ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1995
Seria: AUTOMATYKA z. 116 Nr kol. 1296

JAN WĘGLARZ
Politechnika Poznańska

DYSKRETNO-CIĄGŁE PROBLEMY SZEREGOWANIA - METODYKA ROZWIĄZYWANIA

Streszczenie. W pracy rozpatruje się problemy szeregowania, w których każde zadanie jednocześnie żąda maszyny ze zbioru identycznych maszyn równoległych oraz dowolnej ilości zasobu ciągłego, odnawialnego, z przedziału [0,1]. Zadania są niepodzielne i niezależne, kryterium stanowi długość uszeregowania. Opisano istotę podejścia dokładnego oraz dwie klasy algorytmów przybliżonych wraz z przykładowymi wynikami analizy najgorszego przypadku.

DISCRETE-CONTINUOUS SCHEDULING PROBLEMS - SOLUTION METHODOLOGY

Summary. This paper deals with scheduling problems in which each job simultaneously requires a machine from a set of parallel, identical machines, and an amount of a continuous, renewable resource, arbitrary within the interval [0,1]. Jobs are non-preemptible and independent, and the schedule length is to be minimized. An exact approach is described, and two classes of approximation algorithms are proposed with some worst-case results.

PROBLEMES D'ORDONNACEMENT DE TYPE DISCRET-CONTINU : UNE METHODOLOGIE DE RESOLUTION

Résumé. On considčre des problčmes d'ordonnancement où chaque tâche demande simultanement une machine d'un ensemble de machines identiques parallčles et une quantité de ressource renouvelable, divisible de façon continue dans l'intervalle [0,1]. Les tâches sont non-interruptibles et indépendantes, et le critčre d'ordonnancement est sa longueur. On décrit l'idée de l'approche exacte et deux classes d'algorithmes approximatifs avec une analyse exemplaire du pire cas.

1. Wstęp

Przez dyskretno-ciągłe problemy szeregowania rozumiemy problemy, w których każde zadanie jednocześnie żąda dla swego wykonywania zasobów dyskretnych oraz ciągłych. Przez zasób dyskretny rozumiemy przy tym zasób przydzielany w liczbach (numerach) jednostek z danego zbioru skończonego, a przez zasób ciągły - zasób przydzielany w dowolnych ilościach z danego przedziału. Ważną podklasę problemów dyskretno-ciągłych stanowią problemy, w których zadanie jednocześnie żąda do swego wykonania jednej maszyny z danego,

skończonego zbioru (ogólnie rozumianych) maszyn identycznych i równoległych (zasób dyskretny) oraz dowolnej ilości zasobu ciągłego, odnawialnego, z przedziału [0,1], gdzie 1 jest dostępną ilością tego zasobu. W praktyce z problemami takimi mamy do czynienia np. wówczas, gdy maszyny są zasilane ze wspólnego źródła mocy (elektrycznej, hydraulicznej czy pneumatycznej). Mogą to być np. dystrybutory paliwa zasilane ze wspólnej pompy [2]. Inne zastosowanie dotyczy wieloprocesorowych systemów komputerowych ze stronicowaną pamięcią wirtualną i wspólną pamięcią operacyjną, gdy liczba stron jest dostatecznie duża, by można traktować pamięć operacyjną jako zasób ciągły [6]. Wymieńmy jeszcze problemy, gdy zasób dyskretny stanowią pracownicy o określonych kwalifikacjach, a zasób ciągły - pieniądze. Problemy dyskretno-ciągłe stanowią ważny kierunek w teorii szeregowania zadań zarówno z powodu wspomnianego znaczenia praktycznego, jak i ze względu na walory poznawcze - wymagana jest tu bowiem inna metodyka postępowania niż w klasycznych problemach dyskretnych czy ciągłych. Dotyczy to zwłaszcza problemów, w których zadania są niepodzielne; dla zadań podzielnych wystarczy bowiem stosunkowo niewielka modyfikacja podejść klasycznych [1,6].

W tej pracy przedstawimy główne elementy tej metodyki i scharakteryzujemy dotychczas uzyskane wyniki, a także wskażemy kierunki dalszych badań.

2. Sformułowanie problemu

Załóżmy, że mamy n niepodzielnych, niezależnych zadań i=1,2,...,n, z których każde żąda do swego wykonywania w chwili t jednej, dowolnej maszyny ze zbioru m identycznych, równoległych maszyn j=1,2,...,m oraz, jednocześnie, ilości $u_i(t) \in [0,1]$ zasobu ciągłego, odnawialnego, dostępnego w ilości 1, tzn. $\sum_{i=1}^{n} u_i(t) \le 1$ dla każdego t.

W danej chwili każda maszyna może wykonywać co najwyżej jedno zadanie, a chwile dostępności zadań i maszyn są równe 0.

Prędkość wykonywania zadania i w chwili t w funkcji $u_i(t)$ wyraża się wzorem

$$dx_i(t)/dt = \dot{x}_i(t) = f_i[u_i(t)],\ x_i(0)=0,\ x_i(C_i)=\bar{x}_i \qquad (1)$$

gdzie $x_i(t)$ jest stanem zadania i w chwili t, f_i jest funkcją ciągłą, niemalejącą, $f_i(0)=0$, C_i jest nie znanym a priori momentem zakończenia wykonywania zadania i, a $\bar{x}_i$ jego znanym stanem końcowym, zwanym *rozmiarem*. Dokładną interpretację modelu (1) oraz uzasadnienie jego użycia znaleźć można np. w [1]. Jako kryterium optymalności przyjmiemy długość uszeregowania $M = \max_i(C_i)$.

Problem polega zatem na wyznaczeniu takiego przydziału maszyn i zasobu ciągłego do zadań, by każde zadanie zostało wykonane (tzn. dla zadania i został osiągnięty stan $\bar{x}_i$, i=1,2,...,n) przy spełnieniu przyjętych założeń (przydział taki będziemy nazywać *uszeregowaniem dopuszczalnym*) i by M przyjęło wartość minimalną (przydział taki będziemy nazywać *uszeregowaniem optymalnym*).

Zauważmy, że problem ten można zdekomponować na dwa powiązane z sobą podproblemy [4] :

(i) wyznaczenie przydziału zadań do maszyn,

(ii) wyznaczenie rozdziału zasobu ciągłego pomiędzy zadania wykonywane równocześnie w przydziale (i).

Oczywiście, rozdział zasobu ciągłego zależy od przydziału wyznaczonego w wyniku rozwiązania (i), a ostateczna długość uszeregowania dopuszczalnego zależy od rozdziału zasobu ciągłego. Dekompozycja powyższa wskazuje jednak na możliwość wykorzystania wyników osiągniętych dla klasycznych problemów szeregowania: dyskretnych (i) oraz ciągłych (ii). Różne realizacje tej możliwości będą przedmiotem dalszych rozważań. Zostaną one przedstawione w dwóch kolejnych rozdziałach, odpowiednio dla podejścia dokładnego (konstrukcja uszeregowań optymalnych). i dla podejść przybliżonych (konstrukcja uszeregowań dopuszczalnych o możliwie dobrej jakości). Wspomnijmy jeszcze, że w [2] powyższy problem (dla $f_i=f$, i=1,2,...,n) był rozpatrywany dla modeli zadań w postaci funkcji: prędkość wykonywania - liczba zadań równocześnie wykonywanych. Porównanie tego podejścia z podejściem opartym na modelu (1) można znaleźć w [7].

3. Podejście dokładne

Zacznijmy od przypadku $n \le m$, co sprowadza się do $n = m$, gdyż dla $n < m$, n-m maszyn byłoby nie wykorzystanych. W tym przypadku podproblem (i) jest trywialny : każde zadanie otrzymuje jedną maszynę. Pozostaje zatem do rozwiązania podproblem (ii) polegający na rozdziale zasobu ciągłego pomiędzy n zadań, czyli na wyznaczeniu takiej odcinkami ciągłej funkcji wektorowej $\underline{u}^*(t) = [u_1^*(t), u_2^*, \ldots, u_n^*(t)], u_i^*(t) \ge 0, \sum_{i=1}^{n} u_i^*(t) \le 1$, która zapewnia wykonanie każdego zadania i minimalną wartość M=M*. Wartości tej funkcji $\underline{u}^* = (u_1^*, u_2^*, \ldots, u_n^*)$ będziemy nazywać *optymalnymi rozdziałami zasobu* (ciągłego). Oznaczmy przez $U \in R^n$ zbiór wszystkich punktów $\underline{u}$, $u_i \ge 0$, i=1,2,...,n, spełniających nierówność $\sum_{i=1}^{n} u_i \le 1$ oraz przez V zbiór zdefiniowany następująco

$$v \in V \Leftrightarrow, u \in U, \text{ gdzie } v_i = f_i(u_i), \tag{2}$$

gdzie: f_i są funkcjami występującymi w (1).

Łatwo zauważyć, że zbiory U i V są zbiorami dopuszczalnych rozdziałów zasobu odpowiednio w układzie współrzędnych u i v.

Jak wiadomo (por. np.[5]), przy przyjętych założeniach słuszne jest następujące twierdzenie.

Twierdzenie 1

Minimalna długość uszeregowania w funkcji wektora rozmiarów zadań może być wyrażona wzorem

$$M^*(\bar{x}) = \min\{M > O : \bar{x} / M \in coV\} \tag{3}$$

gdzie coV jest powłoką wypukłą zbioru V. Funkcja ta jest zawsze wypukła.

Z powyższego twierdzenia wynika interpretacja geometryczna optymalnych rozdziałów zasobów w zależności od postaci funkcji f_i i sposób wyznaczania tych rozdziałów. W szczególności wynikają z niego następujące wnioski.

Wniosek 1

Dla f_i wklęsłych, i=1,2,...,n, długość uszeregowania M jest minimalizowana przez równoległe wykonywanie wszystkich zadań (stałymi) ilościami zasobu

$$u_i^* = f_i^{-1}(\bar{x}_i / M^*), i = 1,2,...,n \tag{4}$$

gdzie M* jest (jedynym) dodatnim pierwiastkiem równania

$$\sum_{i=1}^{n} f_i^{-1}(\bar{x}_i / M) = 1 \tag{5}$$

Wniosek 2

Dla $f_i \le c_i u_i$, $c_i = f_i(1)$, i=1,2,...,n, długość uszeregowania jest minimalizowana przez szeregowe wykonywanie pojedynczych zadań ilością zasobu równą 1.

Zauważmy, że Wniosek 2 oznacza w naszym przypadku, że do wykonania zbioru zadań w czasie M* wystarczy jedna maszyna.

Przejdźmy teraz do przypadku n>m. Zauważmy najpierw, że Wniosek 2 słuszny jest również w tym przypadku. Oznacza to, że dla $f_i \le c_i u_i$, $c_i = f_i(1)$, i=1,2,...,n (w szczególności dla f_i wypukłych) wyznaczenie uszeregowania optymalnego jest trywialne. Przypadek ten

wyeliminujemy zatem z dalszych rozważań, zauważając tylko, że nie występuje on w praktyce. Metodykę postępowania dla n>m przedstawimy natomiast na przykładzie f_i wklęsłych, i=1,2,...,n, choć jej główna idea ma charakter ogólny.

Zauważmy najpierw, że w każdym uszeregowaniu dopuszczalnym, w ogólnym przypadku, można zdefiniować $p < \infty$ przedziałów o długości Δ_k, k=1,2,...,p, takich że przydział zasobu do zadań w każdym z tych przedziałów jest stały. Niech z_k oznacza kombinację zadań odpowiadającą przedziałowi Δ_k, i niech Z będzie ciągiem kombinacji z_k, k=1,2,...,p, odpowiadającym uszeregowaniu dopuszczalnemu. Ciąg taki musi spełniać odpowiednie warunki, co prowadzi do następującej definicji.

Definicja 1

Ciągiem dopuszczalnym Z dla danego dyskretno-ciągłego problemu szeregowania nazywamy ciąg kombinacji zadań, $z_1, z_2,...,z_p$, $p<\infty$, spełniający następujące warunki

(a) $|z_k| \le m$, k=1,2,...,p

(b) $\bigwedge_i \bigvee_{z_k \in Z} i \in z_k$

(c) $\bigwedge_i \{k: i \in z_k\}$ jest albo zbiorem jednoelementowym, albo zbiorem kolejnych liczb całkowitych.

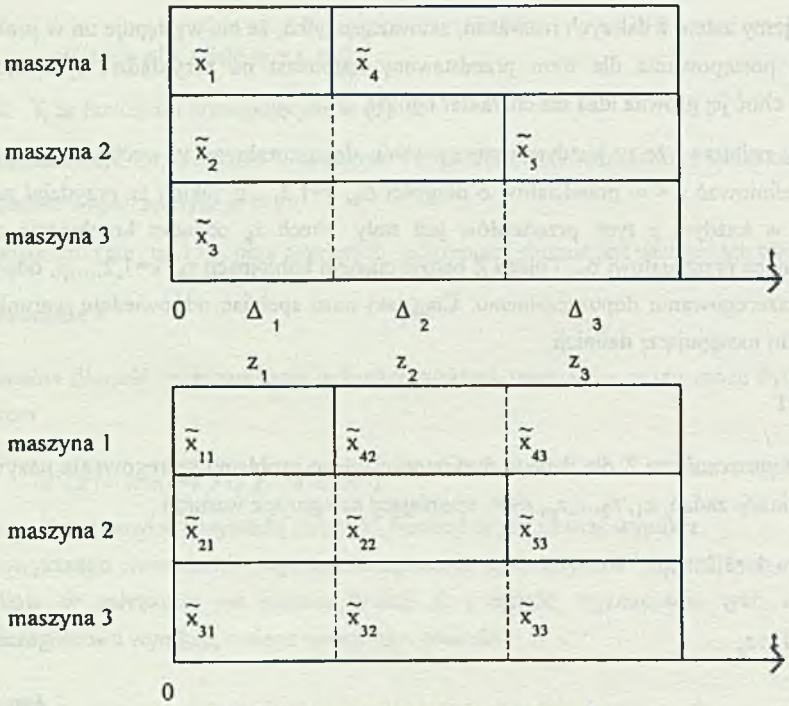
Warunki (a) i (b) są oczywiste, warunek (c) dotyczy niepodzielności zadań.

Zdefiniujmy teraz pewien zbiór ciągów dopuszczalnych.

Definicja 2

Potencjalnie optymalnym zbiorem ciągów dopuszczalnych ***POZ*** dla danego dyskretno-ciągłego problemu szeregowania nazywamy zbiór zawierający co najmniej jeden ciąg odpowiadający uszeregowaniu optymalnemu. Jest oczywiste, że dla danego problemu chcielibyśmy umieć utworzyć jak najmniejszy zbiór ***POZ***. Niestety, w ogólnym przypadku wiemy tylko tyle, że dla danego problemu zbiór wszystkich ciągów o maksymalnej liczbie maksymalnych (czyli m-elementowych) kombinacji jest ***POZ***. Taki ***POZ*** nazwiemy *maksymalnym*. W szczególności, jeśli f_i i=1,2,...,n, są wklęsłe, to łatwo zauważyć (por. Wniosek 1), że przedziały Δ_k zawsze mogą być zdefiniowane przez chwile zakończenia poszczególnych zadań. Oznacza to dalej (por. (c) w Definicji 1), że maksymalny ***POZ*** składa się z ciągów zawierających p = n-m+1 kombinacji (m-elementowych).

Dla danego ciągu dopuszczalnego Z możemy znaleźć optymalny rozdział zasobu ciągłego, wykorzystując wyniki uzyskane dla n = m. Ideę postępowania ilustruje rys. 1.



Rys.1. Idea rozdziału rozmiarów zadań
Fig.1. The idea of allocating processing demands of jobs

Problem polega na takim rozdziale rozmiarów zadań na części realizowane w poszczególnych kombinacjach z_k ciągu Z, który minimalizuje M. W tym celu wyznaczamy minimalne długości przedziałów Δ_k w funkcji rozmiarów częściowych $\Delta_k^*(\tilde{x}_{ik})_{i\in z_k}$, a następnie minimalizujemy sumę tych długości, pod warunkiem że rozmiar każdego zadania zostanie wykonany. Oznaczając przez K_i zbiór numerów tych kombinacji, które zawierają zadanie i, otrzymujemy zatem następujący problem programowania matematycznego :

Problem PM

Zminimalizować: $\sum_{k=1}^{p} \Delta_k^*(\tilde{x}_{ik})_{i\in z_k}$

przy ograniczeniach: $\sum_{k\in K_i} \tilde{x}_{ik} = \tilde{x}_i,\ i = 1,2,...,n$

$$\tilde{x}_{ik} \geq 0, i = 1,2,...,n; k \in K_i$$

Dla wyznaczenia $\Delta_k^*(\tilde{x}_{ik})_{i\in z_k}$ korzystamy z Wniosku 1 zastosowanego do z_k, tzn. znajdujemy jedyny dodatni pierwiastek równania

$$\sum_{i\in z_k} f_i^{-1}(\tilde{x}_{ik} / \Delta_k) = 1$$

Podkreślmy, że równanie to można rozwiązać analitycznie dla pewnych f_i, np. dla $f_i=c_i u_i^{\alpha_i}$ $c_i>0$, $\alpha_i \in \{1,2,3,4\}$, i=1,2,...,n. Zauważmy, że funkcja celu, jako suma funkcji wypukłych, jest zawsze wypukła.

Sformułowany wyżej problem PM w pewnych szczególnych przypadkach się upraszcza [4]. Zauważmy teraz, że rozwiązując ten problem dla każdego ciągu ze zbioru *POZ* i wybierając rozwiązanie o najmniejszej wartości funkcji celu, uzyskujemy uszeregowanie optymalne. Niestety, jak wspomnieliśmy, w ogólnym przypadku musimy korzystać z maksymalnego *POZ*, którego moc rośnie wykładniczo ze wzrostem n.

Podany wyżej ogólny sposób wyznaczania uszeregowania optymalnego nie wyczerpuje jednak możliwości znalezienia takiego uszeregowania. W pewnych przypadkach szczególnych możemy je bowiem znaleźć bardzo prosto. Ogólna idea polega na badaniu zależności między wektorem rozmiarów zadań $\underline{\bar{x}}$ a strukturą ciągów dopuszczalnych, przez którą rozumiemy liczby kombinacji, w których występują poszczególne zadania. Dla ilustracji tej idei rozpatrzymy prosty przykład.

Przykład

Niech n=5, m=3 i niech $f_i=f$, i=1,2,...,n będą wklęsłe. Rozpatrzmy ciąg dopuszczalny postaci Z={1,2,3},{2,3,4},{3,4,5}. Struktura tego ciągu ma postać

zadanie	1	2	3	4	5
$\lvert\{k:i \in z_k\}\rvert$	1	2	3	2	1

Niech będzie dany wektor $\underline{\bar{x}}$={10,20,30,20,10}. Wówczas podział rozmiarów zadań postaci :

$$x_{ik} = \bar{x}_i / |\{k:i \in z_k\}| \qquad (6)$$

jest podziałem optymalnym, gdyż zapewnia równomierne obciążenie każdej maszyny, co dla identycznych, wklęsłych f_i daje oczywiście minimalną długość uszeregowania.

Podobnie dla ciągu dopuszczalnego

Z={1,2,3},{1,2,4},{1,2,5}

którego struktura ma postać

zadanie	1	2	3	4	5
$\lvert\{k:i \in z_k\}\rvert$	3	3	1	1	1

oraz wektora $\underline{\bar{x}}$={30,30,10,10,10}, podział (6) zapewnia optymalność uszeregowania.

4. Algorytmy przybliżone

Znaczenie podejścia dokładnego, opisanego w p.3, polega na tym, że dla danego ciągu dopuszczalnego pozwala ono znaleźć uszeregowanie optymalne. Dla niezbyt licznych zbiorów *POZ* (maksymalnych *POZ*) daje ono także możliwość znalezienia takiego uszeregowania na drodze rozwiązania Problemu PM dla każdego ciągu z tego zbioru. W ogólności jednak podejście to jest obliczeniowo zbyt kosztowne i musimy stosować algorytmy przybliżone. Opisana w poprzednich punktach metodyka sugeruje różne podejścia do konstrukcji takich algorytmów. Poniżej scharakteryzujemy krótko dwa z nich [7].

W pierwszym, oznaczonym $H_1(h)$, znajdowany jest ciąg dopuszczalny na drodze rozwiązania klasycznego problemu szeregowania dla czasów wykonywania zadań wynikających z ilości zasobu wyznaczonych przez funkcję h. Jest to więc faktycznie klasa algorytmów zależna od h. Przebieg algorytmu dla danego h jest następujący.

Algorytm $H_1(h)$

1. Zdefiniuj funkcję $h:\{1,2,...,n\}\rightarrow[0,1/m]$ taką, że $h(i)=u_i$, $i=1,2,...,n$;
2. Oblicz czasy wykonywania zadań jako
$$\tau_i = \tilde{x}_i / f_i(u_i)$$
3. Znajdź uszeregowanie minimalizujące M dla zbioru n zadań o czasach τ_i na m maszynach.
4. Rozwiąż, ze względu na M i u_j, $j=1,2,...,m$, następujący układ równań
$$\sum_{j=1}^{m} u_j = 1$$
$$\sum_{i \in I_j} \tilde{x}_i / f_i(u_j) = M, j = 1,2,...,m$$
$$u_j \geq 0, j = 1,2,...,m,$$
gdzie: I_j jest zbiorem zadań przydzielonych do maszyny j w uszeregowaniu znalezionym w kroku 3;
5. Przydziel ilości zasobu wyznaczone w kroku 4, równe dla zadań wykonywanych na danej maszynie, do każdego zadania i znajdź czasy wykonywania zadań w ostatecznym uszeregowaniu o długości M.

W szczególności, jeśli przyjąć $h(i)=h_1(i)=1/m$, $i=1,2,...,n$, to można wykazać następujące twierdzenia[7].

Twierdzenie 2

Rozpatrzmy dyskretno-ciągły problem szeregowania z funkcjami $f_i=c_iu_i^{1/\alpha}$, $\alpha>1$, $c_i>0$ i=1,2,...,n oraz m=2. Wówczas algorytm $H_1(h_1)$ znajduje uszeregowanie optymalne.

Twierdzenie 3

Rozpatrzmy problem szeregowania jak w Tw.2 dla dowolnego ustalonego m. Wówczas, jeśli uszeregowanie znalezione w kroku 3 Algorytmu $H_1(h_1)$ spełnia warunek

$$\bigwedge_j C_j = C$$

to jest to uszeregowanie optymalne.

Twierdzenie 4

Rozpatrzmy problem szeregowania jak w Tw.2 dla $m \geq 3$. Wówczas

$$\frac{M^{H_1(h_1)}}{M^*} < 2, \quad \lim_{\alpha\to\infty} \frac{M^{H_1(h_1)}}{M^*} = 1, \quad \lim_{m\to+\infty} \frac{M^{H_1(h_1)}}{M^*} = \sqrt[q]{2}$$

gdzie $M^{H_1(h_1)}$ jest długością uszeregowań wyznaczonych przez Algorytm $H_1(h_1)$, a M* długością uszeregowania optymalnego.

W drugim podejściu konstruowany jest za pomocą pewnego algorytmu A zbiór Z ciągów dopuszczalnych, a następnie znajdowane jest uszeregowanie optymalne dla każdego ciągu $Z\in Z$ i wybierane jest uszeregowanie o minimalnej długości. Rodzinę algorytmów opartych na tym podejściu oznaczymy przez $H_2(A)$.

Jako prosty przykład algorytmu A rozważmy następujący algorytm A_1.

Algorytm A_1

1. Utwórz ciąg liczb całkowitych c=1,2,...,n; l:=1;
2. Utwórz ciąg dopuszczalny Z, w którym $z_1=\{1,2,...,m\}$, a z_{k+1} powstaje z z_k przez usunięcie jego l-tego wyrazu i dołączenie (m+k)-tego wyrazu ciągu c;l:=l+1;
3. Każdy wyraz ciągu c pojawia się w s różnych kombinacjach z_k. Uszereguj wyrazy ciągu c według nierosnących s, a zadania według nierosnących $\bar{x}_i / f_i(1/m)$. Przydziel zadania do odpowiednich wyrazów ciągu c.
4. Jeśli $l \leq m$, to wróć do 2, w przeciwnym razie stop.

Łatwo zauważyć, że Algorytm A_1 realizuje wskazówkę (por. Przykład), że zadania o większych rozmiarach powinny być realizowane w większej liczbie kombinacji. Wskazówka ta jest wykorzystana do konstrukcji zbioru ciągów dopuszczalnych.

Na podstawie klasycznych wyników [3] łatwo wykazać, że słuszne są następujące oszacowania.

Twierdzenie 5

Rozpatrzmy dyskretno-ciągły problem szeregowania z funkcjami $f_i=f=u^{1/\alpha}$, $\alpha>0$, i=1,2,...,n.

Wówczas

$$\frac{M(H_1(A_1))}{M^*} \leq 2-\frac{1}{m}$$

Twierdzenie 6

Rozpatrzmy dyskretno-ciągły problem szeregowania jak w Tw.5 dla m=2. Wówczas

$$\frac{M(H_1(A_1))}{M^*} \leq \frac{4}{3}$$

5. Uwagi końcowe

Opisana metodyka postępowania, choć dla prostoty przedstawiona dla podklasy dyskretno-ciągłych problemów szeregowania, może być uogólniona w różnych kierunkach. Po pierwsze, można rozpatrzyć problemy, w których zadania mają różne momenty gotowości do wykonywania i/lub terminy zakończenia. Po drugie, można zbadać inne kryteria szeregowania, w szczególności średni czas przepływu i maksymalne opóźnienie. Po trzecie, zamiast (lub oprócz) maszyn można rozpatrzyć inne zasoby dyskretne, a zamiast (lub oprócz) zasobu ciągłego odnawialnego, zasób (zasoby) podwójnie ograniczony.

LITERATURA

[1] Błażewicz, J., Cellary, W., Słowiński, R.,, Węglarz, J.: Scheduling Under Resource Constraints-Deterministic Models. J.C.Baltzer, Basel, 1986.

[2] Dror, M., Stern, H.J., Lenstra, J.K.: Parallel machine scheduling : processing rates dependent on number of jobs in operation. *Management Sci.* **33**, No.8, 1987, 1001-1009.

[3] Graham, R.L.: Bound on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics* **17**, 1969, 416-429.

[4] Józefowska, J., Węglarz, J.: On a methodology for discrete-continuous scheduling. Report of the Inst. of Computing Science, Poznań Univ. of Technology, RA-95/001, Poznań 1995.

[5] Węglarz, J.: Time-optimal control of resource allocation in a complex of operations framework. *IEEE Trans. Systems, Man. and Cybernet.* **SMC-6**, No.11, 1976, 783-788.

[6] Węglarz, J.: Multiprocessor scheduling with memory allocation - a deterministic approach. *IEEE Trans. Computer* **C-29**, No.8, 703-710.

[7] Weglarz, J.: O pewnych dyskretno-ciągłych problemach szeregowania. *Zeszyty Naukowe Politechniki Śląskiej s.Automatyka* , z.109, 1992, 311-319.

Recenzent: Dr hab. inż. Prof. Pol. Śl. Jan Kałuski

Wpłynęło do Redakcji do 30.06.1995 r.

Abstract

This paper presents a methodology for solving discrete-continuous scheduling problems which arise, e.g., when each job simultaneously requires for its processsing a machine from a set of identical, parallel machines, and an amount (unknown in advance) of a continuous, renewable resource. We assume that jobs are nonpreemptible, independent, available at the start of the process, and that the schedule length is to be minimized. Jobs are characterized by their processing demands and by the functions : processing rate vs. resource amount allotted at time t. We present an exact approach based on nonlinear programming, and approximate algorithms for which some bounds are given.

The presented methodology can be generalized in a number of directions.