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

Adam JANIAK, Piotr SŁONIŃSKI
Politechnika Wrocławska

MINIMALIZACJA SUMARYCZNEJ ILOŚCI ZUŻYTEGO ZASOBU W PROBLEMIE SZEREGOWANIA ZADAŃ O DYNAMICZNYCH MODELACH TERMINÓW DOSTĘPNOŚCI

Streszczenie. W pracy zaprezentowano rozwiązanie problemu minimalizacji sumarycznej ilości zużytego zasobu przy dynamicznych (różniczkowych) modelach terminów dostępności zadań oraz przy ograniczeniu na czas zakończenia wykonywania wszystkich zadań. Wykazano szereg istotnych własności rozważanego problemu (między innymi dotyczących postaci funkcji rozdziału zasobu). Bazując na udowodnionych własnościach podano konstrukcję algorytmu szeregowania zadań na maszynie krytycznej oraz przydział zasobu do terminów dostępności zadań, który zapewnia zakończenie wykonywania zadań w ustalonym czasie minimalizując ilość zużytego na ten cel zasobu.

MINIMIZATION OF TOTAL AMOUNT OF RESOURCE CONSUMED IN THE SCHEDULING PROBLEM WITH DYNAMIC MODELS OF RELEASE DATES

Summary. The aim of this contribution is to present a solution of the problem of minimizing total amount of resource consumed in the resource allocation and scheduling problem with dynamic (differential) models of job release dates. There is also constraint set on the time of finishing processing of all the jobs in this problem. Many important properties of this problem have been proven (e.g. those concerning the character of the resource allocation function). The presented scheduling algorithm is based on the properties proven in this paper. There are also given some generalizations of the considered problem.

1. Wstęp

Klasyczna teoria szeregowania zadań zakłada, że zarówno terminy dostępności, jak i czasy wykonywania zadań są stałe. Termin dostępności zadania może być interpretowany jako pewien wstępny proces, którego czas trwania determinuje moment gotowości

zadania do właściwej obróbki na maszynie krytycznej. W naszej pracy proponujemy przyjęcie dynamicznego (różniczkowego) modelu terminów dostępności zadań. Model ten, w odniesieniu do czasów wykonywania zadań, został wprowadzony przez Burkova [1] oraz wykorzystywany między innymi w badaniach Janiaka [3], Węglarza [6], Józefowskiej [5], Przysady [2]. Stan wspomnianego wstępnego procesu jest opisany równaniem różniczkowym, które określa prędkość zmiany tego procesu w zależności od ilości przydzielonego zasobu ciągłego. Po zakończeniu procesu przygotowawczego (realizowanego na równoległych maszynach) zadania są wykonywane kolejno (według ustalonej permutacji) na pojedynczej maszynie krytycznej. Czas właściwej obróbki zadań na maszynie krytycznej jest stały i znany *a priori*.

Modele dynamiczne mają bardzo szerokie zastosowanie w opisywaniu i interpretacji procesów przemysłowych (np. podgrzewanie wlewków hutniczych w piecach wgłębnych) i informatycznych (np. proces przygotowywania animacji i renderowania obrazów w programach CAD).

W naszej pracy prezentujemy rozwiązanie problemu minimalizacji sumarycznej ilości zużytego zasobu przy dynamicznych modelach terminów dostępności zadań oraz przy ograniczeniu na czas zakończenia wykonywania wszystkich zadań.

Dalsza część pracy jest zorganizowana następująco. Rozdział 2 zawiera dokładne sformułowanie rozważanego problemu. W rozdziale 3 podano szereg istotnych własności rozważanego problemu wraz z ich dowodami. Algorytm wyznaczania kolejności wykonywania zadań znajduje się w rozdziale 4, a uwagi końcowe i możliwości uogólnień zostały zawarte w rozdziale 5.

2. Sformułowanie problemu

Dany jest zbiór n niepodzielnych zadań $J = \{1, 2, \ldots, n\}$. Każde zadanie $j \in J$ przed zasadniczym wykonaniem musi być poddane pewnemu wstępnemu procesowi przygotowawczemu. Długość jego trwania może być interpretowana jako termin dostępności tego zadania.

Wspomniany wstępny proces przygotowawczy jest realizowany na równoległych maszynach poprzedzających maszynę krytyczną, natomiast zasadnicze wykonanie każdego

zadania odbywa się sekwencyjnie na pojedynczej maszynie krytycznej.

Czas trwania procesu przygotowawczego (termin dostępności na maszynie krytycznej) jest dynamicznie zależny od ilości przydzielonego zasobu, natomiast czas zasadniczego wykonywania zadania j na pojedynczej maszynie jest stały i wynosi p_j jednostek czasu ($j = 1, 2, \ldots, n$).

Przyjmujemy następujący dynamiczny model terminu dostępności zadania j:

$$\dot{x}_j^r(t) = \frac{dx_j^r(t)}{dt} = f_j(u_j(t)), \quad x_j^r(0) = 0, \quad (j = 1, 2, \ldots, n) \tag{1}$$

gdzie:

- $x_j^r(t)$ – stan wstępnego procesu determinującego termin dostępności zadania j w chwili t, $x_j^r(0) = 0$ jest stanem początkowym tego procesu;
- $u_j(t)$ – ilość zasobu przydzielona do terminu dostępności zadania j w chwili t;
- $f_j(\cdot)$ – funkcja wklęsła, dodatnia, ciągła, niemalejąca, spełniająca warunek $f_j(0) = 0$.

Każde zadanie j musi osiągnąć w wyniku wspomnianego wstępnego procesu zadany stan końcowy $\hat{x}_j^r$. Dopiero wówczas stanie się dostępne do obróbki na rozpatrywanej maszynie. Innymi słowy zachodzi następująca zależność:

$$\int_0^{r_j} f_j(u_j(t))\, dt = \hat{x}_j^r, \quad (j = 1, 2, \ldots, n). \tag{2}$$

Właściwe wykonywanie zadania j na rozpatrywanej maszynie, trwające p_j jednostek czasu, można rozpocząć po doprowadzeniu terminu dostępności zadania j do stanu końcowego $\hat{x}_j^r$.

Ponadto zadane jest ograniczenie na termin zakończenia wykonywania wszystkich zadań, które oznaczamy $\hat{C}$. Przez r_j oznaczamy (nieznany a priori) termin dostępności zadania j.

Dla ustalonej kolejności wykonywania zadań przez rozdział zasobu będziemy rozumieli odcinkami ciągłą, nieujemną funkcję wektorową:

$$\mathrm{u} = \mathrm{u}(t) = [u_1(t), u_2(t), \ldots, u_n(t)]. \tag{3}$$

Zadane mogą być również technologiczne ograniczenia kolejnościowe $\prec$ pomiędzy zadaniami. Zbiór wszystkich dopuszczalnych kolejności **z** wykonywania zadań oznaczamy Z.

Dla rozpatrywanego tutaj problemu (zapisanego w notacji trójpolowej [4])

$$1\left|\prec, \dot{x}_j^r(t) = f_j(u_j(t)), C_{\max} \leqslant \hat{C}\right| \sum_{j=1}^{n} \int_0^{r_j} u_j(t)\, dt \tag{4}$$

należy znaleźć taką kolejność wykonywania zadań **z** na maszynie krytycznej oraz taki rozdział zasobu **u** do terminów dostępności, aby zminimalizować łączną ilość zużytego zasobu, tj.

$$Q = \sum_{j=1}^{n} \int_0^{r_j} u_j(t)\, dt. \tag{5}$$

Sformułowany problem (4) jest skomplikowanym problemem optymalizacji dynamicznej (modele terminów dostępności zadań) i optymalizacji dyskretnej (szeregowanie zadań). W następnych punktach wykażemy szereg własności rozpatrywanego problemu. Własności te wykorzystamy następnie do skonstruowania przybliżonego algorytmu znajdowania kolejności wykonywania zadań i rozdziału zasobu.

Z rozważań przeprowadzonych w rozdziale 9 książki [4] wynika, że rozpatrywany tutaj problem jest co najmniej tak trudny, jak NP-trudny problem badany we wspomnianej książce.

3. Własności problemu

Poniżej udowodnimy bardzo istotny z punktu widzenia uproszczenia obliczeń fakt, który pozwala na zastąpienie zmiennego w czasie przydziału zasobu do terminu dostępności przydziałem stałym w czasie.

Twierdzenie 1. *Jeżeli w przedziale $[0, r_j]$ istnieje zmienny w czasie przydział zasobu $\bar{u}_j(t)$ do terminu dostępności zadania j, to istnieje w tym przedziale równoważny mu (w sensie funkcji celu) stały w czasie przydział $\tilde{u}_j(t) = \bar{u}_j$ $(j = 1, 2, \ldots, n)$.*

Dowód. *Niech $\bar{u}_j(t)$ oznacza zmienny w czasie przydział zasobu dla terminu dostępności zadania j. Wówczas można go zastąpić stałym w czasie przydziałem zasobu, który można obliczyć następująco:*

$$\bar{u}_j = \frac{\int_0^{r_j} \bar{u}_j(t)\, dt}{r_j}, \quad t \in [0, r_j], \quad (j = 1, 2, \ldots, n). \tag{6}$$

Wykażemy teraz, że obliczony w ten sposób stały w czasie rozdział zasobu nie wpłynie na wartość funkcji celu. Niech zatem Q' oznacza wartość funkcji celu, w której zmienny w czasie przydział zasobu zastąpiono przydziałem stałym w czasie. Wówczas mamy:

$$\begin{aligned} Q' &= \sum_{j=1}^{n} \int_0^{r_j} \bar{u}_j \, dt = \sum_{j=1}^{n} \bar{u}_j \int_0^{r_j} dt = \underline{\sum_{j=1}^{n} \bar{u}_j \cdot r_j} = \\ &= \sum_{j=1}^{n} r_j \cdot \frac{\int_0^{r_j} \tilde{u}_j(t)\, dt}{r_j} = \sum_{j=1}^{n} \int_0^{r_j} \tilde{u}_j(t) \, dt \overset{(5)}{=} Q. \end{aligned} \tag{7}$$

Z równania (7) wynika, że wartość funkcji celu nie uległa zmianie, a więc twierdzenie jest prawdziwe. ■

W dalszych rozważaniach stały przydział zasobu $\tilde{u}_j(t) = \bar{u}_j$ do terminu dostępności zadania j będziemy oznaczali przez u_j.

Zauważmy, że podczas dokonywania przekształceń w równaniu (7) otrzymaliśmy bardzo prostą (i wygodną dla prowadzenia obliczeń) postać funkcji celu:

$$Q = \sum_{j=1}^{n} u_j \cdot r_j, \tag{8}$$

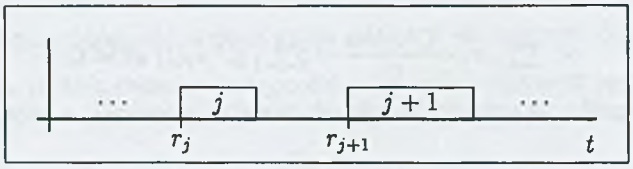
w której u_j jest stałym w czasie przydziałem zasobu obliczonym na podstawie (6), a r_j jest terminem dostępności zadania j.

Poniżej udowodnimy jeszcze jedną istotną własność, która mówi, iż wartość funkcji celu w rozważanym problemie jest minimalna, gdy zadania są wykonywane jedno po drugim bez przerw między nimi. Wykażemy prawdziwość tego twierdzenia przy założeniu, że funkcje modeli terminów dostępności są funkcjami pierwiastkowymi o dodatnich współczynnikach. Twierdzenie to jest prawdziwe dla dowolnych wklęsłych i rosnących funkcji modeli terminów dostępności, ale dowód dla tego przypadku jest znacznie dłuższy i bardziej skomplikowany.

Twierdzenie 2. *W rozpatrywanym problemie (4) funkcja celu Q ma wartość minimalną wtedy i tylko wtedy, gdy zadania są wykonywane jedno po drugim bez przerw.*

Dowód. *Bez straty ogólności możemy rozważyć dwa sąsiednie zadania j i $j+1$, $(j = 1, 2, \ldots, n-1)$. Wykażemy, że ilość zasobu wymagana do osiągnięcia przez termin dostępności zadania j swojego stanu końcowego $\hat{x}_j^r$ jest mniejsza w przypadku, gdy wykonywanie zadania j rozpoczyna się w chwili r'_j $(r'_j + p_j = r_{j+1})$, niż w przypadku, gdy wykonywanie zadania j rozpoczyna się w chwili r_j $(r_j + p_j < r_{j+1})$.*

Załóżmy, że wykonywanie tych zadań na maszynie krytycznej zaczyna się odpowiednio w chwili r_j i r_{j+1}, przy czym $r_j + p_j < r_{j+1}$ (por. rys. 1). Załóżmy ponadto, że do terminu dostępności zadania j przydzielono (na podstawie tw. 1 stałą w całym przedziale $[0, r_j]$ ilość zasobu u_j.



Rys. 1. Przerwa miedzy zadaniami j i $j+1$
Fig. 1. There is idle time between jobs j and $j+1$

Warunek osiągnięcia przez termin dostępności swojego stanu końcowego $\hat{x}_j^r$ (umożliwiającego rozpoczęcie wykonywania zasadniczej części zadania j na maszynie krytycznej) można zapisać zgodnie z warunkiem (2) w następującej postaci:

$$\int_0^{r_j} f_j(u_j)\, dt = \hat{x}_j^r.$$

Stąd po scałkowaniu otrzymujemy zależność:

$$f_j(u_j) \cdot r_j = \hat{x}_j^r, \quad \text{czyli} \quad u_j = f_j^{-1}\left(\frac{\hat{x}_j^r}{r_j}\right). \tag{9}$$

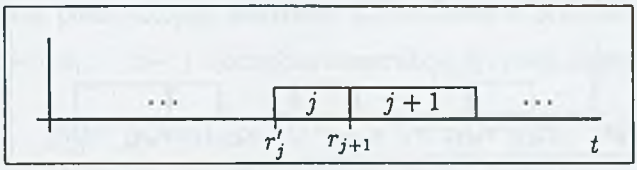
Możemy zatem obliczyć ilość zasobu koniecznego do osiągnięcia przez termin dostępności zadania j swojego stanu końcowego $\hat{x}_j^r$ w czasie r_j:

$$u_j \cdot r_j = f_j^{-1}\left(\frac{\hat{x}_j^r}{r_j}\right) \cdot r_j.$$

Rozważmy teraz przypadek, w którym wykonywanie zadania j na maszynie krytycznej zaczyna się o $r_{j+1} - r_j - p_j$ jednostek czasu później (a więc nie ma przerwy między wykonywaniem zadania j i $j+1$ (por. rys. 2)).

W tym przypadku rozpoczęcie wykonywania zadania j następuje w chwili r'_j ($r'_j > r_j$), a do terminu dostępności tego zadania przydzielono (na podstawie tw. 1) stałą w przedziale $[0, r'_j]$ ilość zasobu u'_j.

Na podstawie analogicznych do przedstawionych rozważań obliczymy ilość zasobu konieczną do osiągnięcia przez termin dostępności zadania j w przedziale $[0, r'_j]$ stanu koń-



Rys. 2. Zadania j i $j+1$ wykonywane bez przerwy
Fig. 2. There is no idle time between jobs j and $j+1$

cowego $\hat{x}^r_j$*:*

$$u'_j \cdot r'_j = f_j^{-1}\left(\frac{\hat{x}^r_j}{r'_j}\right) \cdot r'_j.$$

Warunek konieczny i wystarczający podany w tezie dowodzonego twierdzenia można zapisać w postaci następującej nierówności:

$$u_j \cdot r_j \geqslant u'_j \cdot r'_j, \quad (j = 1, 2, \ldots, n)$$

czyli:

$$f_j^{-1}\left(\frac{\hat{x}^r_j}{r_j}\right) \cdot r_j \geqslant f_j^{-1}\left(\frac{\hat{x}^r_j}{r'_j}\right) \cdot r'_j. \tag{10}$$

W celu uproszczenia dowodu nierówności (10) nasze rozważania zawęzimy do klasy funkcji potęgowych $f_j(u) = \beta_j u^{1/\alpha_j}$, $\alpha_j \geqslant 1, \beta_j > 0$ *(funkcje te mają bardzo szerokie zastosowania praktyczne w modelowaniu i interpretacji procesów przemysłowych). Stąd łatwo obliczyć postać funkcji odwrotnej* $f_j^{-1}(v) = \left(\frac{v}{\beta_j}\right)^{\alpha_j}$, *gdzie* v *należy do przeciwdziedziny funkcji* $f(\cdot)$. *Nierówność (10) możemy zatem zapisać w następującej postaci:*

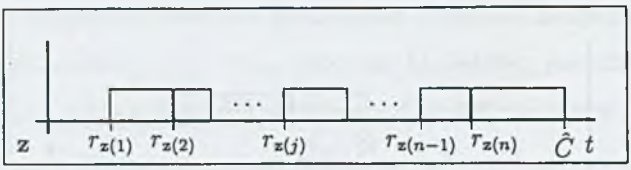
$$\left(\frac{\hat{x}^r_j}{\beta_j r_j}\right)^{\alpha_j} \cdot r_j \geqslant \left(\frac{\hat{x}^r_j}{\beta_j r'_j}\right)^{\alpha_j} \cdot r'_j.$$

Dzieląc obydwie strony tej nierówności przez $\left(\frac{\hat{x}^r_j}{\beta_j}\right)^{\alpha_j}$ *otrzymujemy:*

$$\frac{1}{r_j^{\alpha_j - 1}} \geqslant \frac{1}{\left(r'_j\right)^{\alpha_j - 1}} \Longleftrightarrow \left(r'_j\right)^{\alpha_j - 1} \geqslant r_j^{\alpha_j - 1}.$$

Ostatnia nierówność jest oczywista, ponieważ $r'_j > r_j$. ■

Zauważmy, że na podstawie twierdzenia 2 można wyznaczyć terminy dostępności oraz optymalny przydział zasobu do terminów dostępności zadań w ustalonej permutacji $z \in Z$ (por. rys. 3).



Rys. 3. Zadania wykonywane jedno po drugim bez przerwy
Fig. 3. The jobs are processed one-by-one – no idle time between jobs

Z twierdzenia 2 wynika bowiem, że czas zakończenia wykonywania ostatniego zadania wynosi $\hat{C}$, a zatem termin dostępności poszczególnych zadań można obliczyć ze wzoru:

$$r_{z(j)} = \hat{C} - \sum_{i=j}^{n} p_{z(i)}.$$

Natomiast stały w czasie przydział zasobu do terminów dostępności zadań wynosi:

$$u_{z(j)} = f_{z(j)}^{-1}\left(\frac{\hat{x}_{z(j)}^{r}}{\hat{C} - \sum_{i=j}^{n} p_{z(i)}}\right).$$

Zatem przy ustalonej kolejności wykonywania zadań minimalna wartość funkcji celu wynosi:

$$Q(z) = \sum_{j=1}^{n} u_{z(j)} \cdot r_{z(j)} = \sum_{j=1}^{n} f_{z(j)}^{-1}\left(\frac{\hat{x}_{z(j)}^{r}}{\hat{C} - \sum_{i=j}^{n} p_{z(i)}}\right) \cdot \left(\hat{C} - \sum_{i=j}^{n} p_{z(i)}\right).$$

4. Algorytm przybliżony wyznaczania kolejności wykonywania zadań

W niniejszym rozdziale podamy przybliżony algorytm wyznaczania permutacji wykonywania zadań w rozważanym problemie. W tym celu rozważymy wartości funkcji celu dla permutacji różniących się kolejnością dwóch sąsiednich zadań. Na zakończenie podamy (zapisany w pseudo-Pascalu) algorytm przybliżony wyznaczania permutacji wykonywania zadań.

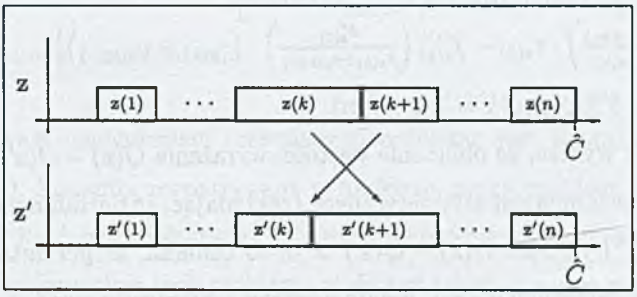
Załóżmy na początek, że zadania są wykonywane w permutacji $z \in Z$. Wówczas, na podstawie poprzednich twierdzeń i spostrzeżeń, wartość funkcji celu wynosi:

$$Q(z) = \sum_{j=1}^{n} u_{z(j)} \cdot r_{z(j)},$$

gdzie $u_{z(j)}$ oznacza stały w czasie przydział zasobu dla zadania j w permutacji z, a $r_{z(j)}$ oznacza termin dostępności zadania j w permutacji z.

Rozważmy teraz permutację z′ otrzymaną z permutacji z przez zamianę miejscami zadań k i $k+1$ ($k = 1, 2, \ldots, n-1$). Formalnie konstrukcję tę opisuje następująca zależność:

$$\begin{aligned} z'(j) &= z(j) \qquad (j = 1, \ldots, k-1, k+2, \ldots, n), \\ z'(k) &= z(k+1), \\ z'(k+1) &= z(k). \end{aligned}$$



Rys. 4. Zamiana zadań k i $k+1$
Fig. 4. Swapping the jobs k and $k+1$

Wówczas wartość funkcji celu wynosi:

$$Q(z') = \sum_{j=1}^{n} u'_{z'(j)} \cdot r'_{z'(j)},$$

przy czym:

$$\left.\begin{aligned} r'_{z'(j)} &= r_{z(j)} \\ u'_{z'(j)} &= u_{z(j)} \end{aligned}\right\} (j = 1, \ldots, k-1, k+2, \ldots, n). \tag{11}$$

Ponadto z rys. 4 wynika, że:

$$\begin{aligned} r'_{z'(k)} &= r_{z(k)}, \\ u'_{z'(k)} &= f^{-1}_{z(k+1)}\left(\frac{\hat{x}^r_{z(k+1)}}{r_{z(k)}}\right), \\ r'_{z'(k+1)} &= r_{z(k)} + p_{z(k+1)}, \\ u'_{z'(k+1)} &= f^{-1}_{z(k)}\left(\frac{\hat{x}^r_{z(k)}}{r_{z(k)} + p_{z(k+1)}}\right). \end{aligned} \tag{12}$$

Aby sprawdzić, która permutacja (z′ czy z) odpowiada mniejszej wartości funkcji celu,

wystarczy zbadać znak różnicy $Q(\mathrm{z}) - Q(\mathrm{z}')$:

$$
\begin{aligned}
& Q(\mathrm{z}) - Q(\mathrm{z}') = \\
= \; & \textstyle\sum_{j=1}^{n} u_{\mathrm{z}(j)} \cdot r_{\mathrm{z}(j)} - \sum_{i=1}^{n} u'_{\mathrm{z}'(j)} \cdot r'_{\mathrm{z}'(j)} = \\
= \; & \left[\textstyle\sum_{j=1}^{k-1} u_{\mathrm{z}(j)} \cdot r_{\mathrm{z}(j)} + u_{\mathrm{z}(k)} \cdot r_{\mathrm{z}(k)} + u_{\mathrm{z}(k+1)} \cdot r_{\mathrm{z}(k+1)} + \sum_{j=k+2}^{n} u_{\mathrm{z}(j)} \cdot r_{\mathrm{z}(j)}\right] - \\
- \; & \left[\textstyle\sum_{j=1}^{k-1} u'_{\mathrm{z}'(j)} \cdot r'_{\mathrm{z}'(j)} + u'_{\mathrm{z}'(k)} \cdot r'_{\mathrm{z}'(k)} + u'_{\mathrm{z}'(k+1)} \cdot r'_{\mathrm{z}'(k+1)} + \sum_{j=k+2}^{n} u'_{\mathrm{z}'(j)} \cdot r'_{\mathrm{z}'(j)}\right] \overset{(11)}{=} \qquad (13) \\
= \; & u_{\mathrm{z}(k)} \cdot r_{\mathrm{z}(k)} + u_{\mathrm{z}(k+1)} \cdot r_{\mathrm{z}(k+1)} - u'_{\mathrm{z}'(k)} \cdot r'_{\mathrm{z}'(k)} - u'_{\mathrm{z}'(k+1)} \cdot r'_{\mathrm{z}'(k+1)} \overset{(12)}{=} \\
= \; & \left[f^{-1}_{\mathrm{z}(k)}\left(\frac{\overset{+}{t}^{r}_{\mathrm{z}(k)}}{r_{\mathrm{z}(k)}}\right) \cdot r_{\mathrm{z}(k)} + f^{-1}_{\mathrm{z}(k+1)}\left(\frac{\overset{+}{t}^{r}_{\mathrm{z}(k+1)}}{r_{\mathrm{z}(k+1)}}\right) \cdot r_{\mathrm{z}(k+1)}\right] - \\
- \; & \left[f^{-1}_{\mathrm{z}(k+1)}\left(\frac{\overset{+}{t}^{r}_{\mathrm{z}(k+1)}}{r_{\mathrm{z}(k)}}\right) \cdot r_{\mathrm{z}(k)} - f^{-1}_{\mathrm{z}(k)}\left(\frac{\overset{+}{t}^{r}_{\mathrm{z}(k)}}{r_{\mathrm{z}(k)} + p_{\mathrm{z}(k+1)}}\right) \cdot \left(r_{\mathrm{z}(k)} + p_{\mathrm{z}(k+1)}\right)\right],
\end{aligned}
$$

gdzie $r_{\mathrm{z}(k)} = \hat{C} - \sum_{i=k}^{n} p_{\mathrm{z}(i)}$ $(k = 1, 2, \ldots, n)$.

Ze wzoru (13) wynika, że obliczenie wartości wyrażenia $Q(\mathrm{z}) - Q(\mathrm{z}')$ wymaga wykonania elementarnych operacji arytmetycznych (zakładając, że potrafimy łatwo wyznaczyć postać funkcji $f_j^{-1}(\cdot)$). Jeśli $Q(\mathrm{z}) - Q(\mathrm{z}') > 0$, to oznacza, że permutacja z' odpowiada mniejszej wartości funkcji celu, a zatem celowa jest zamiana zadań na pozycjach k i $k+1$. Powyższa obserwacja stanowi podstawę konstrukcji przybliżonego algorytmu 1 wyznaczania kolejności wykonywania zadań.

Algorytm 1. *Przybliżony algorytm wyznaczania kolejności wykonywania zadań dla problemu (4).*

1 z := „dowolna permutacja ze zbioru Z"
2 **for** $i := n-1, n-2, \ldots, 1$ **do**
3 **for** $k := 1, 2, \ldots, i$ **do**
4 $\mathrm{z}' := [\mathrm{z}(1), \ldots, \mathrm{z}(k-1), \ldots, \mathrm{z}(k+1), \mathrm{z}(k), \mathrm{z}(k+2), \ldots, \mathrm{z}(n)]$
5 **if** $\mathrm{z}' \in \mathrm{Z}$ **then**
6 **if** $Q(\mathrm{z}) - Q(\mathrm{z}') > 0$ **then** $\mathrm{z} := \mathrm{z}'$
7 $\mathrm{z}^* := \mathrm{z}$

W kroku 1 następuje inicjalizacja algorytmu poprzez wybranie ze zbioru Z *dowolnej permutacji dopuszczalnej. W kroku 4 utworzona zostaje permutacja* z'. *Jeśli permutacja* z' *jest dopuszczalna (krok 5) oraz odpowiada mniejszej wartości funkcji celu niż permutacja* z *(krok 6), to ta permutacja jest używana do dalszych obliczeń. Pętle 2 i 3 sterują kolejną zamianą zadań w rozpatrywanych permutacjach. Złożoność tego algorytmu wynosi* $O(n^2 \cdot g(n))$, *gdzie* $g(n)$ *jest złożonością obliczania funkcji* $f_j^{-1}(\cdot)$.

Po zakończeniu działania powyższego algorytmu otrzymujemy permutację z^*, która odpowiada semioptymalnej wartości funkcji celu dla rozważanego problemu, tj.

$$Q(z^*) = \sum_{j=1}^{n} u_{z^*(j)} \cdot r_{z^*(j)},$$

gdzie

$$r_{z^*(j)} = \hat{C} - \sum_{i=j}^{n} p_{z^*(i)} \quad \text{oraz} \quad u_{z^*(j)} = f^{-1}_{z^*(j)} \left(\frac{\hat{x}^r_{z^*(j)}}{\hat{C} - \sum_{i=j}^{n} p_{z^*(i)}} \right).$$

5. Uwagi końcowe i uogólnienia

Godnym uwagi uogólnieniem rozważanego problemu jest rozważenie problemu odwrotnego do (4). Ponadto, rozpatrywany w niniejszej pracy problem można uogólnić na przypadek, gdy ilość zasobu dostępna do rozdysponowania w danej chwili jest ograniczona. Istotnym rozszerzeniem tego problemu może być także przyjęcie dowolnych (nie tylko wklęsłych) funkcji modeli terminów dostępności.

LITERATURA

1. Burkov W.N.: Sterowanie optymalne w kompleksach operacji, *IFAC*, 5:46–57, 1965.
2. Janiak A., Przysada J.: Minimalizacja nieterminowości wykonania zadań na równoległych maszynach, Zeszyty Naukowe Politechniki Śląskiej, s. Automatyka, z. 117, Gliwice 1996, str. 85–97.
3. Janiak A.: Dokładne i przybliżone algorytmy szeregowania zadań i rozdziału zasobów w dyskretnych procesach przemysłowych, Monografie 87/20, Politechnika Wrocławska, Instytut Cybernetyki Technicznej, 1991.
4. Janiak A.: Wybrane problemy i algorytmy szeregowania zadań i rozdziału zasobów, Problemy współczesnej nauki, Teoria i zastosowania, Informatyka, Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1999.
5. Józefowska J.: Dyskretno-ciągłe problemy szeregowania zadań, Rozprawy nr 318, Wydawnictwo Politechniki Poznańskiej, Poznań, 1997.
6. Węglarz J.: Multiprocessor Scheduling with Memory Allocation - A Deterministic Approach, *IEEE Transactions on Computers*, C-29(8):703–709, August 1980.

Recenzent: Prof. dr inż. T. Puchałka

Abstract

In this paper we present a solution of the problem of minimizing total amount of resource consumed in the resource allocation and scheduling problem with dynamic (differential) models of job release dates. The considered problem is a very hard dynamic optimization problem (because of the models of release dates) and static optimization problem (because of the scheduling). The dynamic models are often met in many computer applications (e.g. preparing of animations and picture rendering in the Computer Aided Design programs) as well as in industrial ones.
Before a job is ready for processing on the critical machine it must undergo initial treatment on parallel machines. The length of this initial treatment can be considered as the release date and is dynamically dependent on the amount of resource consumed. The processing time of each job on the critical machine is constant and known a priori. There is also a constraint set on the time of finishing processing of all the jobs in this problem.

Many important properties of this problem have been proven (e.g. those concerning the character of the resource allocation function). The presented scheduling algorithm is based on the properties proven in this paper. There are also given some generalizations of the considered problem.