ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1990
Seria: AUTOMATYKA z.100 Nr kol.1082

Marek Kubale
Politechnika Gdańska

PODZIELNE USZEREGOWANIA ZADAŃ DWUPROCESOROWYCH NA PROCESORACH DEDYKOWANYCH

Streszczenie. W pracy rozważono złożoność obliczeniową problemu podzielnego szeregowania zadań dwuprocesorowych na procesorach dedykowanych. Pokazano, że rozważany problem jest NP-zupełny w przypadku ogólnym. Zbadano możliwość zastosowania programowania liniowego i kolorowania grafów do efektywnego otrzymywania uszeregowań optymalnych w przypadkach szczególnych.

1. Wstęp

Jednym z założeń przyjmowanych w teorii szeregowania zadań było do niedawna założenie, iż w dowolnej chwili czasu każde zadanie może być realizowane na co najwyżej jednym procesorze. Jednakże, w dobie burzliwego rozwoju technologii półprzewodnikowej, założenie to nie ma racji bytu w niektórych zastosowaniach mikroprocesorowych. Należy do nich np. szeregowanie transmisji plikowych w sieciach komputerowych [3] i wykonywanie testów diagnostycznych w systemach wieloprocesorowych [7]. We wszystkich tych przypadkach - oprócz założeń przyjmowanych powszechnie w problematyce szeregowania zadań - zakłada się, że każde zadanie musi być realizowane jednocześnie na z góry określonym zbiorze procesorów.

Niestety ogólny problem szeregowania podzielnych zadań wieloprocesorowych w celu minimalizacji długości uszeregowania na procesorach dedykowanych jest NP-trudny. Jest więc praktycznie niemożliwe, aby mógł być rozwiązany w czasie wielomianowym. Jednakże, z uwagi na potencjalne zastosowania w dyskretnych procesach przemysłowych, jesteśmy zainteresowani znalezieniem możliwie wielu przypadków szczególnych, które dają się rozwiązać za pomocą algorytmów wielomianowych. Dlatego w pracy tej narzucamy pewne ograniczenia na instancje problemu szeregowania zadań podzielnych. Jednym z nich jest założenie, że każde zadanie wymaga dwóch określonych procesorów jednocześnie.

W niniejszej pracy zbadamy złożoność obliczeniową tak sformułowanego problemu szeregowania zadań w przypadku ogólnym oraz niektórych przypadkach szczególnych. W punkcie 2 zdefiniujemy pewne pojęcia pomocnicze i wykażemy NP-zupełność takiego szeregowania zadań. W punkcie 3 przeanalizujemy możliwość zastosowania programowania liniowego do znajdowania usze-

regowań optymalnych w czasie wielomianowym. Kolejny punkt 4 poświęci sposobowi efektywnego wyznaczania uszeregowań optymalnych metodą przedzi łowego multikolorowania krawędzi grafu. Na zakończenie rozważymy pewne r lacje zachodzące pomiędzy szeregowaniem podzielnym i niepodzielnym.

2. NP-zupełność problemu szeregowania

Aby przedstawić bardziej formalnie cel naszych rozważań, wprowadzimy n stępujący model matematyczny. Niech będzie dany zbiór n zadań $Z = \{z_1,\ldots, z_n\}$, które mają być wykonane na m procesorach tworzących zbiór $P = \{p_1,\ldots,p_m\}$. Przyjmujemy, że zadania te są niezależne w tym sensie, że nie ma ograniczeń kolejnościowych pomiędzy elementami zbioru Z. Każde zadanie z_i wymaga dwóch dedykowanych procesorów $p_j, p_k \in P$ przez okres czasu równy w sumie $t_i = t(z_i)$ jednostek czasu. W odróżnieniu od modelu rozważanego w [8] przyjmujemy, że każde zadanie z_i może być przerwane i następnie kontynuowane po pewnym czasie, lecz na tej samej parze procesorów. Zakładamy, że momenty rozpoczynania i kończenia zadań są wielokrotnościami podstawowej jednostki czasu. Jako kryterium jakości uszeregowania przyjmujemy długość uszeregowania, czyli moment zakończenia tego zadania, które kończy się najpóźniej.

Rozważany problem możemy przedstawić w postaci grafu nieskierowanego $G = (V,E)$ z obciążonymi krawędziami. Zbiór P odpowiada zbiorowi V wierzchołków grafu, zbiór Z odpowiada zbiorowi krawędzi E, tzn. dwa wierzchołki $v_i, v_j \in V$ połączone są krawędzią $e \in E$, gdy istnieje zadanie do wykonania na odpowiednich procesorach $p_i, p_j \in P$. Waga $w(e)$ krawędzi e jest sumarycznym czasem trwania zadań na procesorach p_i i p_j. Graf taki nazywać będziemy grafem szeregowania. Łatwo zauważyć, że rozwiązanie naszego problemu odpowiada multikolorowaniu krawędzi grafu szeregowania odpowiednimi przedziałami czasu i na odwrót. Jest tak dlatego, że każdy kolor można zinterpretować jako jednostkę czasu, w trakcie której wykonują się odpowiednie zadania dwuprocesorowe. Przypomnijmy tutaj, że graf G jest (r,s)-barwny krawędziowo, jeśli każdej krawędzi możemy przydzielić r barw ze zbioru s dostępnych kolorów w ten sposób, że krawędzie sąsiednie otrzymują rozłączne podzbiory kolorów. W dalszej części będzie nam potrzebny

Lemat 1. Jeśli r i s są liczbami naturalnymi spełniającymi $s > 2r$, to (r,s)-BARWNOŚĆ KRAWĘDZIOWA jest problemem NP-zupełnym. ∎

Lemat, którego formalny dowód pomijamy, można wyprowadzić z faktu, że multikolorowanie wierzchołków grafu krawędziowego L(G) odpowiada multikolorowaniu krawędzi grafu G, zaś ten ostatni problem jest tak samo trudny jak zwykłe kolorowanie krawędzi na mocy twierdzenia Farra [4]. Na przykład biorąc dowolny graf stopnia 10 możemy zapytać, czy indeks chroma

tyczny $\chi'(G) = 10$. Kwestię tę można sprowadzić do pytania o to, czy istnieje (2,5)-kolorowanie wierzchołkowe grafu $L^2(G)$ powstałego z $L(G)$ w wyniku redukcji Farra.

Jak wiadomo, problem szeregowania zadań podzielnych na procesorach niededykowanych może być rozwiązany w czasie liniowym [1]. Obecnie pokażemy, że problem ten staje się NP-zupełny, gdy procesory są dedykowane.

Rozważmy następujący problem decyzyjny: "Dane są zbiory Z i P oraz stała l i pytamy, czy istnieje uszeregowanie zadań Z na procesorach P o długości nie przekraczającej l?" Oznaczmy ten problem mianem 2-PROCESOROWEGO SZEREGOWANIA PODZIELNEGO lub krótko 2PSP. Udowodnimy następujące

Twierdzenie 1. Problem 2PSP jest NP-zupełny nawet wówczas, gdy Z jest zbiorem bez powtórzeń, którego wszystkie elementy mają identyczny czas przetwarzania.

Dowód. Dowodzimy poprzez sprowadzenie problemu BARWNOŚCI KRAWĘDZIOWEJ do 2PSP. Dla danego grafu $G = (V,E)$ i liczb (r,s) tworzymy instancję problemu szeregowania w następujący sposób:

1. Przyjmujemy $m = |V|$, $n = |E|$, $l = s$.
2. Graf G traktujemy jako graf szeregowania z wszystkimi zadaniami z_i długości $t(z_i) = r$.

Łatwo zauważyć, że podzielne uszeregowanie zadań dwuprocesorowych na procesorach dedykowanych, którego długość nie przekracza l istnieje wtedy i tylko wtedy, gdy graf G jest (r,s)-barwny krawędziowo. ■

3. Szeregowanie zadań metodą programowania liniowego

Istnieją dwa sposoby otrzymywania przypadków wielomianowych. Pierwszy polega na sformułowaniu problemu szeregowania podzielnego jako zadania programowania liniowego. Jeśli liczba procesorów jest stała, zastosowanie algorytmu Chaczijana lub Karmarkara prowadzi do rozwiązania optymalnego w czasie ograniczonym przez wielomian zmiennej n. Drugi, bardziej efektywny sposób otrzymywania przypadków wielomianowych polega na wyrażeniu naszego problemu szeregowania jako przedziałowego multikolorowania krawędzi grafu. To drugie podejście rozwiniemy w następnym punkcie.

Niech

$$z_1, z_2, \ldots, z_n \tag{1}$$

będzie uporządkowaniem zadań (krawędzi), a

$$M_1, M_2, \ldots, M_l \tag{2}$$

uporządkowaniem wszystkich skojarzeń grafu szeregowania (czyli takich podzbiorów zadań, które mogą być realizowane jednocześnie). Wprowadzamy zmien-

ną $x_{ij} \in \{0,1\}$, która przyjmuje wartość 1, gdy $z_i \in M_j$ i 0 w przypadku przeciwnym. Niech τ_j oznacza czas wykonywania j-tego podzbioru zadań. Wówczas długość optymalnego uszeregowania wyraża się jako

$$\min \sum_{j=1}^{l} \tau_j \tag{3}$$

przy ograniczeniach

$$\tau_j \geq 0 \qquad j = 1,\dots,l \tag{4}$$

$$\sum_{j=1}^{l} x_{ij}\tau_j = t_i \qquad i = 1,\dots,n \tag{5}$$

Rozwiązując problem (3)÷(5) za pomocą algorytmu programowania liniowego otrzymujemy czasy cząstkowe τ_j wykonania poszczególnych podzbiorów zadań.

Oszacujmy złożoność obliczeniową tego podejścia. Liczba zmiennych wynosi l, a ograniczeń n+l, gdzie $l \leq n^{m/2}$. Z drugiej strony złożoność algorytmu elipsoidalnego Chaczijana [2] jest ograniczona od góry przez wielomian zmiennej rozmiaru problemu, czyli n+l. Zatem przy ustalonej liczbie procesorów (co w praktyce ma zwykle miejsce) rozważany problem może być rozwiązany w czasie wielomianowym. Jednakże stopień i stała proporcjonalności tego wielomianu zmiennej n są wysokie, przez co metoda ta ma znaczenie głównie teoretyczne. Dlatego w następnym punkcie rozważymy inne sposoby otrzymywania uszeregowań optymalnych, które działają efektywnie dla każdej liczby procesorów.

4. Szeregowanie zadań metodą kolorowania grafu

Jak wiemy, optymalne multikolorowanie krawędzi grafu szeregowania wyznacza rozwiązanie naszego problemu szeregowania. Poniżej opiszemy kilka algorytmów multikolorowania krawędzi, które działają skutecznie na grafach o niskiej liczbie chromatycznej.

Niech G będzie grafem dwudzielnym i niech ϱ oznacza stopień ważony grafu G, czyli maksymalną sumę wag krawędzi incydentnych z wierzchołkiem. Poniżej pokażemy za [6], w jaki sposób można pomalować krawędzie grafu G ϱ kolorami w czasie kwadratowym.

Twierdzenie 2. Jeśli G jest grafem dwudzielnym, to jego krawędzie można pomalować w czasie $O(n^2)$.

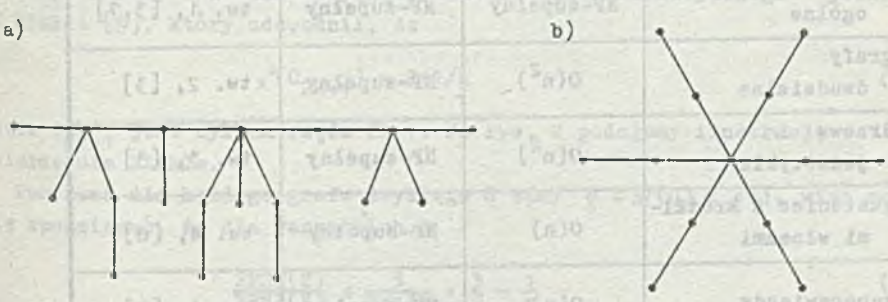
Dowód. Najpierw obliczamy wartość ϱ, a następnie budujemy nadgraf dwudzielny H grafu G z n wierzchołkami po każdej stronie. W grafie H mamy te same krawędzie, które występują w G i dodatkowo pewne nowe krawędzie wprowadzone po to, aby uczynić wszystkie stopnie równymi ϱ. Po zbudowaniu H znajdujemy maksymalne skojarzenie w tym grafie. Niech $M = \{e_1, e_2, \dots, e_n\}$

będzie zbiorem krawędzi tworzących to skojarzenie i niech $\beta = \min w(e_i)$, $i = 1,\ldots,n$. Obecnie wagi wszystkich krawędzi w M zmniejszamy o β. Oznacza to usunięcie z grafu B przynajmniej jednej krawędzi. Proces ten powtarzamy dopóki wszystkie krawędzie nie zostaną usunięte z H. Ponieważ graf G ma n krawędzi, maksymalne skojarzenie musi być wyznaczone najwyżej n razy. Zatem liczba przerwań jednego zadania nie przekracza $O(n)$. Pierwsze skojarzenie można wyznaczyć w czasie $O(n\sqrt{n})$. Każde następne skojarzenie otrzymuje się w czasie liniowym stosując technikę dróg naprzemiennych. Zatem łączny czas działania algorytmu jest $O(n^2)$. ■

Obecnie niech G będzie drzewem lub grafem jednocyklicznym, tzn. drzewem, do którego dodano jedną krawędź pomiędzy dowolnymi wierzchołkami niesąsiednimi. Oczywiście, jeśli G jest drzewem lub grafem jednocyklicznym zawierającym cykl parzysty, to jest on pewnym grafem dwudzielnym i jako taki może być pomalowany w wyżej opisany sposób. Załóżmy zatem, że jedyny cykl grafu ma nieparzystą liczbę krawędzi. Wówczas musimy postępować odmiennie.

Twierdzenie 3. Jeśli G jest grafem jednocyklicznym z cyklem nieparzystym, to jego krawędzie można pomalować w czasie $O(n^2)$.

Dowód. Najpierw znajdujemy cykl C grafu G i kolorujemy jego krawędzie w sposób optymalny. W tym celu każdą krawędź o wadze w(e) traktujemy jako multikrawędź złożoną z w(e) krawędzi równoległych i stosujemy algorytm podobny do opisanego w [3]. Metoda ta wymaga znalezienia $O(n)$ różnych skojarzeń w C, przy czym po wyznaczeniu kolejnego skojarzenia zmniejsza się w odpowiedni sposób wagi krawędzi należących do tego skojarzenia. Zatem złożoność tej fazy algorytmu nie przekracza $O(n^2)$. Następnie kolorujemy drzewa wyrastające z cyklu. Poczynając od dowolnego wierzchołka v cyklu C kolorujemy bezbarwne krawędzie incydentne z v w sposób zachłanny. Następnie przechodzimy do sąsiada wierzchołka v i kolorujemy zachłannie pozostałe nie pomalowane krawędzie. Kontynuując to podejście trawersujemy graf G metodą pogłębiania i kolorujemy zachłannie napotkane krawędzie. Liczba przerwań jednego zadania nie przekracza $O(n)$, zaś czas działania drugiej fazy, tak jak i całego algorytmu, wynosi $O(n^2)$. ■



Rys. 1. Przykłady grafów: a) gąsienica, b) supergwiazda.
[Examples of graphs: a) caterpillar, b) superstar]

Kwadratową złożoność algorytmu możemy obniżyć w istotnym stopniu, gdy G jest drzewem o uproszczonej strukturze. Mianowicie, jeżeli G jest tzw. gąsienicą z krótkimi włosami (tj. długości ≤ 2) lub wręcz supergwiazdą, to odpowiednie pokolorowanie można uzyskać w czasie liniowym. Przykłady takich grafów pokazujemy na rys. 1.

Twierdzenie 4. Jeśli G jest gąsienicą z krótkimi włosami, to optymalne pokolorowanie można otrzymać w czasie O(n).

Dowód. Najpierw obliczamy wartość ρ. Następnie kolorujemy korpus gąsienicy poczynając od dowolnego końca i malując kolejne krawędzie na przemian kolorami $\{1,2,\ldots\}$ i $\{\ldots,\rho-1,\rho\}$, tzn. najniższymi możliwymi i najwyższymi możliwymi. W ostatniej fazie kolorujemy zachłannie włosy gąsienicy w kierunku od korpusu. Opisany algorytm ma liniową złożoność obliczeniową, przy czym każda krawędź otrzymuje najwyżej dwa zbiory kolorów. ∎

Supergwiazda jest szczególnym typem gąsienicy, w którym korpus redukuje się do jednego wierzchołka, zaś wszystkie włosy są długości 2. Dlatego liniowe pokolorowanie jej promieni jest sprawą trywialną.

5. Uwagi końcowe

Rozważania nasze podsumowujemy w postaci tabeli 1. Kolejne wiersze tej tabeli poświęcone są grafom o coraz prostszej strukturze. Dla porównania złożoności obliczeniowej wprowadzamy kolumnę 2PSN, przez co rozumiemy problem decyzyjny 2-PROCESOROWEGO SZEREGOWANIA NIEPODZIELNEGO na procesorach dedykowanych.

Tabela 1. Złożoność obliczeniowa problemu szeregowania zadań na procesorach dedykowanych.

Grafy szeregowania	Problemy		Referencje
	2PSP	2PSN	
grafy ogólne	NP-zupełny	NP-zupełny	tw. 1, [3,7]
grafy dwudzielne	$O(n^2)$	NP-zupełny	tw. 2, [3]
drzewa i jednocykle	$O(n^2)$	NP-zupełny	tw. 3, [3]
gąsienice z krótkimi włosami	O(n)	NP-zupełny	tw. 4, [8]
supergwiazdy	O(n)	NP-zupełny	tw. 4, [8]

Z tabeli 1 wynika, że dopuszczenie możliwości dzielenia zadań prowadzi do radykalnego uproszczenia problemu. Co więcej, uszeregowania podzielne są zwykle krótsze od odpowiednich uporządkowań niepodzielnych. Niech będzie dany zbiór zadań Z do wykonania na procesorach dedykowanych ze zbioru P. Przez 2PSP(Z) oznaczymy długość optymalnego uszeregowania podzielnego, a przez 2PSN(Z) długość optymalnego uszeregowania niepodzielnego tych zadań. Ponieważ $2PSP(Z) \leq 2PSN(Z)$, więc

$$1 \leq \frac{2PSN(Z)}{2PSP(Z)} \tag{6}$$

dla wszystkich Z.

Dolne oszacowanie tego stosunku jest ścisłe w tym sensie, że istnieją grafy szeregowania, dla których 2PSN(Z) = 2PSP(Z) dla każdej liczby procesorów. Przykłady takich grafów występują zarówno wśród grafów rzadkich, jak i gęstych. Z jednej strony są to gwiazdy, z drugiej grafy pełne. To ostatnie stwierdzenie wymaga pewnego uzasadnienia.

Niech $\chi^*(G)$ oznacza <u>indeks multichromatyczny</u> grafu G, tzn. liczbę zdefiniowaną jako

$$\chi^*(G) = \inf \chi'(G^k)/k \tag{7}$$

gdzie G^k jest multigrafem otrzymanym z G przez zastąpienie każdej jego krawędzi zbiorem k równoległych krawędzi. Jest oczywiste, że

$$\chi^*(K_{2k}) = \chi'(K_{2k}) = 2k-1 \tag{8}$$

gdzie K_{2k} jest 2k-wierzchołkowym grafem pełnym. Natomiast Fiorini i Wilson [5] udowodnili, że

$$\chi^*(K_{2k+1}) = \chi'(K_{2k+1}) = 2k+1 \tag{9}$$

A zatem, jeśli wszystkie zadania $z_i \in Z$ mają jednakowy czas przetwarzania i graf szeregowania jest pełny, to 2PSP(Z) = 2PSN(Z).

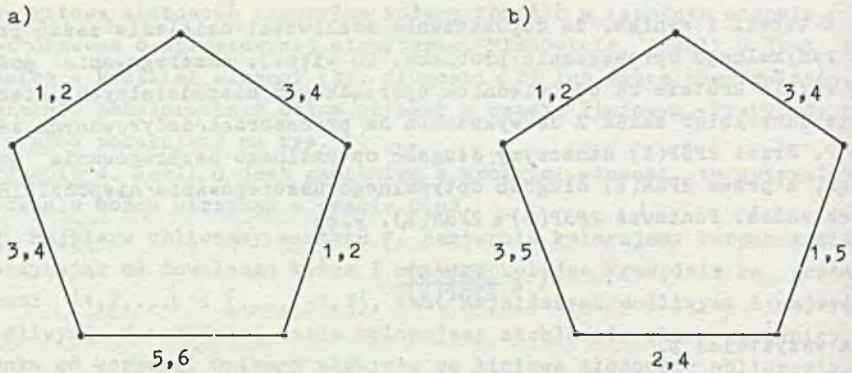
Inny ciekawy rezultat dotyczący indeksu multichromatycznego pochodzi od Stahla [9], który udowodnił, że

$$\chi^*(C_{2k+1}) = 2+1/k \tag{10}$$

gdzie C_{2k+1} jest cyklem rzędu 2k+1. Na rys. 2 podajemy ilustrację dla twierdzenia Stahla.

Ponieważ dla każdego grafu zwykłego G mamy $\varrho \leq \chi'(G) \leq \varrho+1$, więc można się spodziewać, że dla wszystkich Z

$$\frac{2PSN(Z)}{2PSP(Z)} \leq \frac{3}{2+1/k} < \frac{3}{2} - \frac{1}{m} \tag{11}$$



Rys. 2. Graf C_5 pomalowany: a) niepodzielnie, b) podzielnie.
[Graph C_5 colored: a) nonpreemptively, b) preemptively]

LITERATURA

[1] Błażewicz J., Drabowski M., Węglarz J.: Scheduling multiprocessor tasks to minimize schedule length, IEEE Trans. Comput. C-35, 1986, 389-393.

[2] Chaczijan L.D.: Algorytm wielomianowy dla programowania liniowego (po rosyjsku), Dokł. Akad. Nauk ZSRR 244, 1979, 1093-1096.

[3] Coffman E.G., Jr., Garey M.R., Johnson D.S., LaPaugh A.S.: Scheduling file transfers, SIAM J. Comput. 14, 1985, 744-780.

[4] Farr G.: The complexity of multicoloring, Disc. Math. 69, 1988, 219-223.

[5] Fiorini S., Wilson R.J.: Edge-Colourings of Graphs, Pitman, London, 1977.

[6] Gonzales T., Sahni S.: Open shop scheduling to minimize finish time, J. ACM 23, 1976, 665-679.

[7] Krawczyk H., Kubale M.: An approximation algorithm for diagnostic test scheduling in multicomputer systems, IEEE Trans. Comput. C-34, 1985, 869-872.

[8] Kubale M.: The complexity of scheduling independent two-processor tasks on dedicated processors, Inf. Process. Lett. 24, 1987, 141-147.

[9] Stahl S.: n-tuple colorings and associated graphs, J. Combin. Theory Ser. B 20, 1976, 185-203.

Recenzent: Doc.dr h.inż.J.Klamka

Wpłynęło do Redakkcji do 1990-04-30.

PREEMPTIVE SCHEDULING OF TWO-PROCESSOR TASKS ON DEDICATED PROCEDURE

S u m m a r y

The paper considers the computational complexity of the problem of preemptive scheduling of 2-processor tasks on dedicated processors. It has been shown that in general the problem is NP-complete. He possibility of using a linear programming and graph coloring approach for an efficient construction of the optimal schedulles in some special cases has been analysed.

ДЕЛИМЫЕ УПОРЯДОЧЕНИЯ ДВУХПРОЦЕССОРНЫХ ЗАДАЧ ДЛЯ НАЗНАЧЕННЫХ ПРОЦЕССОРОВ

Р е з ю м е

В работе рассмотрена вычислительная сложность проблемы делительного упорядочения двухпроцессорных задач для назначенных процессоров. Доказано, что в общем случае рассматриваемая задача является НП - полной проблемой. Рассмотрена возможность применения линейного программирования и раскраски графов для эффективного получения оптимального упорядочения в частных случаях.