ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996
Seria: AUTOMATYKA z.117 Nr kol. 1337

Adam JANIAK, Jacek PRZYSADA
Politechnika Wrocławska

CZASOWO-OPTYMALNE SZEREGOWANIE ZADAŃ Z RÓŻNYMI TERMINAMI GOTOWOŚCI NA RÓWNOLEGŁYCH MASZYNACH

Streszczenie. W pracy sformułowano problem szeregowania zadań na równoległych maszynach. Zakłada się, że dla każdego zadania dany jest termin jego gotowości oraz różniczkowy model w postaci funkcji opisującej prędkość zmian stanu zadania w zależności od ilości zasobów przydzielonych w każdej chwili czasu. Zadany jest też stan początkowy zadania i stan końcowy, który zadanie musi osiągnąć, aby było wykonane. Przyjęto ponadto, że zadania są niepodzielne. Globalna ilość podzielnych w sposób ciągły zasobów dostępna w każdej chwili czasu jest stała. Zadania mają być wykonane na identycznych równoległych maszynach, przy czym zakłada się, że ilość zadań jest nie mniejsza niż ilość maszyn. Problem polega na znalezieniu takiego przyporządkowania zadań do maszyn oraz takiego uporządkowania zadań na maszynach, a także takiego rozdziału ograniczonych zasobów pomiędzy zadania w każdej chwili czasu, aby czas wykonania wszystkich zadań był minimalny. Problem należy do klasy problemów NP-trudnych, więc do jego rozwiązania zaproponowano algorytm heurystyczny, wykorzystujący pewne własności tego problemu.

TIME-OPTIMAL TASK SCHEDULING WITH DIFFERENT READY TIMES

Summary. In the paper we describe and solve tasks scheduling and additional continuous resource allocation problem on identical, parallel machines. The results are obtained for differential task models. We assume that all tasks are independent, nonpreemptive and we have their ready times which are not equal to zero, initial and final states of each task are given. Task is finished when final state is achieved. We also assume that number of tasks is greater than number of machines and the amount of resources available at each moment is constant. The purpose of the optimization is to find such a schedule of tasks on machines and such an allocation of limited continuous resources among tasks at each moment that schedule lenght criterion is minimized. Because our problem belongs to the class of NP-hard problems we propose a heuristic algorithm which employs some problem properties.

1. Wstęp

W praktyce bardzo często spotykamy się z problemem czasowo-optymalnego sterowania rozdziałem zasobów i przydziałem zadań do maszyn, np. w wieloprocesorowych i wielodostępnych systemach cyfrowych, w sieciach komputerowych i wielozadaniowych systemach operacyjnych, złożonych procesach produkcyjnych. Problemem jest zarówno

uszeregowanie zadań na poszczególnych maszynach lub procesorach, jak i optymalny rozdział dodatkowych, podzielnych w sposób ciągły zasobów. Zwykle dane są terminy gotowości do realizacji poszczególnych zadań, ich modele, a także rozmiary poszczególnych zadań. Dodatkowym zasobem może być np. energia elektryczna, natężenie gazu, ogólnie paliwa, środki ludzkie, pamięć komputerowa itp. (tzn. zasoby podzielne w sposób ciągły), których chwilowe zużycie może podlegać pewnym ograniczeniom.

Problematyka ta była rozpatrywana między innymi w pracach [1-7]. W niniejszej pracy sformułowano i rozwiązano problem szeregowania zadań i rozdziału ograniczonych zasobów na maszynach równoległych przy założeniu, że terminy dostępności zadań nie są jednakowe. Jako kryterium optymalizacji przyjęto czas realizacji wszystkich zadań. Wykazano pewne własności problemu, w tym przynależność do klasy problemów NP-trudnych. Pokazano np. (dla przypadku gdy liczba zadań nie jest większa niż liczba maszyn), że jeśli tylko kolejne zadania pojawiają się zanim poprzednie jeszcze nie zostały zrealizowane, to z punktu widzenia minimum czasu realizacji całej sekwencji, wszystkie zadania kończą się równocześnie. Wykazane własności wykorzystano do konstrukcji algorytmu heurystycznego rozwiązującego rozpatrywany problem.

W rozdziale następnym w sposób precyzyjny sformułowano problem pracy. W rozdziale 3 zaproponowano jego rozwiązanie. Mianowicie w punkcie 3.1 rozpatrzony został przypadek, gdy liczba zadań nie jest większa od liczby maszyn. Sprowadzono ten podproblem do problemu programowania wypukłego. W punkcie 3.2 uogólniono rozwiązanie na przypadek, gdy liczba zadań jest większa od liczby maszyn. Rozdział 4 zawiera podsumowanie uzyskanych rezultatów.

2. Sformułowanie problemu

DANE:

- zbiór A-n zadań A_i, $i=1,\dots,n$, $A=\{A_1, A_2,\dots, A_i,\dots, A_n\}$,
- zbiór M-m maszyn M_j, $j=1,\dots,m$, $M=\{M_1, M_2,\dots, M_j,\dots, M_m\}$,
- liczba zadań jest większa od liczby maszyn: $n > m$,
- jeden rodzaj dodatkowych zasobów,
- stan każdego zadania w chwili t jest funkcją przydzielonych do jego wykonywania zasobów i opisany jest za pomocą równania różniczkowego:

$$\frac{dx_i(t)}{dt} = f_i[u_i(t)], \qquad i = 1,\dots,n,$$

- stan początkowy x_i^0 oraz stan końcowy x_i^* zadania A_i, $i=1,\dots,n$.

gdzie: - $x_i(t)$ stan zadania A_i, a $u_i(t)$ ilość zasobów przydzielona do zadania A_i w chwili t,
- $f_i(\cdot)$ ciągła, rosnąca, wklęsła funkcja,

- dla każdego zadania A_i znany jest jego moment t_i gotowości do rozpoczęcia wykonywania, $i=1,\dots,n$.

OGRANICZENIA:

- zadania niezależne i niepodzielne (nie można zatem przerwać wykonywania zadania i dokończyć go w innym terminie),
- różne czasy gotowości zadań,
- identyczne, równoległe maszyny,
- zbiór dopuszczalnych rozdziałów zasobów $U_{N(t)}$ pomiędzy poszczególne zadania w każdej chwili czasu t jest zdefiniowany następująco:

$$U_{N(t)} \equiv \{\bar{u}(t) \in R^{+} : u(t) = [u_1(t), u_2(t), \dots, u_i(t), \dots, u_n(t)] \wedge$$

$$\forall_{i=1,\dots,n} (\forall_{t<t_i} u_i(t) = 0 \wedge \forall_{t \geq t_i} u_i(t) \geq 0) \wedge \forall_{t' \geq 0} \forall_{i=1,\dots,n} [(x_i(t') = x_i^{*}) \Rightarrow$$

$$\Rightarrow (\forall_{t \geq t'} u_i(t) = 0)] \wedge \forall_{t \geq 0} \sum_{i=1}^{n} u_i(t) \leq N(t)\}.$$

- ograniczenia dotyczące zadań i funkcji zasobowych:

$$\forall_{i=1,\dots,n}\ x_i(0) = 0,\ \forall_{i=1,\dots,n}\ f_i(0) = 0,\ \forall_{i=1,\dots,n}\ f_i(u_i) \in C_0,$$

NALEŻY ZNALEŹĆ

takie uszeregowanie zadań na maszynach i taki rozdział ograniczonych zasobów pomiędzy zadania w każdej chwili czasu, aby czas wykonania wszystkich zadań był minimalny. Kryterium optymalizacji przyjmuje zatem postać:

$$Q = T = \min\{t \in R^{+} : \forall_{i=1,\dots,n} x_i(t) = x_i^{*}\},$$

zakładamy bowiem, że: $\forall_{t' \geq 0} \forall_{i=1,\dots,n} [(x_i(t') = x_i^{*}) \Rightarrow (\forall_{t \geq t'} x_i(t) = x_i^{*})]$,

co oznacza, że jeśli w danym momencie dane zadanie osiągnęło swój zadany stan końcowy, to w każdej następnej chwili pozostaje ono w tym samym stanie.

Należy zatem minimalizować kryterium Q = T, czyli czas, w którym wszystkie zadania osiągną stany końcowe, znajdując równocześnie optymalny rozdział zasobów pomiędzy zadania:

$\bar{u}^{*}(t) = [u_1^{*}(t), u_2^{*}(t), \dots, u_i^{*}(t), \dots, u_n^{*}(t)]$. Minimalny czas realizacji wszystkich zadań będziemy oznaczali przez T^{*}, tzn. $T^{*} = \min_{u(t) \in U_N(t)} T$.

Opisany powyżej problem oznaczymy jako π_1.

3. Rozwiązanie problemu π_1

Problem π_1 jest uogólnieniem klasycznego problemu szeregowania zadań na maszynach równoległych i rozdziału zasobów podzielnych w sposób ciągły dla liczby zadań większej niż liczba maszyn. Wersja decyzyjna problemu π_1 należy zatem do klasy problemów NP-zupełnych i dlatego do jego rozwiązania proponuje się algorytm heurystyczny.

W proponowanym algorytmie heurystycznym najpierw tworzy się listę zadań, w której kolejność wyznaczana jest przez czasy gotowości zadań do wykonania - od najmniejszego, do największego. Algorytm składa się z dwóch części. Pierwsza z nich dotyczy rozwiązania problemu w odpowiednio stworzonych podzbiorach zadań, w których liczba zadań nie przekracza liczby maszyn (n <=m). Druga część algorytmu dotyczy odpowiedniego łączenia rozwiązań otrzymanych w poszczególnych podzbiorach.

3.1. Przypadek $n \le m$

3.1.1. Własności rozwiązania problemu π_1

Pokażemy teraz pewne własności rozwiązania problemu π_1, przy założeniu, że istnieje rozwiązanie tego problemu, by następnie korzystając z tego, że jeśli istnieje rozwiązanie, to musi mieć pokazane poniżej własności, przejść w podrozdziale 3.1.2, do udowodnienia istnienia rozwiązania problemu π_1. Cechy rozwiązania problemu π_1 ujmiemy w zwięzłą formę własności.

Własność 1

Jeśli spełnione są założenia problemu π_1 i warunek gotowości kolejnych zadań do realizacji zanim mogło nastąpić zrealizowanie poprzedzających je zadań, to rozwiązanie optymalne ma następującą własność:

$$\forall_{i=1,\dots,n}\exists_{r=i,\dots,n}\exists_{\theta_i>0}\forall_{t\in<t_r,t_r+\theta_i>}(x_i^* > 0 \Rightarrow u_i(t) > 0) \wedge$$

$$\wedge\forall_{t\ge 0}[(\exists_{i=1,\dots,n} 0 \le x_i(t) < x_i^*) \Rightarrow \sum_{i=1}^{n} u_i(t) = N(t)],$$

Oznacza to, że z punktu widzenia minimum czasu realizacji wszystkich zadań (dla $n \le m$) optymalny czas rozpoczęcia realizacji każdego zadania jest albo momentem pojawienia się danego zadania, albo momentem pojawienia się pewnego następującego zadania po zadaniu rozpatrywanym. Poza tym, w każdej chwili czasu realizacji zadań należy wykorzystać całkowitą ilość zasobów przeznaczoną do rozdysponowania w danej chwili czasu.

Dowód własności 1, podobnie jak dowody kolejnych własności, znajduje się w pracy [1], gdzie rozpatrywano między innymi problem czasowo-optymalnego sterowania sekwencją

kompleksów operacji, do której można sprowadzić rozpatrywany tutaj przypadek $n \leq m$. Wyjaśnienia wymaga dodatkowe ograniczenie, jakim jest konieczność spełnienia warunku gotowości kolejnych zadań do realizacji, zanim mogło nastąpić zrealizowanie poprzedzających je zadań. Warunek "ciągłości" pojawiania się zadań jest jednak w praktyce na ogół spełniony.

Własność 2

Jeśli spełnione są założenia problemu π_1 i

$$\forall_{t' \geq 0} \forall_{t'' \geq 0} (t' < t'' \Rightarrow N(t') \geq N(t'')),$$

to jedno z rozwiązań optymalnych ma następującą własność:

$$\forall_{i=1,..,n} \exists_{\theta_i > 0} \forall_{t \in <t_i, t_i + \theta_i>} (x_i^* > 0 \Rightarrow u_i(t) > 0).$$

Oznacza to, że jeśli funkcja dysponowanej do rozdziału ilości zasobów jest funkcją nierosnącą, to z punktu widzenia minimum czasu realizacji wszystkich zadań, realizację poszczególnych zadań należy rozpocząć w momentach ich pojawienia się. Będziemy zakładać, że $N(t) = N = const$.

Własność 3

Jeśli spełnione są założenia problemu π_1, to rozwiązanie optymalne ma następującą własność:

$$\exists_{\theta_n > 0} \forall_{t_n \leq t \leq t_n + \theta_n} (x_n^* > 0 \Rightarrow u_n(t) > 0) \wedge$$

$$\wedge \forall_{t \geq t_n} [(\exists_{i=1,..,n} 0 \leq x_i(t) < x_i^*) \Rightarrow \sum_{i=1}^{n} u_i(t) = N].$$

co oznacza, że z punktu widzenia minimum czasu realizacji wszystkich zadań optymalnie jest rozpocząć realizację ostatniego zadania w momencie pojawienia się go wykorzystując całkowitą ilość zasobu.

Własność 4

Jeśli istnieje optymalne rozwiązanie problemu π_1:

$$\bar{u}^*(t) = [u_1^*(t), u_2^*(t), ..., u_i^*(t), ..., u_n^*(t)].$$

to istnieje równoważne mu w sensie kryterium optymalne rozwiązanie mające następujące własności: $\forall_{r=1,...,n-1} \forall_{t \in <t_r, t_{r+1})} \forall_{i=1,...,r} u_i^*(t) = u_{ir}^* = const. \wedge$

$$\forall_{t \in <t_n, T^*>} \forall_{i=1,..,n} u_i^*(t) = u_{in}^* = const.$$

Własność 4 mówi zatem, że w przedziałach jednoczesnej stałości dysponowanej do rozdziału ilości zasobów i liczby zadań optymalne rozwiązanie będzie stałe w czasie.

Własność 5

Jeśli spełnione są założenia problemu π_1 i jeśli założymy, że poszukujemy rozwiązania optymalnego o własnościach wymienionych w Własności 4, to:

$$\forall_{i=1,\dots,n}(\exists_{k=2,\dots,n;\ k>i}\ x_i(t_k)=x_i^* \vee x_i(T^*)=x_i^*).$$

Oznacza to, że realizacja każdego zadania może kończyć się tylko w którymś z momentów pojawienia się następnych po nim zadań bądź - w momencie zakończenia realizacji wszystkich zadań.

Własność 6

Jeśli spełnione są założenia problemu π_1, to kryterium Q = T można zastąpić równoważnym pod względem rozdziału zasobów następującym kryterium:

$$Q_n = H^n$$

gdzie $H^n=T-t_n$ oznacza minimalny czas realizacji ostatniego zadania i nie zrealizowanych części zadań poprzedzających to ostatnie zadanie.

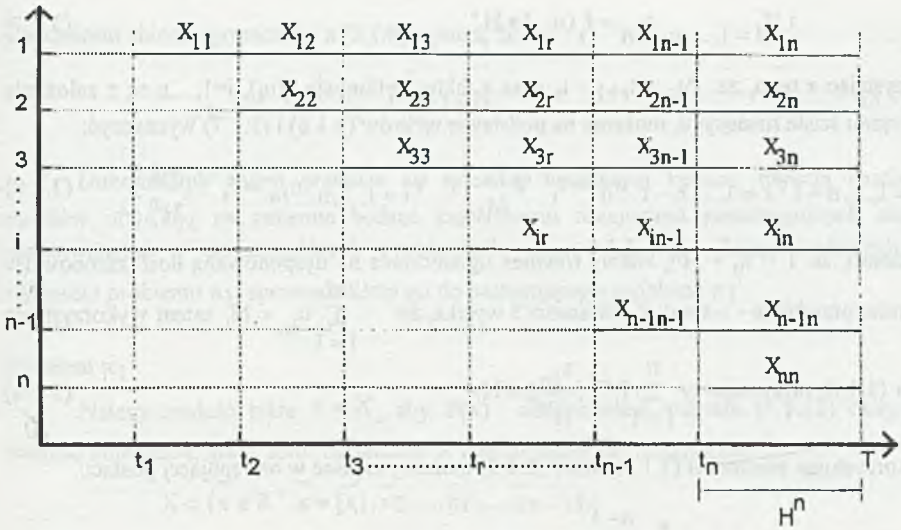
H^n jest zależne od stanów poszczególnych zadań osiągniętych w chwili t_n, czyli w momencie pojawienia się ostatniego zadania.

Zbiór dopuszczalnych rozdziałów zasobów w ostatnim n-tym przedziale czasowym, tzn. dla $t\in\langle t_n,T^*\rangle$, zdefiniowany jest następująco:

$$U_N = \{\bar{u}_n \in R^+ : \bar{u}_n = [u_{1n}, u_{2n}, \dots, u_{in}, \dots, u_{nn}] \wedge \forall_{i=1,\dots,n}\ u_{in} \geq 0 \wedge \sum_{i=1}^{n} u_{in} = N\}.$$

Naszym głównym celem (dla przypadku n<=m) jest znalezienie optymalnych funkcji rozdziału zasobów dla każdego zadania. Jednak celem znacznego uproszczenia notacji matematycznej problemu, jak i samych algorytmów rozwiązania, zamiast poszukiwać bezpośrednio optymalnego rozdziału zasobów, będziemy w poszczególnych przedziałach jednoczesnej stałości dysponowanej do rozdziału liczby zasobów i liczby zadań poszukiwali optymalnej ilości zadań częściowych poszczególnych operacji wykonanych w tych przedziałach. W tym celu wprowadzimy nowe oznaczenia. Przez x_{ir} będziemy oznaczali r-tą część i-tego zadania, która została wykonana w przedziale czasu $\langle t_r, t_{r+1})$, (por. rys.1). Oznaczamy także $\Delta t_r = t_{r+1} - t_r$, gdzie t_r oraz t_{r+1} są momentami gotowości odpowiednio zadania A_r oraz A_{r+1}, tzn.:

$$\forall_{i=1,\dots,n}\forall_{r=1,\dots,n}\ x_{ir} = x_i(t_{r+1}) - x_i(t_r) \text{ oraz } \forall_{A_i\in A}\ x_i^* = \sum_{r=i}^{n} x_{ir}. \qquad (3.1.1)$$



Rys.1.Podział zadań na części realizowane w kolejnych przedziałach czasowych

Fig.1.Division of tasks on parts realized in successive time intervals

3.1.2. Sprowadzenie problemu do zadania programowania wypukłego. Istnienie i jednoznaczność rozwiązania

W niniejszym podrozdziale, wykorzystując zaprezentowane wcześniej własności rozwiązania problemu π_1, pokażemy, że problem π_1, będący skomplikowanym problemem optymalizacji dynamicznej, można sprowadzić do znanego problemu optymalizacji statycznej, a mianowicie do zadania programowania wypukłego.

Z (3.1.1) mamy:

$$\forall_{i=1,\ldots,n-1}\forall_{r=1,\ldots,n-1}\ x_{ir} = x_i(t_{r+1}) - x_i(t_r), \tag{3.1.2}$$

a dla ostatniego (n-tego) przedziału, tzn.dla $t\in\langle t_n,T\rangle$ otrzymujemy:

$$\forall_{i=1,\ldots,n}\ x_{in} = x_i^* - x_i(t_n), \tag{3.1.3}$$

Zatem możemy także napisać, że: $\forall_{i=1,\ldots,n-1}\ x_i(t_n) = \sum_{r=i}^{n-1} x_{ir}$, (3.1.4)

Na podstawie (3.1.3) i (3.1.4) możemy napisać, że:

$$\forall_{i=1,\ldots,n-1}\ x_{in} = x_i^* - \sum_{r=i}^{n-1} x_{ir},\quad x_{nn} = x_n^*,\ \text{bowiem}\ x_n(t_n) = 0. \tag{3.1.5}$$

Wykorzystując różniczkowy model zadania oraz Własność 4, możemy napisać:

$$\forall_{i=1,\ldots,n-1}\forall_{r=1,\ldots,n-1}\ x_{ir} = f_i(u_{ir})(t_{r+1} - t_r) \tag{3.1.6}$$

$$\text{i } \forall_{i=1,\dots,n}\ x_{in} = f_i(u_{in}) \bullet H^n \tag{3.1.7}$$

Korzystając z tego, że $\Delta t_r = t_{r+1} - t_r$ oraz z faktu, że funkcje $f_i(u_i)$, $i=1,\dots,n$ są z założenia funkcjami ściśle rosnącymi, możemy na podstawie wzorów (3.1.6) i (3.1.7) wyznaczyć:

$$\forall_{i=1,\dots,n-1}\forall_{r=1,\dots,n-1}\ u_{ir} = f_i^{-1}\left(\frac{x_{ir}}{\Delta t_r}\right) \text{ i } \forall_{i=1,\dots,n}\ u_{in} = f_i^{-1}\left(\frac{x_{in}}{H^n}\right). \tag{3.1.8}$$

Wiadomo, że $T = t_n + H^n$, znamy również ograniczenia na dysponowaną ilość zasobów (w ostatnim przedziale - n-tym). Z Własności 3 wynika, że $\sum_{i=1}^{n} u_{in} = N$, zatem wykorzystując wzór (3.1.8) otrzymujemy:

$$\sum_{i=1}^{n} f_i^{-1}\left(\frac{x_{in}}{H^n}\right) = N, \tag{3.1.9}$$

Wykorzystując zależności (3.1.5), wzór (3.1.9) możemy zapisać w następującej postaci:

$$\sum_{i=1}^{n-1} f_i^{-1}\left(\frac{x_i^* - \sum_{r=i}^{n-1} x_{ir}}{H^n}\right) + f_i^{-1}\left(\frac{x_n^*}{H^n}\right) = N. \tag{3.1.10}$$

Ze wzoru (3.1.10) możemy obecnie wyznaczyć H^n jako funkcję częściowych zadań wykonywanych w poszczególnych przedziałach czasu.

$$H^n \equiv F'(\bar{\bar{x}}, \bar{x}^*, N), \tag{3.1.11}$$

gdzie: $\bar{\bar{x}} = [\bar{x}_1, \bar{x}_2, \dots, \bar{x}_r, \dots, \bar{x}_{n-1}]$ i

$$\forall_{r=1,\dots,n-1}\forall_{i=1,\dots,n-1}\ \bar{x}_r = [x_{1r}, x_{2r}, \dots, x_{ir}, \dots, x_{rr}], \quad \text{natomiast}$$

$$\forall_{i=1,\dots,n}\ \bar{x}^* = [x_1^*, x_2^*, \dots, x_i^*, \dots, x_n^*].$$

Ponieważ $\bar{x}^*$ i N są zadanymi parametrami, więc dalej będzie nas interesowała tylko zależność:

$$H^n \equiv F(\bar{\bar{x}}), \tag{3.1.12}$$

Przyjrzyjmy się obecnie ograniczeniom, jakie muszą spełniać rozmiary częściowe zadań x_{ir} ($i=1,\dots,n-1$, $r=1,\dots,n-1$). Z własności funkcji $f_i(u_i(t))$, $i=1,\dots,n$ oraz z własności zbioru ograniczeń U_N wynika, że:

$$\forall_{i=1,\dots,n-1}\forall_{r=1,\dots,n-1}\ x_{ir} \geq 0, \tag{3.1.13}$$

Z równania (3.1.5) i z faktu, że $x_{in} \geq 0$ wynika:

$$\forall_{i=1,\dots,n-1}\ \sum_{r=i}^{n-1} x_{ir} - x_i^* \leq 0. \tag{3.1.14}$$

Z własności zbioru ograniczeń i z (3.1.8) wynika, że:

$$\forall_{r=1,\dots,n-1} \sum_{i=1}^{r} f_i^{-1}\left(\frac{x_{ir}}{\Delta t_r}\right) \le N. \tag{3.1.15}$$

Dokonaliśmy zatem przejścia ze zmiennej niezależnej będącej funkcją rozdziału zasobów $u(t) \in U_N$ na zmienne będące częściowymi rozmiarami poszczególnych zadań, realizowanymi w poszczególnych przedziałach Δt_r $t = 1,2,\dots,n$. Zatem wykorzystując własności problemu π_1, sprowadziliśmy go do następującego problemu π_1':

Problem π_1'

Należy znaleźć takie $\bar{\bar{x}} \in X$, aby $F(\bar{\bar{x}})$ zdefiniowane wzorem (3.1.12) osiągnęło wartość minimalną, gdzie zbiór ograniczeń X zdefiniowany jest następująco:

$$X \equiv \{\bar{\bar{x}} \in R^+ : \bar{\bar{x}} = [\bar{x}_1, \bar{x}_2, \dots, \bar{x}_r, \dots, \bar{x}_{n-1}] \wedge$$

$$\forall_{r=1,\dots,n-1} \forall_{i=1,\dots,n-1} [(\bar{x}_r = [x_{1r}, x_{2r}, \dots, x_{ir}, \dots, x_{rr}] \wedge x_{ir} \ge 0) \wedge$$

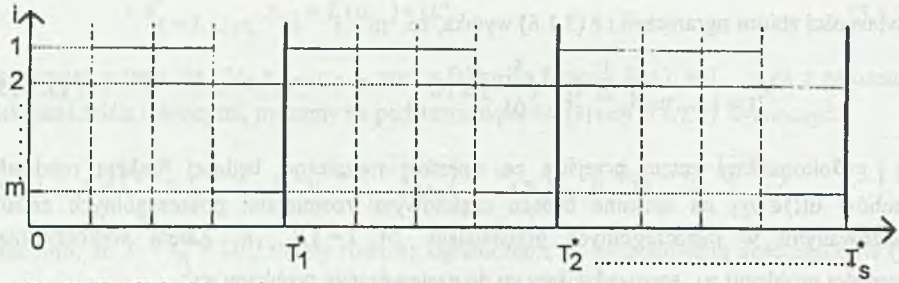
$$\sum_{r=i}^{n-1} x_{ir} - x_i^* \le 0 \wedge \sum_{i=1}^{r} f_i^{-1}\left(\frac{x_{ir}}{\Delta t_r}\right) \le N]\}.$$

Zależności pomiędzy funkcjami rozdziału zasobów $\bar{u}(t) \in U_N$ z problemu π_1 dla poszczególnych części zadań rozpatrywanych w kolejnych przedziałach r, r=1,2,...,n a $\bar{\bar{x}}$ - rozmiarami częściowymi poszczególnych zadań realizowanych w tych przedziałach określają wzory (3.1.8).

Dowód istnienia rozwiązania problemu π_1', (a co za tym idzie, problemu π_1) został podany w pracy [1], gdzie wykazano również, że problem π_1 jest problemem programowania matematycznego wypukłego, zatem do jego rozwiązania można zastosować znane metody numeryczne.

3.2. Przypadek n>m

W rozdziale 3.1 pokazaliśmy sposób rozwiązania przypadku, gdy $n \le m$. Zadania były pobierane z listy po kolei (według rosnących czasów gotowości) aż do momentu, gdy ich liczba zrównała się z liczbą maszyn. W dalszej fazie algorytmu kolejne zadania mogą być przydzielone do tych samych maszyn po zakończeniu realizacji pierwszej m-elementowej grupy zadań. Przy czym czas rozpoczęcia wykonywania kolejnej grupy zadań będzie równy czasowi zakończenia realizacji poprzedzającej je grupy zadań. Algorytm kończy działanie, gdy wyczerpiemy wszystkie zadania z listy zadań (rys.2).



Rys.2. Uogólnienie rozwiązania problemu na przypadek gdy n>m
Fig.2. Generalization of the problem solution on the case with n>m

Globalny minimalny czas realizacji wszystkich zadań jest zatem sumą minimalnych czasów realizacji zadań w poszczególnych podzbiorach, tzn. $T^* = T_1^* + T_2^* + ... + T_s^*$, gdzie: T_1^*, T_2^*, T_s^* - minimalne czasy realizacji zadań w poszczególnych podzbiorach.

4. Podsumowanie

Zaprezentowany algorytm umożliwia rozwiązanie skomplikowanego problemu optymalizacji dynamicznej. Podział zadań na podzbiory, w których liczba zadań nie przekracza liczby maszyn, upraszcza problem, gdyż w obrębie tych podzbiorów metoda rozwiązania sprowadza się do odpowiedniej zmiany rozdziału zasobów pomiędzy poszczególne zadania, bez potrzeby szeregowania zadań (przy czym w każdej chwili czasu warunek na globalną ilość zasobów musi być spełniony). Wydaje się, że algorytm może być poprawiony w części, w której uogólniono rozwiązanie na przypadek, gdy liczba zadań jest większa od liczby maszyn. Niektóre zadania z kolejnych m-elementowych podzbiorów mogłyby być realizowane wcześniej na zwolnionych maszynach. Zarówno polepszenie zaproponowanego algorytmu, jak i rozwiązanie podobnych problemów (np. z dodatkowymi ograniczeniami dotyczącymi zadań i zasobów), jest przedmiotem dalszych badań autorów niniejszej pracy.

LITERATURA

1. Janiak A.: Czasowo-optymalne sterowanie sekwencją kompleksów operacji niezależnych, Raporty Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej, nr 426, 976 r. (praca doktorska).
2. Józefowska J., Węglarz J.: On a Methodology for Discrete-Continuous Scheduling, Poznań University of Technology, Research Report RA-004/95.
3. Węglarz J.: Minimalno-czasowe sterowanie rozdziałem zadań i zasobów w kompleksie operacji w warunkach deterministycznych, Wydawnictwo Politechniki Poznańskiej, Poznań 1976.

4. Węglarz J.: Multiprocessor Scheduling with Memory Allocation-A Deterministic Approach, IEEE Transactions On Computers, vol. C-29. no. 8, August 1980, pp.703-709.

5. Węglarz J.: Project Scheduling with Discrete and Continuous Resources, IEEE Transactions On Systems, Man, And Cybernetics, vol. smc-9, no. 10, October 1979, pp.644-650.

6. Węglarz J.: Scheduling under continuous performing speed vs. resource amount activity models,Elsevier Science Publishers B.V., Amsterdam, 1989, pp.273-295.

7. Węglarz J.: Synthesis problems in Allocating continuous, doubly constrained resources among dynamic activities,Operation Research 1990, pp.715-730.

Recenzent: Dr hab. inż. Jan Kałuski, prof. Pol.Śl.

Wpłynęło do Redakcji do 30.06.1996 r.

Abstract

In the paper we formulate and solve tasks scheduling and additional continuous resource allocation problem on identical, parallel machines. Results obtained here are for differential task models. This model describes the speed of task state change in the function of allocated resources and seems to be more suitable for practical situations than other models. We assume that all tasks are independent, nonpreemptive and we have their ready times which are not equal to zero. We also know initial and final states of each task. Task is finished when final state is achieved. We assume that number of tasks is greater than number of parallel, identical machines and the amount of continuous resources available at each moment is constant. The purpose of optimization is to find such a schedule of tasks on machines and such an allocation of limited resources between tasks at each moment that schedule lenght criterion is minimized. Our problem belongs to the class of NP-hard problems what justifies searching for heuristic algorithm. In algorithm presented here, we create some subsets of tasks in which number of tasks is not greater then number of machines. We solve subproblems using consecutive subsets of tasks. Next, we properly join solutions of subproblems. It gives final solution. To solve subproblems in substes of tasks, we formulate and prove some properties of the problem. Next, on this basis we transform complicated dynamic optimization problem to easier convex programming problem.