ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1986
Seria: AUTOMATYKA z. 84 Nr kol. 894

Jacek Błażewicz^, Mieczysław Drabowski+,
Jan Węglarz^

^Politechnika Poznańska
+Zakłady MERA,Kraków

SZEREGOWANIE ZADAŃ WIELOPROCESOROWYCH Z OGRANICZENIAMI ZASOBOWYMI

Streszczenie. W pracy przeanalizowano problem szeregowania zadań wieloprocesorowych,to znaczy żądających więcej niż jednego procesora do swego wykonania,przy ograniczeniach ze strony dodatkowych zasobów. Rozpatrzono minimalizację długości uszeregowania dla przypadku zadań podzielnych.

1. Wstęp

W ostatnich latach ukazało się szereg prac poświęconych nowemu kierunkowi w dziedzinie teorii szeregowania,związanemu z problemami szeregowania zadań wieloprocesorowych,to jest żądających więcej niż jednego procesora do swego wykonywania [3,4,5,11,13] . Problemy te znajdują istotne zastosowanie,między innymi,przy analizie i projektowaniu nowoczesnych struktur systemów wielomikroprocesorowych [2,9,14] . W niniejszej pracy poszerzymy powyższy model zakładając,że zadania mogą żądać do swego wykonania także dodatkowych zasobów,którymi w systemach komputerowych mogą na przykład być: pamięć operacyjna,kanały,pamięć masowa itp. Rozpatrzona zostanie minimalizacja długości uszeregowania dla przypadku zadań podzielnych.

W ogólności rozpatrywać będziemy zbiór m identycznych procesorów $P = \{P_1,P_2,\ldots,P_m\}$,zbiór s rodzajów dodatkowych zasobów $R = \{R_1,R_2,\ldots,R_s\}$ dostępnych odpowiednio w liczbie $r_1,r_2,\ldots,r_s$ jednostek, oraz zbiór n niezależnych i podzielnych zadań Z.
Zbiór Z składa się z k podzbiorów $Z^1 = \{z_1^1, z_2^1,\ldots,z_{n_1}^1\}$, $Z^2 = \{z_1^2, z_2^2,\ldots,z_{n_2}^2\},\ldots,Z^k = \{z_1^k, z_2^k,\ldots z_{n_k}^k\}$, gdzie $n_1 + n_2 +\ldots+ n_k = n$. Każde zadanie l-procesorowe Z_i^l, $i = 1,2,\ldots,n_1$; $l = 1,2,\ldots,k$, potrzebuje do swego wykonania l procesorów,które wykonają to zadanie pracując jednocześnie w ciągu t_i^l jednostek czasu. Oprócz odpowiedniej liczby procesorów zadanie Z_i^l, $i = 1,2,\ldots,n_1$; $l = 1,2,\ldots,k$ potrzebuje do swego wykonania także określonych liczb jednostek dodatkowych zasobów podanych w postaci wektora $R(Z_i^l) = [R_1(Z_i^l), R_2(Z_i^l),\ldots,R_s(Z_i^l)]$.

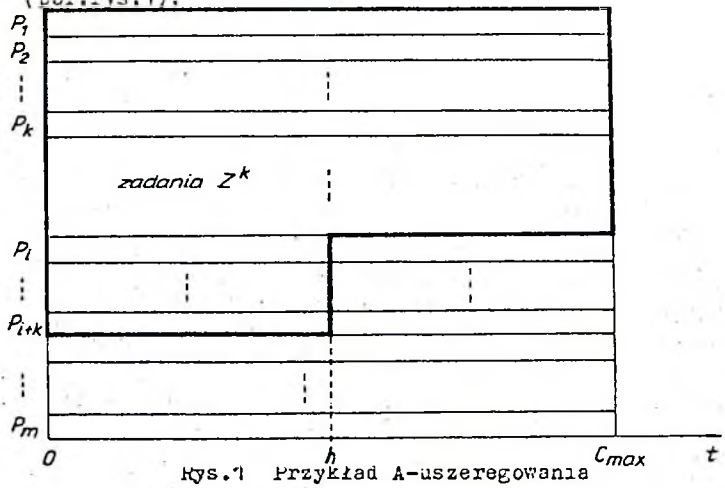
W kolejnych paragrafach pracy zostaną przedstawione: ulepszony algorytm szeregowania zadań bez ograniczeń zasobowych o złożoności $O(n)$ stanowiący podstawę do dalszych rozważań (paragraf 2); algorytmy wielomianowe dla poszczególnych przypadków żądań zasobowych zero-jedynkowych (pa-

ragraf 3); podejście programowania liniowego do ogólnego przypadku dowolnych żądań zasobowych (paragraf 4).

2. Algorytm szeregowania zadań 1- i k-procesorowych o złożoności O(n).

W pracach [3,4] przedstawiono algorytm szeregowania na m identycznych procesorach zadań podzielnych i niezależnych jedno- i k-procesorowych nie żądających dodatkowych zasobów o złożoności O(n log n + kn). Ostatnio w pracy [5] podano ulepszoną wersję tego algorytmu o złożoności O(n). Opiszemy krótko jego działanie, gdyż będzie on wykorzystany w dalszej części pracy przy szeregowaniu zadań żądających dodatkowych zasobów.

Algorytm ten, podobnie jak jego wersja pierwotna [4], wykorzystuje pojęcie A-uszeregowania, to znaczy uszeregowania, w którym najpierw do kolejnych procesorów w przedziale czasu $[0, C_{max}]$ przydzielane są wszystkie zadania k-procesorowe zgodnie z regułą Mc Naughtona [12] (por. także [8]), a następnie w pozostałej części uszeregowania umieszczone są zadania ze zbioru Z^1 (por. rys. 1).



Rys. 1 Przykład A-uszeregowania
An example A-schedule

Można wykazać [3], że wśród uszeregowań o minimalnej długości dla zbioru zadań o powyższych właściwościach, istnieje A-uszeregowanie. Korzystając z tego wyniku poszukuje się optymalnego A-uszeregowania.

Wprowadzimy teraz następujące oznaczenia.

$$X = \sum_{i=1}^{n_1} t_i^1 , \quad Y = \sum_{i=1}^{n_k} t_i^k , \quad Z = X + kY$$

$$t_{max}^1 = \max \left\{ t_i^1 : Z_i^1 \in Z^1 \right\} , \quad t_{max}^k = \max \left\{ t_i^k : Z_i^k \in Z^k \right\}$$

Ograniczenie na długość C_{max} dane jest wzorem [3,4]:

$$C_{max} \geq C = \max \left\{ Z/m, \; Y/\lfloor m/k \rfloor , \; t_{max}^1 , \; t_{max}^k \right\} \qquad /1/$$

Następujące rozumowanie prowadzi do znalezienia uszeregowania o minimalnej długości [5] . Niech $p = \left\lfloor \frac{Y}{C} \right\rfloor$ Długość optymalnego uszeregowania C^*_{max} musi spełniać następującą nierówność :

$$C \leq C^*_{max} \leq \frac{Y}{p}$$

Z poprzednich rozważań wiemy,że istnieje optymalne A-uszeregowanie,w którym k·p procesorów wykonuje jedynie zadania Z^k, k procesorów wykonuje zadania Z^k w przedziale $[0,h]$,a zadania Z^1 przydzielone są w pozostałej części uszeregowania (por.rys.1). Niech liczba procesorów,które mogą wykonywać zadania Z^1 w przedziale $[0,h]$ wynosi

$$m_1 = m - (p + 1) \cdot k$$

Dla A-uszeregowania o długości D,gdzie $C \leq D \leq \frac{Y}{p}$, h będzie dane wzorem

$$h = Y - Dp.$$

Zatem minimalna wartość C^*_{max} będzie najmniejszym D ($D \geq C$), takim że zadania Z^1 mogą być uszeregowane na $m_1 + k$ procesorach w przedziale $[Y - Dp, D]$ i na m_1 procesorach w przedziale $[0,D]$. Poniżej określimy konieczny i dostateczny warunek takiej uszeregowalności. W tym celu załóżmy na razie,że zadania Z^1 są uporządkowane tak,by $t^1_1 \geq t^1_2 \geq \ldots \geq t^1_{n_1}$. Teraz dla danej pary D,h ($h = Y - Dp$), niech $t^1_1, t^1_2, \ldots t^1_j$ będą jedynymi czasami wykonywania zadań większymi od D - h. Wówczas zadanie Z^1 można uszeregować na m procesach (łącznie z zadaniami Z^k) w przedziale $[0,h]$ wtedy i tylko wtedy,gdy

$$\sum_{i=1}^{j} \left[t^1_i - (D-h) \right] \leq m_1 h \qquad /2/$$

Dowód poprawności tego warunku zamieszczono w [5] .

Opiszemy teraz w jaki sposób można znaleźć minimalną wartość C^*_{max} . Niech $W_j = \sum_{i=1}^{j} t^1_i$. Nierówność /2/ można zapisać w następującej postaci

$$W_j - j \left(D - \left\lfloor Y - Dp \right\rfloor \right) \leq m_1 \left(Y - Dp \right).$$

Rozwiązując tę nierówność względem D uzyskujemy

$$D \geq \frac{(j - m_1) Y + W_j}{(j - m_1) p + j}$$

Zdefiniujmy teraz

$$C_j = \frac{(j - m_1) Y + W_j}{(j - m_1) p + j}$$

Możemy zatem napisać

$$C^*_{max} = \max \left\{ C, C_1, C_2, \ldots C_{n_1} \right\} .$$

Najbardziej oczywistą metodą znalezienia powyższego maksimum byłoby posortowanie zadań Z^1 względem ich czasów wykonywania t_i^1 i następnie określenie maksymalnej wartości C_j. Złożoność tego podejścia byłaby $O(n_1 \log n_1)$. Można jednak wykonać to szybciej [5], biorąc pod uwagę następujące fakty.

1. $C_j \leq C$ dla $j \leq m_1$ oraz $j \geq m_1 + k$

2. C_j nie ma lokalnych maksimów dla $j = m_1 + 1, \ldots, m_1 + k - 1$.

Stąd, aby znaleźć maksimum wśród $C_{m_1+1}, \ldots, C_{m_1+k-1}$ oraz C należy zastosować liniowy algorytm znajdowania mediany [1] i przeszukiwanie binarne, co prowadzi do algorytmu określania C^*_{max} o złożoności $O(n_1)$. Wynika to z faktu, że znajdowanie mediany po raz pierwszy, wymaga czasu $O(n_1)$, po raz drugi - $O(n_1/2)$, po raz trzeci - $O(n_1/4)$... Zatem łączny czas jest $O(n_1)$.

Możemy teraz przedstawić algorytm szeregowania badań podzielnych należących do zbiorów Z^1 oraz Z^k, w celu minimalizacji długości uszeregowania.

Algorytm 1

1. Oblicz minimalną długość uszeregowania C^*_{max}.
2. Przydziel zadania ze zbioru Z^k (w dowolnej kolejności) w przedziale $[0, C^*_{max}]$, korzystając z reguły Mc Naughtona.
3. Przydziel nadmiar (przekraczający $C^*_{max} - h$) długich zadań należących do Z^1 i być może krótsze zadania do m_1 procesorów w przedziale $[0, h]$. Uszereguj pozostałe zadania (lub ich część) w przedziale $[h, C^*_{max}]$ zgodnie z regułą Mc Naughtona.

Nietrudno zauważyć, że złożoność powyższego algorytmu jest $O(n_1 + n_k) = O(n)$.

3. Szeregowanie zadań 1- i k-procesorowych przy 0 - 1 żądaniach zasobowych

W paragrafie tym rozpatrzymy pewne szczególne przypadki ogólnego problemu szeregowanie zadań wieloprocesorowych przy ograniczeniach zasobowych, dla których istnieją proste algorytmy znajdujące uszeregowania o minimalnej długości.

Załóżmy, podobnie jak w paragrafie 2, że zadania należą do dwóch zbiorów Z^1 i Z^k, a w systemie znajduje się jeden rodzaj dodatkowego zasobu np. kanały. Żądania zasobowe zadań są zero-jedynkowe, to znaczy zadanie może żądać jednostki zasobu bądź nie żądać go wcale. Zatem zadania należące do zbioru Z^1 można podzielić na dwa podzbiory: zadań żądających jednostki zasobu i oznaczonych przez Z^{1r} i nie żądających dodatkowego zasobu oznaczonych przez Z^{10}. Podobnie zbiór Z^k dzieli się na dwa podzbiory: Z^{kr} oraz Z^{k0}.

Można wyróżnić następujące przypadki szczególne, dla ktorych stosunkowo

łatwo konstruuje się uszeregowania optymalne [6], wykorzystując podstawowy algorytm 1.

Przypadek 1. W systemie występują zadania należące do zbiorów Z^{kr}, Z^{kO} oraz Z^{1O}. W tym przypadku należy zastosować poniższy algorytm.

Algorytm 2

1. Podstaw

$$C:=\max\left\{Z/m\ ,(Y^O + Y^r)/\lfloor m/k\rfloor\ ,\ t^1_{max},\ t^k_{max},\ Y^r/r_1\right\},$$

gdzie $Y^O = \sum_{z_i \in Z^{kO}} t_i^{kO}$

$$Y^r = \sum_{z_i \in Z^{kr}} t_i^{kr}$$

2. Uszereguj zadania zgodnie z algorytmem 1 (zadania ze zbioru Z^{kr} są przydzielane jako pierwsze).

Przypadek 2. W systemie występują zadania należące do zbiorów Z^{kr} i Z^{1r}.

Poniższy algorytm znajduje uszeregowanie optymalne.

Algorytm 3

1. Podstaw

$$C:=\max\left\{Z/m,\ Y^r/\lfloor m/k\rfloor\ ,\ t^1_{max},\ t^k_{max},\ (Y^r + X^r)/r_1,\right\}$$

gdzie $X^r = \sum_{z_i \in Z^{1r}} t_i^{1r}$.

2. Przydziel zadania zgodnie z algorytmem 1.

Przypadek 3. W systemie występują zadania należące do podzbiorów Z^{kO} i Z^{1r}. W tym przypadku należy testować poniższy algorytm

Algorytm 4

1. Podstaw

$$C:=\max\left\{Z/m,\ Y^O/\lfloor m/k\rfloor\ ,\ t^1_{max},\ t^k_{max},\ X^r/r_1\right\}.$$

2. Przy obliczaniu C^*_{max} weź pod uwagę, że po prawej stronie linii h w uszeregowaniu A tylko r_1 procesów może wykonywać zadania ze zbioru Z^{1r}.
3. Uszereguj zadania zgodnie z algorytmem 1.

Nietrudno zauważyć, że powyższe algorytmy znajdują uszeregowanie optymalne dla odpowiednich przypadków, a ich złożoność wynosi O (n). Bar-

dziej skomplikowane problemy szeregowania,różniące się od powyższych rodzajem żądań zasobowych zadań,bądź stopniem ich wieloprocesorowości, należy rozwiązywać stosując ogólne podejście programowania liniowego, opisane w paragrafie 4.

4.Ogólny algorytm szeregowania.

W punkcie tym rozpatrzymy ogólny problem szeregowania zadań podzielnych wieloprocesorowych dla dowolnych ograniczeń i żądań zasobowych. Do jego rozwiązania wykorzystamy zmodyfikowane podejście opisane w [15] , służące do rozwiązania ogólnych zagadnień przydziału zasobów.

Zdefiniujmy zbiór zasobowo-dopuszczalny jako podzbiór zadań,które mogą być wykonywane równolegle ze względu na ograniczenia dotyczące procesorów i dodatkowych zasobów. Niech liczba różnych zasobowo-dopuszczalnych zbiorów będzie równa M. Przez x_i oznaczymy czas wykonywania i-tego zbioru zasobowo-dopuszczalnego,a przez Q_j zbiór indeksów tych zbiorów zasobowo-dopuszczalnych,które zawierają zadanie J_j

$$J_j \in \{Z_1^1,\ldots,Z_{n_1}^1,\ Z_1^2,\ldots,Z_{n_2}^2,\ldots,Z_1^k,\ldots,Z_{n_k}^k\}.$$

Stąd wynika następujący problem programowania liniowego:

Minimalizować

$$\sum_{i=1}^{M} x_i$$

przy ograniczeniach

$$\sum_{i \in Q_j} x_i = t_j^l \qquad J_j \in \{Z_1^l,\ldots,Z_{n_l}^l\},\ l = 1,2,\ldots,k$$

Rozwiązując powyższy problem programowania liniowego,uzyskujemy optymalne wartości x_i^* , długości części uszeregowania optymalnego. Zadania wykonywane w każdej z tych części uszeregowania są elementami odpowiadających zbiorów zasobowo-dopuszczalnych. Analizując złożoność obliczeniową powyższego podejścia widzimy,że liczba ograniczeń jest równa n, natomiast liczba zmiennych jest równa $O(n^m)$,zatem dla ustalonej liczby procesorów m, jest ograniczona przez wielomian od rozmiaru instancji (czyli od liczby zadań). Korzystając następnie z wielomianowych algorytmów programowania liniowego [7,10] uzyskujemy łącznie wielomianowy algorytm rozwiązania wyjściowego problemu szeregowania.

LITERATURA

[1] Aho A.V.,Hopcroft J.E.,Ullman J.D.: Projektowanie i analiza algorytmów komputerowych,PWN,Warszawa 1983.

[2] Avizienis A.: Fault tolerance: the survival attribute of digital systems,Proc. of the IEEE 66, No 10,1978,pp.1109-1125.
[3] Błażewicz J.,Drabowski M.,Węglarz J.: Scheduling independent 2-processor tasks to minimize schedule length, Information Processing Letters,18,No 5,1984,pp.267-273.
[4] Błażewicz J.,Drabowski M.,Węglarz J.: O niektórych problemach szeregowania zadań wieloprocesorowych, Zeszyty Naukowe Politechniki Śląskiej,s.Automatyka,No 74,1984,pp.39-48.
[5] Błażewicz J.,Drabowski M.,Węglarz J.: Scheduling multiprocessor tasks to minimize schedule length, IEEE Trans.on Computers,w druku.
[6] Błażewicz J.,Drabowski M.,Ecker K.,Węglarz J.: Multiprocessor task scheduling with single resource constraints, Proc. 1st European Workshop on Parallel Processing Techniques,Manchester,1985.
[7] Chaczijan L.G.: Polinomialnyj algoritm w liniejnom programirowanii, Dokłady Akademii Nauk SSSR 244,No 5,1979,pp.1093-1096.
[8] Coffman E.G.Jr.: Teoria szeregowania zadań, WNT,Warszawa,1980.
[9] Hakimi S.L.,Amin A.T.: Characterization of connection assignment of diagnosable systems, IEEE Trans.on Computers C-23,No 1,1974,pp.86-88.
[10] Karmarkar N.: A new polynomial-time algorithm for linear programming, Combinatorica 4, No 4,1984, pp.373-395.
[11] Krawczyk H.,Kubale M.; An approximation algorithm for diagnostic test scheduling in multicomputer systems, IEEE Trans. Comput. C-34,1985, pp.869-872.
[12] Mc Naughton R.: Scheduling with deadlines and loss functions, Management Sci.12,1959,p.1-12.
[13] Monma C.L.: A scheduling problem with simultaneous machine requirement, TIMS XXVI, Copenhagen 1984.
[14] Preparata,F.P.,Metze G.,Chien R.T.: On the connection assignment problem of diagnosable system, IEEE Trans.on Electronic Comput.EC-16, No 6,1967, pp.848-854.
[15] Węglarz J.,Błażewicz J.,Cellary W.,Słowiński R.: An automatic revised simplex method for constrained resource network scheduling, ACM Trans. on Mathematical Software 3,No.3,1977,pp.295-300.

Recenzent: Doc.dr h.inż.Józef Grabowski

Wpłynęło do Redakcji do 1986.04.30

СОСТАВЛЕНИЕ РАСПИСАНИЯ МНОГОПРОЦЕССОРНЫХ ЗАДАЧ ПРИ НАЛИЧЬИ ДОБАВОЧНЫХ РЕСУРСОВ

Р е з ю м е

В работе рассмотрена проблема составления расписания задач, которые необходимо выполнить на двух или более процессорах при наличьи добавочных ресуросов. Рассмотрена минимизация длины расписания для первичных задач.

SCHEDULING OF MULTIPROCESSOR TASKS UNDER RESOURCE CONSTRAINTS

S u m m a r y

The problem to be considered is one of scheduling of multiprocessor tasks under resource constraints. It is assumed that certain tasks may require more than one processor at a time .Tasks are to be scheduled preemtively in order to minimize schedule lenght.