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

Adam JANIAK, Jacek PRZYSADA
Politechnika Wrocławska

MINIMALIZACJA NIETERMINOWOŚCI WYKONANIA ZADAŃ NA RÓWNOLEGŁYCH MASZYNACH

Streszczenie. W pracy rozpatrzono problem szeregowania zadań i rozdziału zasobów na maszynach równoległych. Przyjęto różniczkowy model zadania opisujący prędkość zmian stanu zadania w funkcji przydzielonych zasobów. Założono, że zadania są niepodzielne, a ich czasy gotowości są równe zeru. Liczba zadań jest większa od liczby równoległych i identycznych maszyn. Globalna ilość podzielnych w sposób ciągły zasobów dostępna w każdej chwili czasu jest stała. Celem optymalizacji jest takie uszeregowanie zadań na maszynach i taki rozdział ograniczonych zasobów pomiędzy zadania w każdej chwili czasu, aby nieterminowość wykonania zadań była minimalna. Pokazano, że problem należy do klasy problemów NP-trudnych i w związku z tym zaproponowano algorytm heurystyczny.

MINIMIZATION OF TASKS MAXIMUM LATENESS ON PARALLEL MACHINES

Summary. In the paper we describe and solve tasks scheduling and additional continuous resource allocation problem on identical, parallel machines. Results obtained here are for differential task models. We assume that all tasks are independent, nonpreemptive and their ready times are 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 optimization is to find such a schedule of tasks on machines and such an allocation of limited continuous resources between tasks at each moment that maximum lateness criterion is minimized. We show that our problem belongs to the class of NP-hard problems and we propose a heuristic algorithm which employs some problem properties which are also proved.

1. Wstęp

Jednym z bardzo ważnych kryteriów optymalizacji spotykanych w praktyce jest nieterminowość zadanych do wykonania zadań, zleceń itp. Dotyczyć to może zarówno niewykonania w określonym terminie jakiegoś produktu, jak i pewnej usługi. Na ogół realizacja zlecenia polega na wykonaniu kilkunastu lub kilkudziesięciu zadań przy wykorzystaniu określonej z góry liczby maszyn lub potencjału ludzkiego. Terminowe wywiązanie się z przyjętych przez przedsiębiorcę czy projektanta obowiązków zależy w dużej

mierze od możliwości technicznych urządzeń i zdolności ludzkich. Istotne jest jednak, aby zarówno możliwości ludzkie, jak i wydajność urządzeń wykorzystać jak najlepiej. Nie zawsze zlecenie musi być wykonane "jak najszybciej". Często bowiem mamy zadany termin końcowy i w tym samym czasie możemy wykonywać również inne zadania. W takiej sytuacji musimy tak przyporządkowywać zadania do maszyn i wykorzystywać ewentualne dodatkowe zasoby, aby terminy wykonania wszystkich zadań zostały przekroczone w jak najmniejszym stopniu lub w ogóle. Dodatkowym zasobem może być np. energia elektryczna, natężenie gazu, ogólnie paliwa, środki ludzkie itp. (tzn. zasoby podzielne w sposób ciągły), których chwilowe zużycie może podlegać pewnym ograniczeniom.

W niniejszej pracy sformułowano i rozwiązano problem szeregowania zadań i rozdziału ograniczonych zasobów na maszynach równoległych. Jako kryterium optymalizacji przyjęto nieterminowość wykonania zadań. Udowodniono, że problem należy do klasy problemów NP-trudnych. Wykazano pewne własności problemu, które wykorzystano do stworzenia algorytmu heurystycznego rozwiązującego problem.

W rozdziale następnym w sposób precyzyjny sformułowano problem pracy. W rozdziale 3 zaproponowano jego rozwiązanie. 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:

- n zadań. Zbiór zadań oznaczamy jako A,
  $A=\{A_1, A_2,..., A_i,..., A_n\}$ $i = 1,...,n;$
- m maszyn. Zbiór maszyn oznaczamy jako M,
  $M=\{M_1, M_2,..., M_j,..., M_m\}$ $j = 1,...,m;$
- 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:

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

gdzie: - $x_i(t)$ stan zadania A_i w chwili t,

- $u_i(t)$ ilość zasobów przydzielona do zadania A_i w chwili t,
- $f_i(\cdot)$ ciągła, rosnąca, wklęsła funkcja,
- x_i^* zadany stan końcowy zadania A_i, i=1,...,n,
- dla każdego zadania A_i zadany jest pożądany termin jego zakończenia d_i, i=1,...,n.

OGRANICZENIA:

- zadania niezależne i niepodzielne (nie można zatem przerwać wykonywania zadania i dokończyć go w innym terminie);
- wszystkie zadania są gotowe do realizacji w chwili zerowej;
- identyczne, równoległe maszyny;
- zbiór dopuszczalnych rozdziałów zasobów $U_{N(t)}$ jest zdefiniowany poniżej:

Definicja 2.1

Zbiór dopuszczalnych rozdziałów zasobów pomiędzy poszczególne zadania w każdej chwili czasu t zdefiniowany jest następująco:

$$U_{N(t)} \equiv \{\bar{u}(t) \in R^+ : \bar{u}(t) = [u_1(t), u_2(t), \ldots, u_i(t), \ldots, u_n(t)] \wedge$$
$$\wedge \forall_{t' \geq 0}, \forall_{i=1..n} [(x_i(t') = x_i^*) \Rightarrow (\forall_{t \geq t'} u_i(t) = 0)] \wedge$$
$$\wedge \forall_{t \geq 0} \sum_{i=1}^{n} u_i(t) \leq N(t)\}$$

- ograniczenia dotyczące zadań i funkcji zasobowych:

$\forall_{i=1..n}\ x_i(0) = 0,$

$\forall_{i=1..n}\ f_i(0) = 0,$

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

$\forall_{i=1..n}\ \forall u_i^1 \in U_{N(t)}\ \forall u_i^2 \in U_{N(t)}\ [(u_i^1 > u_i^2) \Rightarrow (f_i(u_i^1) > f_i(u_i^2))]$, tzn. $f_i(u_i)$ jest rosnąca,

$\forall_{i=1..n}\ \forall u_i^1 \in U_{N(t)}\ \forall u_i^2 \in U_{N(t)}\ \forall \lambda \in [0,1]\ [f_i[\lambda \bullet u_i^1 + (1-\lambda) \bullet u_i^2] >$
$> [\lambda \bullet f_i(u_i^1) + (1-\lambda) \bullet f_i(u_i^2)]$, tzn $f_i(u_i)$ jest wklęsła.

NALEŻY ZNALEŹĆ

takie uszeregowanie zadań na maszynach i taki rozdział ograniczonych zasobów pomiędzy zadania w każdej chwili czasu, aby nieterminowość wykonania zadań była minimalna. Należy zatem minimalizować kryterium $Q = L_{max}$:

$$Q = L_{max} = \max_{1 \leq i \leq n}(L_i),$$

(gdzie $L_i = C_i - d_i$ jest nieterminowością zadania A_i, a C_i jest rzeczywistym czasem zakończenia zadania A_i),
znajdując optymalny rozdział zasobów pomiędzy zadania:

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

Opisany powyżej problem oznaczymy jako π_1 i będziemy w nim zakładać, że $N(t) = N = \text{const}$.

3. Metody rozwiązania problemu π_1

Powyższy problem jest skomplikowanym problemem optymalizacji dynamicznej, bowiem modele operacji - równania stanu - są funkcjami nieliniowymi i nieciągłymi względem czasu. Liczba zadań jest większa od liczby maszyn, więc zadania są uwarunkowane czasowo od poprzedzających je na maszynach zadań, co powoduje, że nawet nie można tego problemu sformułować w przyjętej konwencji zapisu problemów optymalizacji dynamicznej. Zatem nie można tutaj stosować znanych metod optymalizacji dynamicznej, takich jak zasada maksimum, programowanie dynamiczne, rachunek wariacyjny. Poniżej udowodnimy, że problem należy do klasy problemów NP- trudnych.

Twierdzenie 3.1

Wersja decyzyjna problemu π_1 opisanego szczegółowo w punkcie 2 jest problemem NP-zupełnym.

Dowód:

W celu udowodnienia twierdzenia 3.1. nie ma potrzeby konstruowania transformacji wielomianowej z decyzyjnej wersji pewnego problemu π_2, należącego do klasy problemów NP- zupełnych, do problemu π_1. Dowiedziono bowiem (J.Du i J.Leung, 1989 r.), że szczególny przypadek naszego problemu, tzn. problem szeregowania zadań na identycznych maszynach równoległych, w przypadku braku podzielności zadań i przy ograniczeniach kolejnościowych jest NP-zupełny nawet dla 2 maszyn (procesorów). Dla zbioru niezależnych i niepodzielnych zadań, dla m=2 lub m=3, problem może być rozwiązany przez algorytmy pseudowielomianowe. Jeśli natomiast zostaną nałożone dodatkowe ograniczenia, np. równe czasy wykonywania zadań, to problem dla m=2 lub m=3 oraz dla zbioru zadań niezależnych i niepodzielnych może mieć rozwiązanie wielomianowe. Dla m>5 problem jest silnie NP-zupełny.

Ograniczenie na równe czasy wykonywania zadań nie występuje w naszym problemie. Liczba niezależnych i niepodzielnych zadań w problemie π_1 jest większa od liczby maszyn, a liczba maszyn m > 2. Dodatkowym utrudnieniem w problemie π_1, oprócz uszeregowania zadań na maszynach, jest optymalny rozdział w każdej chwili czasu ograniczonych zasobów oraz fakt, że czasy wykonywania zadań nie są znane. Wynika z tego, że wersja decyzyjna problemu π_1 należy co najmniej do klasy problemów NP- zupełnych.

q.e.d. (tw. 3.1.)

Ze względu na fakt, iż nie istnieje algorytm wielomianowy rozwiązujący jakikolwiek problem NP-zupełny, jedyną drogą efektywnego rozwiązania problemu π_1 przy dowolnej liczbie maszyn i zadań jest udowodnienie specyficznych własności rozwiązania tego problemu, przy możliwie najsłabszych założeniach, a następnie wykorzystanie ich do znalezienia algorytmu optymalnego lub efektywnego algorytmu heurystycznego. W niniejszej pracy zajmiemy się konstrukcją algorytmu heurystycznego.

W proponowanym algorytmie heurystycznym na początku tworzy się listę zadań, w której kolejność wyznaczana jest przez pożądane terminy zakończenia zadań - 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≤m

3.1.1. Własności rozwiązania problemu π_1

Przedstawimy i udowodnimy dwie zasadnicze własności rozwiązania problemu dla przypadku n<=m. Zanim to jednak zrobimy, dokonamy odpowiedniego podziału poszczególnych zadań na części. W tym celu wprowadzimy zmienną $r \in C^+$ i podamy nowe definicje:

Przez x_{ir} będziemy oznaczali r-tą część i-tego zadania, która została wykonana w przedziale czasu $<C_{r-1}, C_r)$, gdzie C_{r-1} oraz C_r są momentami zakończenia odpowiednio zadania A_{r-1} oraz A_r, tzn.:

Definicja 3.1.1 (por. rys. 1)

$$\forall_{i=1...n}\forall_{r=1...n} x_{ir} = x_i(C_r) - x_i(C_{r-1})$$

Oczywiście:

$$\forall_{A_i \in A} \; x_i^* = \sum_{r=1}^{i} x_{ir} \tag{3.1.1}$$

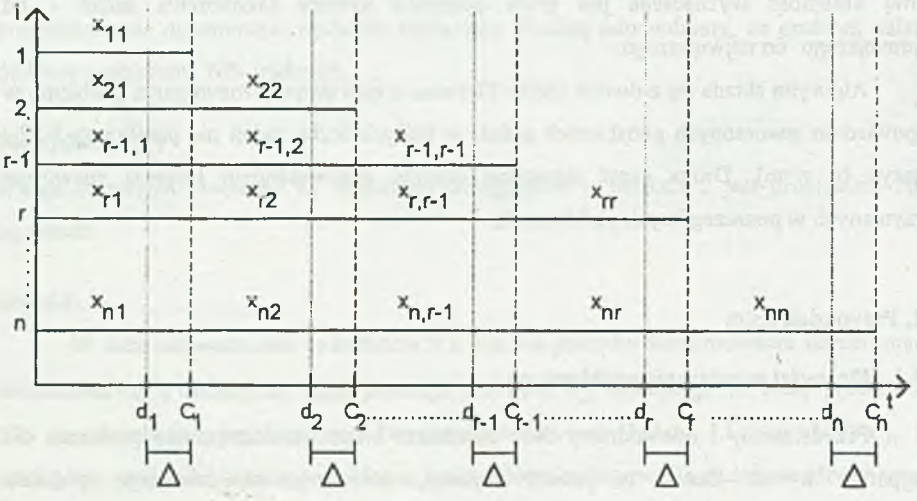
W celu łatwiejszego znalezienia rozwiązania wygodnie będzie poszukiwać optymalnego przydziału zasobów do poszczególnych części zadań. Tak więc konsekwencją wprowadzenia podziału zadań na części jest wprowadzenie podziału poszukiwanych optymalnych przydziałów zasobów, rozpatrywanych w poszczególnych przedziałach r, tzn. dla $t \in \langle C_{r-1}, C_r)$.

Definicja 3.1.2

Poszukiwany optymalny rozdział zasobów jest zdefiniowany następująco:

$$\bar{u}^*(t) = [\bar{u}_1^*(t), \bar{u}_2^*(t), \ldots, \bar{u}_i^*(t), \ldots, \bar{u}_n^*(t)] \text{ przy czym:}$$

$$\bar{u}_i^*(t) = [u_{i1}^*(t), u_{i2}^*(t), \ldots, u_{ir}^*(t), \ldots, u_{ii}^*(t)]$$



Rys. 1. Podział zadań na części realizowane w poszczególnych przedziałach czasowych
Fig.1. Division of tasks on parts realized in respective time intervals

Możemy teraz przystąpić do sformułowania własności:

Twierdzenie 3.1.1 (Własność 1)

Jeśli istnieje optymalne rozwiązanie problemu π_1:

$$\bar{u}^*(t) = [\bar{u}_1^*(t), \bar{u}_2^*(t), \ldots, \bar{u}_i^*(t), \ldots, \bar{u}_n^*(t)] \wedge$$
$$\wedge \; \bar{u}_i^*(t) = [u_{i1}^*(t), u_{i2}^*(t), \ldots, u_{ir}^*(t), \ldots, u_{ii}^*(t)],$$

to istnieje równoważne mu w sensie kryterium optymalne rozwiązanie mające następujące własności:

$$\forall_{t\in(C_{r-1},C_r)}\forall_{i=1..n}\forall_{r=1..i}\ u_{ir}^*(t) = u_{ir}^* = \text{const.}$$

Powyższe twierdzenie jest pewnym szczególnym przypadkiem znacznie ogólniejszego twierdzenia, które można by sformułować następująco:

Jeśli spełnione są założenia problemu π_1 i jeśli w przedziale $<C_{r-1}, C_r)$ istnieje pewien zmienny w czasie rozdział zasobów $u_{ir}^*(t) \in U_{N(t)}$, taki że powoduje on przejście zadania ze stanu $x_i^*(C_{r-1})$ do stanu $x_i^*(C_r)$, to istnieje w tym przedziale stały w czasie dopuszczalny rozdział zasobów $u_{ir}(t) = u_{ir}^* = \text{const.}$, $u_{ir}^* \in U_{N(t)}$, powodujący przejście zadania z tego samego stanu $x_i^*(C_{r-1})$ do tego samego stanu $x_i^*(C_r)$ w tym samym czasie $\Delta C = C_r - C_{r-1}$ co zmienny w czasie rozdział $u_{ir}^*(t)$.

Dowód twierdzenia 3.1.1 jest dość skomplikowany i długi. Szczegółowo przedstawiono go w pracy [6].

Dla rozpatrywanego problemu można wykazać następującą własność:

Twierdzenie 3.1.2 (Własność 2)

Jeżeli istnieje rozwiązanie optymalne dla π_1, to istnieje równoważne mu rozwiązanie, w którym wszystkie nieterminowości zadań są sobie równe.

Aby sformułować trzecią własność rozwiązania problemu π_1, zdefiniujemy pewną nową wielkość:

Definicja 3.1.4

Niech C_i, d_i oznaczają czasy odpowiednio rzeczywistego i pożądanego terminu zakończenia wykonywania zadania A_i. Definiujemy zmienną Δ_i, taką że

$$\forall_{i=1..n}\quad \Delta_i = C_i - d_i$$

Twierdzenie 3.1.3 (Własność 3)

Kryterium $Q = L_{max} = \max(C_i - d_i)$ można zastąpić równoważnym pod względem rozdziału zasobów następującym (efektywnym w praktycznym zastosowaniu) kryterium:

$$Q' = \Delta, \quad \text{gdzie } \Delta = \Delta_1 = \Delta_2 = \ldots = \Delta_i = \ldots = \Delta_n.$$

Zatem należy dokonać takiego rozdziału zasobów pomiędzy wykonywane zadania, aby dla wszystkich zadań różnice między rzeczywistymi a planowanymi czasami zakończenia zadań były równe i wynosiły Δ.

Dowód:

Spośród n wartości Δ_i należy wybrać największą i dążyć do jej minimalizacji. Poszczególne Δ_i są jednak od siebie zależne w sposób dynamiczny, tzn. zmiana przydziału zasobów do poszczególnych zadań powoduje zmianę długości odpowiednich Δ_i, przy czym skrócenie lub wydłużenie dowolnej Δ_i wpływa odpowiednio na wydłużenie lub skrócenie innych Δ_i, bowiem globalna ilość zasobów dostępnych w każdej chwili czasu jest stała i ograniczona. Dysponując największą Δ_{imax} spośród Δ_i, oczywisty jest fakt, że warto zmniejszyć wartość Δ_{imax} kosztem zwiększenia wartości innych Δ_i, tak jednak, by nowe wartości Δ_i nie przekroczyły nowej wartości Δ_{imax}. Idąc dalej tym tokiem rozumowania, doprowadzimy w konsekwencji do zrównania wszystkich wartości Δ_i.

q.e.d. (tw. 3.1.2.)

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

Wykorzystując twierdzenia udowodnione wcześniej, a dotyczące własności rozwiązania problemu π_1, udowodnimy, że problem π_1, będący skomplikowanym problemem optymalizacji dynamicznej, można sprowadzić do znanego problemu optymalizacji statycznej, a mianowicie do zadania programowania wypukłego.

Głównym zadaniem (dla przypadku $n \leq m$) jest znalezienie optymalnych funkcji rozdziału zasobów dla każdego zadania. W celu skonstruowania algorytmu i uproszczenia matematycznej notacji problemu wprowadzimy następujące oznaczenia:

$$\forall_{r=1...n}\ \Delta C_r = C_r - C_{r-1}.$$

Z danych i założeń wiemy, że:

$$\forall_{i=1..n}\forall_{t\in R^+}\ \frac{dx_i(t)}{dt} = f_i[u_i(t)],$$

zatem:

$$\forall_{i=1..n}\forall_{r=1..n}\forall_{t\in\Delta C_r}\ \int_{C_{r-1}}^{C_r}\frac{dx_i(t)}{dt}dt = \int_{C_{r-1}}^{C_r} f_i(u_i(t))dt\,.$$

Wykazaliśmy, że $\forall_{i=1...n}\ \forall t\in\Delta C_r\ u_i(t) = u_i = \text{const.}$ (Własność 1), zatem:

$$x_i(C_r) - x_i(C_{r-1}) = f_i(u_i)\bullet C_r - f_i(u_i)\bullet C_{r-1}, \text{ więc:}$$

$$x_i(C_r) - x_i(C_{r-1}) = f_i(u_i)\bullet\Delta C_r. \qquad (3.1.2)$$

Dla r=1, $\Delta C_r = \Delta C_1 = C_1 - C_0$, gdzie $C_0 = 0$ i $C_1 = d_1+\Delta$, czyli $\Delta C_1 = d_1+\Delta$.

Dla r=2...n, $\Delta C_r = C_r - C_{r-1}$, gdzie $C_r = d_r+\Delta$, $C_{r-1} = d_{r-1}+\Delta$,

czyli $\Delta C_r = d_r+\Delta - d_{r-1}-\Delta = d_r - d_{r-1}$.

Dla pierwszego przedziału r=1(rys.1, tj. dla $t\in<0,C_1)$) otrzymujemy:

$x_i(d_1+\Delta) - x_i(0) = f_i(u_{i1})\bullet(d_1+\Delta)$, czyli $x_{i1} = f_i(u_{i1})\bullet(d_1+\Delta)$,

zatem:

$$u_{i1} = f_i^{-1}\left(\frac{x_{i1}}{d_1+\Delta}\right). \tag{3.1.3}$$

Oczywiste jest, że $x_{11}=x_1^*$ (dla r=n=1).

Dla pozostałych przedziałów (rys.1, tj. dla $t\in<C_{r-1},C_r)$ r=2,...,n, otrzymujemy:

$x_i(C_r)-x_i(C_{r-1})=f_i(u_{ir})\bullet(C_r-C_{r-1})$, czyli $x_{ir}=f_i(u_{ir})\bullet(d_r-d_{r-1})$,

zatem:

$$u_{ir} = f_i^{-1}\left(\frac{x_{ir}}{d_r-d_{r-1}}\right). \tag{3.1.4}$$

Ze wzoru 3.1.1 otrzymujemy:

$$x_i^* = x_{i1} + \sum_{r=2}^{i} x_{ir}, \qquad i = 1,\ldots,n. \tag{3.1.5}$$

Biorąc pod uwagę (3.1.3), (3.1.4), (3.1.5) oraz Definicję 2.1, dostajemy:

Dla r=1 (pierwszy przedział):

$$\sum_{i=1}^{n} f_i^{-1}\left(\frac{x_i^* - \sum_{r=2}^{i} x_{ir}}{d_1+\Delta}\right) = N. \tag{3.1.6}$$

Dla r=2,...,n (pozostałe przedziały)

$$\sum_{i=r}^{n} f_i^{-1}\left(\frac{x_{ir}}{d_r-d_{r-1}}\right) \le N. \tag{3.1.7}$$

Przekształcając dalej zależność (3.1.6), dla pierwszego przedziału otrzymujemy:

$$f_i^{-1}\left(\frac{x_1^*}{d_1+\Delta}\right) + \sum_{i=2}^{n} f_i^{-1}\left(\frac{x_i^* - \sum_{r=2}^{i} x_{ir}}{d_1+\Delta}\right) = N. \tag{3.1.8}$$

Ze wzoru (3.1.8) możemy obecnie wyznaczyć Δ jako funkcję częściowych zadań wykonywanych w poszczególnych przedziałach czasu.

$$\Delta \equiv G\left(\bar{d}, \bar{x}^*, \bar{x}, N\right), \quad \text{gdzie:} \tag{3.1.9}$$

$\bar{d} \equiv [d_1, d_2, \ldots, d_i, \ldots, d_n]$,

$\bar{x}^* \equiv [x_1^*, x_2^*, \ldots, x_i^*, \ldots, x_n^*]$,

$\bar{x} \equiv [x_2, x_3, \ldots, x_r, \ldots, x_n]$, gdzie $\forall_{r=2..n}\ \bar{x}_r \equiv [x_{rr}, x_{r+1r}, \ldots, x_{ir}, \ldots, x_{nr}]$,

$N = \text{const}$.

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

$$\Delta \equiv G\left(\bar{x}\right). \tag{3.1.10}$$

Wobec tego, że były nałożone ograniczenia na funkcję rozdziału zasobów, przyjrzyjmy się obecnie ograniczeniom, jakie muszą spełniać rozmiary częściowe zadań x_{ir} (i=1...n, r=2...n). Z własności funkcji $f_i(u_i(t))$, i=1,...,n oraz z własności zbioru ograniczeń $U_{N(t)}$ wynika, że:

$$\forall_{i=2..n} \forall_{r=2..n} \quad x_{ir} \geq 0. \tag{3.1.11}$$

Z równania (3.1.5) mamy:

$$\forall_{i=1,..,n} \quad x_{i1} = x_i^* - \sum_{r=2}^{i} x_{ir} \ \text{ i } \ x_{i1} \geq 0,$$

$$\text{zatem musi zachodzić} \qquad \forall_{i=2,..,n} \sum_{r=2}^{n} x_{ir} - x_i^* \leq 0. \tag{3.1.12}$$

Z (3.1.7) otrzymujemy:

$$\forall_{r=2,..,n} \sum_{i=r}^{n} f_i^{-1}\left(\frac{x_{ir}}{d_r - d_{r-1}}\right) - N \leq 0 \tag{3.1.13}$$

Dokonaliśmy zatem przejścia ze zmiennej niezależnej będącej funkcją rozdziału zasobów $u(t) \in U_{N(t)}$ na zmienne będące częściowymi rozmiarami poszczególnych zadań, realizowanymi w poszczególnych przedziałach jednoczesnej stałości dysponowanej ilości zasobów. Zatem wykorzystując własności problemu π_1, sprowadziliśmy go do następującego problemu π_1':

Problem π_1'

Należy znaleźć takie $\bar{x} \in X$, aby $G(\bar{x})$ zdefiniowane wzorem (3.1.10) osiągnęło wartość minimalną. Zbiór ograniczeń zdefiniowany jest następująco:

Definicja 3.1.5

$$X \equiv \{\bar{x} \in R^{+} : \bar{x} = [\bar{x}_2, \bar{x}_3 \ldots, \bar{x}_r, \ldots, \bar{x}_n] \wedge$$
$$\wedge \forall_{r=2\ldots n} \bar{x}_r = [x_{rr}, x_{r+1r}, \ldots, x_{ir}, \ldots, x_{nr}] \wedge$$
$$\wedge \forall_{i=2\ldots n} \forall_{r=2\ldots n} x_{ir} \geq 0 \wedge$$
$$\wedge \forall_{i=2\ldots n} \sum_{r=2}^{n} x_{ir} - x_i^{\bullet} \leq 0 \wedge$$
$$\wedge \forall_{r=2\ldots n} \sum_{i=r}^{n} f_i^{-1}\left(\frac{x_{ir}}{d_r - d_{r-1}}\right) - N \leq 0\} .$$

Zależności między funkcjami rozdziału zasobów $\bar{u}(t) \in U_{N(t)}$ z problemu π_1 (dla poszczególnych części zadań rozpatrywanych w przedziałach jednoczesnej stałości dysponowanej do rozdziału ilości zasobów) a $\bar{x}$ - rozmiarami częściowymi poszczególnych zadań realizowanych w tych samych przedziałach określają równania (3.1.3) i (3.1.4)

Można udowodnić, że powyżej przedstawiony problem π_1' jest problemem programowania matematycznego wypukłego. Można także wykazać istnienie rozwiązania problemu π_1', co służy do wykazania istnienia rozwiązania problemu π_1. Ze względu na ograniczone rozmiary tej pracy nie będziemy tu przytaczać dowodów wyżej wymienionych twierdzeń. Metodologia postępowania, jaką należy przyjąć, jest pokazana w [6].

Pierwszą metodą rozwiązania problemu π_1', narzucającą się niemal natychmiast, jest metoda Kuhna-Tuckera. Metoda ta jest bardzo przydatna do rozważań teoretycznych dotyczących rozwiązania, ale mniej - do praktycznego wykorzystania. Inne metody to znane metody numeryczne, takie jak np. metoda Schmita i Foxa, metoda Rosenbrocka, metoda Carolla czy metoda Powela.

3.2. Przypadek n>m

Uogólnienie rozwiązania na przypadek, gdy liczba zadań jest większa od liczby maszyn, polega na stworzeniu m-elementowych podzbiorów zadań, a następnie znalezieniu rozwiązania w każdym podzbiorze. Podzbiory tworzymy biorąc kolejne zadania ze stworzonej wcześniej listy zadań, w której kolejność wyznaczana jest przez pożądane terminy zakończenia zadań - od najmniejszego do największego. W obrębie poszczególnych podzbiorów każde zadanie wykonywane jest na osobnej maszynie. Rozwiązanie całości problemu stanowi złożenie rozwiązań w poszczególnych podzbiorach, gdzie $n \leq m$, tj. rozwiązań przypadku omówionego w rozdziale 3.1.

4. Podsumowanie

Problem minimalizacji nieterminowości wykonania zadań na równoległych maszynach i jednoczesnego rozdziału ograniczonych, ciągłych zasobów jest skomplikowanym problemem optymalizacji dynamicznej. Sprowadzenie go do problemu programowania wypukłego umożliwia rozwiązanie poprzez zastosowanie jednej z kilku metod numerycznych. Wydaje się, że algorytm może być poprawiony w części, w której uogólniono rozwiązanie na przypadek, gdy liczba zadań jest większa od liczby maszyn. Zaproponowana metodologia rozwiązania może być zastosowana do rozwiązania podobnych problemów, np. z dodatkowymi ograniczeniami dotyczącymi zadań (np. różne czasy gotowości) lub zasobów (np. różne postacie funkcji zasobowych). Zarówno polepszenie zaprezentowanego algorytmu, jak i rozwiązanie innych problemów jest przedmiotem dalszych badań autorów niniejszej pracy.

LITERATURA

1. Błażewicz J., Drabowski M., Węglarz J.: Scheduling Independent 2-processor tasks to minimize schedule length. Information Processing Letters18 (1984), pp.267-273.
2. Błażewicz J., Drabowski M., Węglarz J.: Scheduling Multiprocessor Tasks to Minimize Schedule Length. IEEE Transactions On Computers, vol.C-35, no.5, May1986, pp.389-393
3. Błażewicz J., Drozdowski M., Węglarz J.: Scheduling Multiprocessor Tasks-A Survey. Microcomputer Applications, vol.13, no. 2, 1994, pp.89-97.
4. Błażewicz J., Drozdowski M., Schmidt G., de Werra D.: Scheduling Multiprocessor Tasks on Uniform Processors. Parallel Computing, Elsevier Science B.V. 1994, pp.249-256.
5. Błażewicz J., Drozdowski M., Węglarz J.: Szeregowanie zadań przed liniami krytycznymi. Zeszyty Naukowe Politechniki Śląskiej, Seria Automatyka z.109, Nr kol.1175, 1992, pp.19-28.
6. Janiak A.: Czasowo-optymalne sterowanie sekwencją kompleksów operacji niezależnych. Raporty Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej, Nr 426, 1976 r. (Praca Doktorska).
7. Józefowska J., Węglarz J.: On a Methodology for Discrete-Continuous Scheduling. Poznań University of Technology, Research Report RA-004/95.
8. Węglarz J.: Minimalno-czasowe sterowanie rozdziałem zadań i zasobów w kompleksie operacji w warunkach deterministycznych. Wydawnictwo Politechniki Poznańskiej, Poznań 1976.
9. Węglarz J.: Multiprocessor Scheduling with Memory Allocation-A Deterministic Approach. IEEE Transactions On Computers, vol. C-29. no. 8, August 1980, pp.703-709.
10. 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.

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

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

Recenzent: Dr hab.inż. Ewa Dudek-Dyduch

Wpłynęło do Redakcji do 30.06.1996 r.

Abstract

In the paper we describe and solve tasks scheduling and additional continuous resource allocation problem on identical, parallel machines. Results obtained here are for differential task model. 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 their ready times are 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 maximum lateness criterion is minimized. We show that 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. Finally we select numerical or analytical counting procedure.