale przedstawimy wstępne wyniki dotyczące analizy najgorszego przypadku dla algorytmów rozwiązujących problemy (P1)-(P2).

Po pierwsze, zauważmy, że dla algorytmu L1 (dla problemu P1) polegającego na arbitralnym wyborze początkowego wektora skróceń $x^0 \in X$, dokładnym rozwiązaniu zadania $P||C_{max}$ dla $p_j = a_j - x_j^0$, $j \in J$ (tzn. wyznaczeniu $I^{L1} \in \mathbf{I}$ minimalizującego $C_{max}(x^0, I)$) i następnie wyliczeniu $x^{L1} \in X$ minimalizującego $K(x, I^{L1})$, zachodzi

$$K(x^{L1}, I^{L1}) = \min_{x \in X} K(x, I^{L1}) \le K(x^*, I^{L1}) = C_{max}(x^*, I^{L1}) + \Sigma_{j \in J} c_j x_j^* \le m C_{max}(x^*, I^*) + \Sigma_{j \in J} c_j x_j^* \le m C_{max}(x^*, I^*) + \Sigma_{j \in J} c_j x_j^* = m K(x^*, I^*). \quad (7)$$

Podobnie dla analogicznego algorytmu L2 dla problemu (P2) mamy

$$C_{max}(x^{L2}, I^{L2}) \le C_{max}(x^*, I^{L2}) \le m C_{max}(x^*, I^*). \quad (8)$$

Co więcej, ograniczenia (7)-(8) są osiągalne. Rzeczywiście, rozważmy następujący problem konkretny: $n = m^2$, $a_j = 1$, $u_j = 0$, $c_j = 0$, $j = 1,...,m$; $a_j = u_j = 1$, $c_j = \varepsilon$, $j = m+1,...,m^2$, gdzie $\varepsilon > 0$ jest dowolnie małą liczbą. Niech $x_j^0 = 0$, $j \in J$. Wtedy $I_i^{L1} = \{(i-1)m+1, (i-1)m+2, ..., im\}$, $i \in M$. Stąd $x_j^{L1} = 0$, $j \in J$, oraz $K(x^{L1}, I^{L1}) = m$. Łatwo sprawdzić, że $I_i^* = \{i, im-i+2, im-i+3, ..., (i+1)m-i\}$, $i \in M$; $x_j^* = 0$, $j = 1,...,m$, $x_j^* = u_j = 1$, $j = m+1,...,m^2$ oraz $K(x^*, I^*) = 1 + \varepsilon m(m-1)$. Stąd ostatecznie $K(x^{L1}, I^{L1}) / K(x^*, I^*) \to m$, gdy $\varepsilon \to 0$. Analogiczny rezultat otrzymujemy dla algorytmu L2 rozwiązującego problem P2 kładąc np. $N = 1$. Z powyższej analizy wynika, że współczynnik najgorszego przypadku $\rho^{L1} = m$ oraz że współczynnik najgorszego przypadku dla algorytmu H(A) jest nie większy niż m (niezależnie od algorytmu A). Podobna własność zachodzi także dla problemu (P2). Otrzymany rezultat świadczy o skali trudności problemów (P1)-(P2) i potwierdza jeszcze raz tezę o konieczności odpowiedniego wyboru początkowego wektora skróceń x^0. Zauważmy tutaj tylko, że współczynnik najgorszego przypadku dowolnego

algorytmu listowego, rozwiązującego problem $P||C_{max}$, jest nie większy niż 2-1/m, [7].

Przeprowadzimy teraz analizę najgorszego przypadku algorytmu H(A), ograniczając się do problemu (P1). Niech $r = \rho + \rho(m-\rho)/[2\rho+2\sqrt{\rho(m-1)}-1]$, gdzie $\rho=\rho^A$ jest współczynnikiem najgorszego przypadku algorytmu A rozwiązującego problem $P||C_{max}$. Łatwo sprawdzić, że $r=(\rho+\sqrt{\rho(m-1)})/2 + 1/4 + O(1/\sqrt{m})$.

Twierdzenie

$$K(x^{H(A)},I^{H(A)})/K(x^*,I^*) \le r.$$

Dowód twierdzenia można przeprowadzić podobnie jak [10]. Zauważmy, że dla $\rho=1$ (algorytm dokładny dla problemu $P||C_{max}$) $r=\sqrt{(m-1)}/2 + 3/4 + O(1/\sqrt{m})$, zaś dla $\rho=4/3-1/(3m)$ (algorytm Grahama) $r = \sqrt{m-1}/\sqrt{3} + 11/12 + O(1/\sqrt{m})$.

Pokażemy teraz, że ograniczenie z twierdzenia jest w przybliżeniu osiągalne dla $\rho=1$. W tym celu rozważmy następujący przykład konkretny problemu (P1): $n=m=k^2+1$, $a_1=u_1=k$, $c_1=1/(k+1)$, $a_j=u_j=1$, $c_j=1/(k+1)$, $j=2,3,...,n$, gdzie $k\ge2$ jest dowolną liczbą naturalną. Zachodzi $a_1-x_1^o=k(a_j-x_j^o)$, $j=2,...,n$ (niezależnie od wartości współczynnika przy u_j w (5), który zależy od c_j, algorytmu A i od m). Stąd możemy przyjąć, że $I_1^{H(A)} = \{1\}$, $I_i^{H(A)} = \{(i-2)k+2, (i-2)k+3,..., (i-1)k+1\}$, $i=2,3,...,k+1$, $I_i^{H(A)}=\emptyset$, $i=k+2,...,m$, $x_j^{H(A)}=0$, $j\in J$ oraz $K(x^{H(A)},I^{H(A)})=k$. Z kolei $I_i^*=\{i\}$, $i\in M$, $x_1^*=k-1$, $x_j^*=0$, $j=2,3,...,n$ i minimalna wartość $K(x^*,I^*)= 1+(k-1)/(k+1)$. Ostatecznie $K(x^{H(A)},I^{H(A)})/K(x^*,I^*) = (k+1)/2 = \sqrt{m-1}/2 +1/2$.

Z powyższej analizy wynika, że odpowiedni wybór wektora x^o spowodował zmniejszenie wartości współczynnika najgorszego przypadku z m na $\sqrt{m}/2$ w przybliżeniu. Pomimo, iż jest to jeszcze bardzo duży błąd względny, rezultaty analizy eksperymentalnej są obiecujące. Wydaje się, że zachodzi tutaj podobna sytuacja jak dla klasycznego problemu permutacyjnego $F||C_{max}$, w którym algorytmy aproksymacyjne o współczynnikach rzędu m/2 lub nawet $m/\sqrt{2}$ (patrz [11]-[13]) produkują uszeregowania bliskie optymalnym . W badaniach eksperymentalnych, których szczegółowo nie będziemy opisywać, na kilka tysięcy generowanych przykładów średni błąd względny był rzędu 1%. W badaniach tych jako algorytm A wykorzystano algorytm Grahama. Do wyliczenia dolnego ograniczenia minimalnej wartości łącznego kosztu uszeregowania $K(x^*,I^*)$ dopuszczono możliwość przerywania zadań (patrz rozdział 4). Konkludując, algorytm H(A) może być polecany w zastosowaniach praktycznych.

4. Zadania podzielne

Rozważymy teraz problemy (P1)-(P4) przy założeniu upraszczającym, polegającym na dopuszczeniu możliwości przerywania zadań w trakcie ich wykonywania. Dla ustalonego $x\in X$ uszeregowanie jest określone przez zestaw czwórek (j,i,s,t), gdzie j jest numerem zadania, a i maszyną, na której to zadanie jest wykonywane w przedziale czasowym (s,t). Niech s(x) określa uszeregowanie dopuszczalne, a S(x) - zbiór wszystkich uszeregowań dopuszczalnych, dla ustalonego $x\in X$. W celu precyzyjnego zdefiniowania problemów (P1)-(P4) wystarczy zamienić I oraz $\mathbb{I}$ przez s(x) oraz S(x), odpowiednio. Po tej modyfikacji będziemy je nazywać problemami (P1P)-(P4P). Dla uszeregowań zwartych $s(x)\in S(x)$, $x\in X$ zachodzi, [7],

$$C_{max}(x,s(x)) = \max\{\frac{1}{m}\sum_{j\in J}(a_j-x_j),\ \max_{j\in J}(a_j-x_j)\}. \qquad (9)$$

Znając wartość $C_{max}(x,s(x))$ można wyznaczyć uszeregowanie s(x), z co najwyżej m-1 przerwaniami, wykorzystując algorytm Mc Naughtona, [1].

Dopuszczenie podzielności zadań radykalnie upraszcza omawiane problemy. Dla najbardziej ogólnego z nich, tzn. dla (P4P), podamy teraz algorytm dokładny o złożoności wielomianowej. Oznaczmy przez P zbiór wszystkich punków efektywnych w przestrzeni (C,K), gdzie $C = C_{max}(x,s(x))$, $K = \Sigma_{j\in J} c_j x_j$, $s(x)\in S(x)$, $x\in X$. Niech $\tau_1 = \max\{(1/m)\Sigma_{j\in J}(a_j-u_j), \max_{j\in J}\{a_j-u_j\}\}$, $\tau_2 = \max\{(1/m)\Sigma_{j\in J}a_j, \max_{j\in J}\{a_j\}\}$. Łatwo pokazać, uwzględniając (9), że

$$\{(t,g(t)): \tau_1 \le t \le \tau_2\} = P,$$

gdzie

$$g(t) = \min\{\Sigma_{j\in J} c_j x_j: (1/m)\Sigma_{j\in J}(a_j-x_j)\le t,\ a_j-x_j\le t,\ 0\le x_j\le u_j,\ j\in J\}.$$

Z postaci funkcji g wynika, że zbiór P jest wypukłą łamaną z pewną liczbą punktów wierzchołkowych (C^k,K^k), k= 1,2,...,r. Punkty te wyznacza poniższa procedura zachłanna AZP.

```
procedure AZP;
1  begin
2    r:=1; C^r:=τ2; K^r:=0; x_j:=0, j∈J;
3    while A:={j∈J: u_j>0, j∈J} <> ∅ then
4    begin
5      T1:=(1/m)Σ_{j∈J}(a_j-x_j); T2:=max_{j∈J}{a_j-x_j};
6      Z:={j∈J: a_j-x_j=max{T1,T2}}; z:=|Z|; (* Z - zbiór zadań krytycznych *)
7      if T1 < T2 then begin Δ1:=m(T2-T1)/(m-z); d:=0 end
         else begin c_d:=min_{j∈A}{c_j}; if d∈Z then Δ1:=u_d/(m+1-z) else Δ1:=u_d/(m-z) end;
8      Δ2:=min_{j∈Z}{u_j}: if Δ2 = 0 then STOP;
9      if T1 > T2 then Δ3:=T1-max_{j∈J, j≠d}{a_j-x_j} else Δ3:=T2-max{a_j-x_j: a_j-x_j<T2, j∈J, j≠d};
10     Δ:=min{Δ1,Δ2,Δ3};
11     if T1 ≥ T2 then begin x_d:=x_d+Δ(m-z); u_d:=u_d-Δ(m-z) end;
12     x_j:=x_j+Δ; u_j:=u_j-Δ, j∈Z; r:=r+1; C^r:=C^{r-1}-Δ;
13     K^r:=K^{r-1}+ΔΣ_{j∈Z}c_j; if T1 ≥ T2 then K^r:=K^r+Δ(m-z)c_d;
14   end
15 end;
```

W opisie procedury przyjmujemy, że min{∅}=∞, max{∅}=0. W głównej iteracji (linie 5-13) skracamy zadanie d o najmniejszym jednostkowym koszcie skróceń i/lub zadania *krytyczne* ze zbioru Z: zadanie $j\in J$ nazywamy krytycznym, przy pewnym $x\in X$, jeżeli jego czas wykonania jest równy długości uszeregowania tzn. wielkości $\max\{T_1,T_2\}$. Po wykonaniu głównej iteracji długość uszeregowania zmniejsza się o wielkość Δ, wyliczaną w różny sposób, w zależności od relacji między wartościami T_1, T_2 oraz prawdziwości warunku: $d\in Z$.

$(T_2>T_1)$. Skracamy każde zadanie krytyczne o Δ. Skracamy tak długo, aż: $T_1=T_2$ $(\Delta=\Delta_1)$, jedno zostanie skrócone całkowicie $(\Delta=\Delta_2)$ lub pojawią się nowe zadania krytyczne $(\Delta=\Delta_3)$.

($T_2<T_1$). Skracamy zadanie d o $m\Delta$; wtedy zbiór Z jest pusty (z=0). Skracamy tak długo, aż: zadanie d zostanie skrócone całkowicie ($\Delta=\Delta_1$) lub pojawią się zadania krytyczne ($\Delta=\Delta_3$).

($T_2=T_1$, $d\notin Z$). Skracamy każde zadanie krytyczne o Δ oraz zadanie d o $\Delta(m-z)$. Skracamy tak długo, aż: zadanie d zostanie skócone całkowicie ($\Delta=\Delta_1$), jedno z zadań krytycznych zostanie skrócone całkowicie ($\Delta=\Delta_2$) lub pojawią się nowe zadania krytyczne ($\Delta=\Delta_3$).

($T_2=T_1$, $d\in Z$). Skracamy każde zadanie krytyczne (z wyjątkiem d) o Δ oraz d o $\Delta(m-z+1)$. Proces skracania przeprowadzamy tak jak w przypadku poprzednim.

Koszt skrócenia zadań w głównej iteracji wynosi Δc_{sum}, gdzie $c_{sum}=\Sigma_{j\in Z}c_j+(m-z)c_d$, jeżeli $T_2\leq T_1$ lub $c_{sum}=\Sigma_{j\in Z}c_j$ w przeciwnym wypadku. Zauważmy, że liczba tych iteracji jest O(n) i każda ma złożoność O(n). Stąd złożoność procedury jest $O(n^2)$. Zbiór P tworzy suma r-1 odcinków łączących pary punktów (C^k,K^k), (C^{k+1},K^{k+1}), k=1,2,...,r-1. Dysponując tym zbiorem możemy łatwo rozwiązać problemy (P1P)-(P3P). Jednakże, dla (P1P) nie musimy wykonywać procedury AZP do końca. Wystarczy między linie 7 a linie 8 wprowadzić linie 7a:

7a $c_{sum}:=\Sigma_{j\in Z}c_j$; if $T_1 \geq T_2$ then $c_{sum}:=c_{sum}+(m-z)c_d$; if $c_{sum} \geq 1$ then *STOP;*

Podobnie dla (P2P), między linie 4 a 5 należy wprowadzić linię 4a, zaś między linie 10 a 11 linię 10a:

4a if N = 0 then *STOP;*

10a $c_{sum}:=\Sigma_{j\in Z}c_j$; if $T_1 \geq T_2$ then $c_{sum}:=c_{sum}+(m-z)c_d$; $\Delta:=\min\{\Delta,N/c_{sum}\}$; $N:=N-\Delta c_{sum}$;

Z kolei do rozwiązania problemu (P3P) nie musimy wykorzystywać procedury AZP. Jeżeli $\tau<\tau_1$, to (P3P) nie ma rozwiązania. W przypadku przeciwnym poniższa procedura AZP3 rozwiązuje ten problem w czasie O(nlogn).

```
procedure AZP3;
begin
  for j:=1 to n do begin x_j:=max{a_j-τ,0); u_j:=u_j-x_j end; Δ:= Σ_{j∈J}(a_j-x_j)-mτ;
  wyznacz permutacje π zbioru J wg. niemalejących jednostkowych kosztów skróceń;
  for i:=1 to n do
  begin if Δ ≤ 0 then STOP; j:=π(i); x_j:=x_j+min(Δ,u_j); Δ:=Δ-min(Δ,u_j) end;
end;
```

Poprawność działania AZP3 wynika bezpośrednio z definicji g(t) dla $t=\tau$.

LITERATURA

[1] Błażewicz J.: Złożoność obliczeniowa problemów kombinatorycznych. WNT, Warszawa 1988.

[2] Daniels R.L.: A multi-objective approach to resource allocation in single machine scheduling, *European J. Oper. Res.* 48, 1990, 226-241.

[3] Garey, M.R and Johnson D.S.: Computers and intractability. A guide to the theory of NP-completeness, W.H. Freeman and Company, San Francisco 1979.

[4] Janiak, A.: Single machine scheduling problem with a common deadline and resource

dependent release dates, *European J. Oper. Res.* 53, 1991, 317-325.

[5] Graham, R.L., Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: A survey, *Ann. Discrete Math.* 5, 1979, 287-326.

[6] Ishi, H., Martel, C., Masuda, T. and Nishida T.: A generalized uniform processor system, *Oper. Res.* 33, 1985, 346-362.

[7] Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B.: Sequencing and scheduling: Algorithms and complexity. Report BS-R8909, Centre for Mathematics and Computer Science, Amsterdam 1989.

[8] Nowicki E.: Algorytmy aproksymacyjne dla dwumaszynowego problemu przepływowego z wypukłą funkcją kosztu, *Archiwum Automatyki i Telemechaniki* 33, z.3, 1988.

[9] Nowicki E.: Problem szeregowania zadań na równoległych maszynach ze zmiennymi czasami wykonywania, *Zeszyty Naukowe AGH* Nr. 59, 1991, 135-144.

[10] Nowicki E.: An approximation algorithm for the m-machine permutation flow shop scheduling problem with controllable job processing times, *European J. Oper. Res.*, 1992 (w druku).

[11] Nowicki E. and Smutnicki C.: Worst-case analysis of an approximation algorithm for flow-shop scheduling, *Operations Research Letters* 8, 1989, 171-177.

[12] Nowicki E. and Smutnicki C.: Worst-case analysis of Dannenbring's algorithm for flow-shop scheduling. *Operations Research Letters* 10, 1991, 473-480.

[13] Nowicki E. and Smutnicki C.: New results in the worst-case analysis for flow-shop scheduling. *Discrete Applied Mathematics*, 1992 (w druku)

[14] Nowicki, E. and Zdrzałka, S.: Two-machine flow shop scheduling problem with controllable job processing times, *European J. Oper. Res.* 34, 1988, 208-220.

[15] Nowicki, E. and Zdrzałka, S.: A survey of results for sequencing problems with controllable processing times, *Discrete Appl. Math.* 26, 1990, 271-287.

[16] Vickson, R.G.: Choosing the job sequence and processing times to minimize total processing plus flow cost on single machine, *Oper. Res.* 28, 1980, 1155-1167.

[17] Van Wassenhove, L.N. and Baker K.R.: A bicriterion approach to time/ cost tradeoffs in sequencing, *European J. Oper. Res.* 11, 1982, 48-54

Recenzent: Prof.dr h.inż. Andrzej Świerniak
Wpłynęło do Redakcji do 30.04.1992 r.

Abstract:

In the real-life applications of scheduling research, apart from the machines, processing a job requires additional resources. One of the simplest forms of resource allocation is represented by a time/cost model. In this model, the time required to perform a job can be reduced by the application of additional nonrenewable resources (measured by their cost). In consequence, job processing time can be considered as a decision variable. The paper deals with a problem of scheduling on m identical, parallel machines with controllable job processing times. It is assumed that the cost of performing a job is a linear function of its processing time, and the overall schedule cost to be minimized is the total processing cost plus maximum completion time cost. An approximation algorithm for minimizing the overall schedule cost with the worst-case performance ratio equal to $(\rho+\sqrt{\rho+(m-1)})/2 + 1/4 + O(1/\sqrt{m})$ is provided, where ρ is the worst-case performance ratio of a procedure for solving the pure scheduling problem.