ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: AUTOMATYKA z.114 Nr kol. 1250

Konrad PIWAKOWSKI
Politechnika Gdańska

SZEREGOWANIE ZADAŃ W OTWARTYM SYSTEMIE DUOPROCESOROWYM

Streszczenie: W niniejszej pracy rozważamy szczególny przypadek otwartego systemu obsługi, dla którego przedstawiamy optymalny algorytm wielomianowy. Ograniczenie w stosunku do ogólnie sformułowanego problemu "Open Shop", który jest NP-trudny, polega na przyjęciu założenia, że czas wykonywania poszczególnej operacji zależy jedynie od procesora, na którym jest ona wykonywana i może przyjmować jedną z dwóch wartości: 1 (na szybkich procesorach) lub L (na wolniejszych), gdzie $L \in N$.

OPEN SHOP SCHEDULING IN DUOPROCESSOR SYSTEM

Summary: A particular case of the Open-Shop Scheduling Problem for which we show an exact polynomial algorithm is presented. The restriction of the generally formulated problem, which is NP-hard, is the assumption that the processing time of each task depends only on the type of the processor which executes it and can be equal to one of two values: 1 (for fast processors) and L (for slower ones), where $L \in N$.

УПОРЯДОЧЕНИЕ ЗАДАЧ В ОТКРЫТОЙ СИСТЕМЕ С ДВУМЯ ТИПАМИ ПРОЦЕССОРОВ

Резюме: В работе рассмотрен особый случай открытрой системы обслуживания, для которой представлен оптимальный алгоритм. Ограничение по отношению к более общей проблеме "Open Shop", которая является NP-трудной, состоит в принятии предположения, что время выполнения отдельной операции зависит только от процессора, на котором она выпоняется, и может принимать одно из двух значений: 1 (на быстрых процессорах) или L (на более медленных), где $L \in N$.

1. Wstęp

Model otwartego systemu obsługi zajmuje istotną pozycję w problematyce szeregowania zadań. Notacja i specyfikacja problemu "Open Shop" rozważanego w poniższej pracy zostały zaczerpnięte z pozycji [1]. W ogólnym przypadku już dla trzech procesorów problem znajdowania optymalnego uszeregowania jest NP-trudny [2]. Tym niemniej dla pewnych szczególnych przypadków znane są dokładne algorytmy wielomianowe. Obszerny przegląd takich przypadków przedstawiono ostatnio w pracy [3]. Przedstawione dalej rozważania dotyczą przypadku, w którym czas wykonywania operacji zależy jedynie od typu procesora, na którym jest ona wykonywana, przy czym zakłada się, że w systemie występują jedynie dwa typy procesorów. W pierwszej części przedstawiamy dolne i górne oszacowania długości uszeregowania. Prezentowane tam konstrukcje optymalnych uszeregowań są następnie bezpośrednio wykorzystane do przedstawienia wielomianowego, dokładnego algorytmu rozwiązującego postawiony problem. Na zakończenie omawiamy ewentualne możliwości uogólnienia modelu, dla którego zaprezentowany algorytm może mieć zastosowanie.

2. Definicja problemu

Niech odpowiednio:

$M^s = \{M_0^s, M_1^s, \ldots, M_{k-1}^s\}$ - oznacza zbiór szybkich procesorów,

$M^w = \{M_0^w, M_1^w, \ldots, M_{r-1}^w\}$ - zbiór wolnych procesorów,

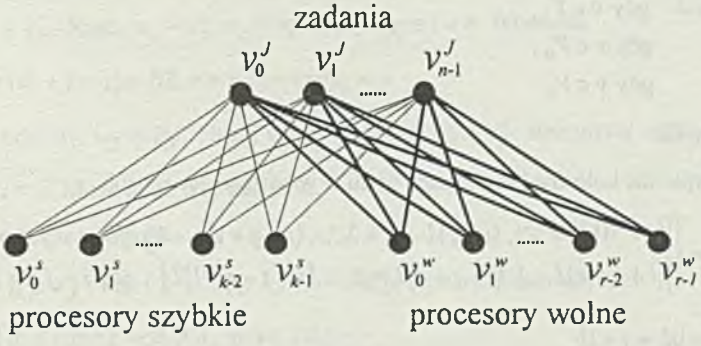
$J = \{J_0, J_1, \ldots, J_{n-1}\}$ - zbiór zadań,

gdzie $k, r, n \in \mathrm{N} \cup \{0\}$.

Każde zadanie J_i składa się z $k+r$ operacji: $o_{i,0}^s, o_{i,1}^s, \ldots, o_{i,k-1}^s, o_{i,0}^w, o_{i,1}^w, \ldots, o_{i,r-1}^w$, które muszą być wykonywane odpowiednio na procesorach $M_0^s, M_1^s, \ldots, M_{k-1}^s, M_0^w, M_1^w, \ldots, M_{r-1}^w$. Czas wykonania operacji $o_{i,j}^s$ ($o_{i,j}^w$) na procesorze szybkim (wolnym) wynosi 1 (L), gdzie L jest dowolną liczbą naturalną. Ponadto zakłada się, że żadne dwie operacje wchodzące w skład tego samego zadania nie mogą być wykonywane równocześnie oraz każdy procesor wykonuje

co najwyżej jedną operację w tym samym czasie. Rozwiązanie problemu polega na określeniu momentów rozpoczęcia wykonywania każdej operacji w taki sposób, aby moment zakończenia wykonywania ostatniej z nich był najwcześniejszy z możliwych. Operacje nie mogą być przerywane, a ich kolejność wykonywania jest dowolna. Tak określony model jest szczególnym (duoprocesorowym) przypadkiem klasycznego problemu otwartego systemu obsługi (Nonpreemptive Open-Shop Scheduling) NOSS.

Określony powyżej problem duoprocesorowego systemu otwartego można przedstawić w terminach teorii grafów jako przypadek przedziałowego kolorowania krawędzi grafu dwudzielnego $G = (V_J, V_S, V_W, E)$, gdzie każdemu procesorowi M_i^s (M_i^w) odpowiada wierzchołek $v_i^s \in V_S$ $(v_i^w \in V_W)$, każdemu zadaniu J_i odpowiada wierzchołek $v_i^J \in V_J$, a każdej operacji $o_{i,j}^s$ $(o_{i,j}^w)$ odpowiada krawędź $e_{i,j}^s \in E$ $(e_{i,j}^w \in E)$. V_J oraz $V_S \cup V_W$ stanowią podział wierzchołków grafu G na dwa zbiory niezależne. *k-pokolorowanie krawędziowe* grafu G polega na przypisaniu każdej krawędzi $e_{i,j}^s$ $(e_{i,j}^w)$ koloru (interwału L kolorów) ze zbioru $\{1, 2,...,k\}$, tak aby żadne dwie sąsiednie krawędzie nie otrzymały tego samego koloru. W tym przypadku optymalne rozwiązanie polega na znalezieniu k-pokolorowania krawędziowego, gdzie $k = \chi'(G)$ jest równe indeksowi chromatycznemu grafu G.



Rys. 1. Graf ilustrujący przypadek szeregowania w otwartym systemie duoprocesorowym

Fig. 1. Graph of open shop scheduling in a duoprocessor system

Krawędzie typu $e_{i,j}^s$ $(e_{i,j}^w)$ zostały zaznaczone cienką (grubą) linią.

3. Dolne i górne oszacowanie indeksu chromatycznego

Przyjmijmy następujące oznaczenia:

$\rho_1(v) = \left|\{(v,v') \in E \mid v \in V_S \vee v' \in V_S\}\right|$ (stopień wierzchołka v w podgrafie indukowanym przez zbiór wierzchołków $V_J \cup V_S$),

$\rho_L(v) = \left|\{(v,v') \in E \mid v \in V_W \vee v' \in V_W\}\right|$ (stopień wierzchołka v w podgrafie indukowanym przez zbiór wierzchołków $V_J \cup V_W$),

$\delta(v) = \rho_1(v) + L \cdot \rho_L(v)$ (stopień ważony wierzchołka v),

$\Delta(G) = \max\limits_{v \in V_J \cup V_S \cup V_W} \delta(v)$ (maksymalny stopień ważony w grafie G).

Lemat 1. $\chi'(G) \geq \Delta(G)$.

Dowód: Nierówność wynika bezpośrednio z definicji indeksu chromatycznego i przedziałowego pokolorowania krawędziowego.

❑

Lemat 2. Jeżeli $r = n$, to $\chi'(G) \leq \Delta(G)$.

Dowód:

$$\delta(v) = \begin{cases} r \cdot L + k & \text{gdy } v \in V_J, \\ r \cdot L & \text{gdy } v \in V_W, \\ r & \text{gdy } v \in V_S. \end{cases}$$

Stąd $\Delta(G) = rL+k$.

Określmy przypisanie kolorów krawędziom grafu G w następujący sposób:

$$(*)\quad kol\left(e^w_{i,j}\right) = \begin{cases} \left[(i+j)L+k+1, (i+j)L+k+2, \ldots, (i+j+1)L+k\right] & gdy\ i+j<r \\ \left[(i+j-r)L+1, (i+j-r)L+2, \ldots, (i+j-r+1)L\right] & gdy\ i+j \geq r \end{cases}$$

$$(**)\quad kol\left(e^s_{i,j}\right) = \left[iL+j+1\right],$$

gdzie $kol(e)$ oznacza kolor lub interwał kolorów przydzielony krawędzi e. Pokażemy, że przypisanie to daje $\Delta(G)$-pokolorowanie krawędziowe grafu G, tzn. dowolne dwie krawędzie (v, w_1), (v, w_2) incydentne do dowolnego wierzchołka v nie otrzymują tego samego koloru oraz maksymalny użyty kolor jest nie większy niż $\Delta(G)$. W tym celu rozpatrzmy kolejno wszystkie możliwe przypadki:

1. $v \in V_J$. Niech $v = v_i^J$. Możliwe są trzy przypadki:

1.1. $w_1, w_2 \in V_W$. Niech $w_1 = v_j^W$, $w_2 = v_h^W$. Bez utraty ogólności możemy założyć, że $j < h$. Możliwe są trzy przypadki:

1.1.1. $i + j \geq r$. Wówczas: $kol(v, w_1) = [(i + j - r)L + 1, \ldots, (i + j - r + 1)L]$,

$kol(v, w_2) = [(i + h - r)L + 1, \ldots, (i + h - r + 1)L]$. Wystarczy zauważyć, że:

$(i + j - r + 1)L < (i + h - r)L + 1$, gdyż $j + 1 \leq h$.

1.1.2. $i + h \geq r$ oraz $i + j < r$. Wówczas: $kol(v, w_1) = [(i + j)L + k + 1, \ldots, (i + j + 1)L + k]$,

$kol(v, w_2) = [(i + h - r)L + 1, \ldots, (i + h - r + 1)L]$. Wystarczy zauważyć, że:

$(i + h - r + 1)L < (i + j)L + k + 1$, gdyż $h - j \leq r - 1$.

1.1.3. $i + h < r$. Wówczas rozumowanie jest podobne jak w przypadku 1.1.1.

1.2. $w_1 \in V_W$ oraz $w_2 \in V_S$. Niech $w_1 = v_j^W$, $w_2 = v_h^S$. Możliwe są dwa przypadki:

1.2.1. $i + j \geq r$. Wówczas: $kol(v, w_1) = [(i + j - r)L + 1, \ldots, (i + j - r + 1)L]$,

$kol(v, w_2) = [iL + h + 1]$. Wystarczy zauważyć, że: $(i + j - r + 1)L < iL + h + 1$, gdyż $j \leq r - 1$.

1.2.2. $i + j < r$. Wówczas: $kol(v, w_1) = [(i + j)L + k + 1, \ldots, (i + j + 1)L + k]$,

$kol(v, w_2) = [iL + h + 1]$. Wystarczy zauważyć, że: $(i + j)L + k + 1 > iL + h + 1$, gdyż $h < k$.

1.3. $w_1, w_2 \in V_S$. Niech $w_1 = v_j^S$, $w_2 = v_h^S$. Oczywiście $j \neq h$. Wówczas:

$kol(v, w_1) = [iL + j + 1] \neq [iL + h + 1] = kol(v, w_2)$.

2. $v \in V_W$ oraz $w_1, w_2 \in V_J$. Niech $v = v_j^W, w_1 = v_i^J, w_2 = v_p^J$. Bez utraty ogólności możemy założyć, że $i < p$. Możliwe są trzy przypadki:

2.1. $i + j \geq r$. Wówczas $kol(v, w_1) = [(i + j - r)L + 1, \ldots, (i + j - r + 1)L]$,

$kol(v, w_2) = [(p + j - r)L + 1, \ldots, (p + j - r + 1)L]$. Wystarczy zauważyć, że:

$(i + j - r + 1)L < (p + j - r)L + 1$, gdyż $i \leq p - 1$.

2.2. $p + j \geq r$ oraz $i + j < r$. Wówczas: $kol(v, w_1) = [(i + j)L + k + 1, \ldots, (i + j + 1)L + k]$,

$kol(v, w_2) = [(p + j - r)L + 1, \ldots, (p + j - r + 1)L]$. Wystarczy zauważyć, że:

$(p + j - r + 1)L < (i + j)L + k + 1$, gdyż $p - i \leq r - 1$.

2.3. $p + j < r$. Wówczas rozumowanie jest podobne jak w przypadku 2.1.

3. $v \in V_S$ oraz $w_1, w_2 \in V_J$. Niech $v = v_j^S, w_1 = v_i^J, w_2 = v_p^J$. Oczywiście $i \neq p$. Wówczas:

$kol(v, w_1) = [iL + j + 1] \neq [pL + j + 1] = kol(v, w_2)$.

Łatwo zauważyć, że w każdym przypadku przydzielone są kolory z przedziału $\{1, 2, \ldots, \Delta(G)\}$, gdyż:

$(i + j)L + k + 1 \geq 1$ oraz $i + j < r \Rightarrow (i + j + 1)L + k \leq rL + k = \Delta(G)$,

$i + j \geq r \Rightarrow (i + j - r)L + 1 \geq 1$ oraz $(i + j - r + 1)L \leq (r - 1)L < \Delta(G)$ oraz

$1 \leq iL + j + 1 \leq (r - 1)L + k < \Delta(G)$.

❑

Lemat 3. Jeżeli $r > n$, to $\chi'(G) \leq \Delta(G)$.

Dowód:

$$\delta(v) = \begin{cases} rL + k & \text{gdy } v \in V_J, \\ nL & \text{gdy } v \in V_W, \\ n & \text{gdy } v \in V_S. \end{cases}$$

Stąd, wobec $r > n$, $\Delta(G) = rL + k$. Legalne pokolorowanie w tym przypadku otrzymujemy przez dołączenie do grafu G dodatkowych $r - n$ wierzchołków (wraz z odpowiednimi, wychodzącymi z nich, krawędziami) odpowiadających dodatkowym zadaniom, sprowadzając problem do przypadku objętego przez lemat 2. Łatwo zauważyć, że dodanie owych dodatkowych zadań w tym przypadku nie zmienia wartości Δ, a zatem na mocy lematu 2 istnieje $\Delta(G)$-pokolorowanie krawędziowe nadgrafu grafu G, i tym samym grafu G.

❑

Lemat 4. Jeżeli $r < n$ oraz $k \leq (n - r)L$, to $\chi'(G) \leq \Delta(G)$.

Dowód: $\Delta(G) = max\{rL + k, nL, n\} = nL$, ponieważ $k \leq (n - r)L$. W tym przypadku zbiór k szybkich procesorów może być podzielony na $n - r$ podzbiorów o mocy nie większej niż L: $V_S = V_0' \cup V_1' \cup \ldots \cup V_{n-r-1}'$, gdzie $\underset{i,j}{\forall} i \neq j \Rightarrow V_i' \cap V_j' = \varnothing$ oraz $\underset{i}{\forall} |V_i'| \leq L$. Traktując chwilowo każdy ze zbiorów V_i' jako "zastępczy" wolny procesor, otrzymujemy przypadek objęty lematem 2 (n zadań, n wolnych procesorów, 0 szybkich procesorów). Możliwe jest zatem pokolorowanie takiego grafu $G' = \left(V_J, \varnothing, V_W \cup \{V_0', \ldots, V_{n-r-1}'\}, E'\right)$ przy użyciu nie więcej niż $\Delta(G') = nL$ kolorów. Legalne pokolorowanie grafu G otrzymujemy przez dowolne rozbicie każdego z interwałów długości L przydzielonych w G' krawędziom typu $\left(v_i^J, V_j'\right)$ na co

najwyżej krawędzi L $\left(v_i^J, v_h^S\right)$ w grafie G, gdzie $v_h^S \in V_j'$ i $0 \le h < L$. W przypadku gdy $\left|V_j'\right| < h$, niektóre kolory z interwału przydzielonego $\left(v_i^J, V_j'\right)$ pozostaną niewykorzystane.

❑

Lemat 5. Jeżeli $r < n$ oraz $k > (n - r)L$, to $\chi'(G) \le \Delta(G)$.

Dowód: $\Delta(G) = max\{rL + k, nL, n\} = rL + k$, ponieważ $k > (n - r)L$. Konstrukcja $\Delta(G)$-pokolorowania krawędziowego przebiega analogicznie jak w dowodzie lematu 4, z tą różnicą że w konstrukcji grafu G' spełniającego założenia lematu 2 jedynie $(n - r)L$ szybkich procesorów dzielimy na $n - r$ grup po L procesorów traktując chwilowo każdą z nich jako procesor wolny.

❑

4. Algorytm

Bezpośrednim wnioskiem z lematów 1, 2,..., 5 jest:

Twierdzenie 1. $\chi'(G) = \Delta(G)$ dla każdej instancji problemu duoprocesorowego systemu otwartego.

❑

Konstruktywne dowody lematów 2, 3, 4, 5 bezpośrednio kształtują dokładny algorytm szeregowania, którego sterowanie wygląda następująco:

procedure *DuoSzereg*(n, r, k);
{ n, r, k oznaczają odpowiednio liczbę zadań, liczbę wolnych oraz szybkich procesorów }
begin
 if $n = r$ **then** przydziel czas operacjom zgodnie z formułami (*) oraz (**) (lemat 2)
 else
  if $n < r$ **then** przydziel czas operacjom zgodnie z formułami (*) oraz (**) (lemat 3)
  else
   if $k \le (n - r)L$ **then** przydziel czas operacjom zgodnie z konstrukcją opisaną w
                dowodzie lematu 4
   else przydziel czas operacjom zgodnie z konstrukcją opisaną w dowodzie lematu 5
end

Złożoność obliczeniowa powyższego algorytmu wynosi O(nm), gdzie $m = r + k$ jest liczbą wszystkich procesorów. Wynika to z faktu, że każdej z nm operacji musi być przydzielony odpowiedni interwał czasu, którego wartość może być wyliczona w czasie O(1). Konstrukcje zastępczego grafu G', w przypadku gdy $n > k$, mogą być oczywiście zrealizowane w czasie O(nm). Z drugiej strony, ponieważ liczba kroków musi być co najmniej proporcjonalna do nm, więc jest to algorytm optymalny w sensie złożoności obliczeniowej.

5. Podsumowanie

Opisany powyżej przypadek otwartego systemu obsługi jest na tyle szczególny, że trudno mieć nadzieję na bezpośrednie praktyczne wykorzystanie przedstawionego algorytmu. Może on jednak służyć do stworzenia heurystyki pomocnej w zaprojektowaniu suboptymalnych algorytmów uwzględniających bardziej ogólny przypadek problemu "Open Shop".

Z drugiej strony należy zauważyć, że przedstawiony algorytm może być wykorzystany do szerszej klasy przypadków, niż to wynika z określonej w drugim paragrafie definicji problemu. Np. założenie o tym, że każde zadanie składa się dokładnie z $r + k$ operacji, może być zastąpione bardziej ogólnym (dopuszczającym, że niektóre zadania mają mniej operacji), mianowicie że istnieje przynajmniej jedno zadanie o $r + k$ operacjach, o ile $r \geq n$ lub $k > (n - r)L$, oraz że istnieje przynajmniej jeden procesor wolny, na którym wykonywanych jest n operacji, o ile $r < n$ i $k \leq (n - r)L$. W takiej bowiem sytuacji wartość Δ pozostaje niezmieniona (uzasadnienie jak w dowodzie lematu 3). Również założenie o wykluczeniu możliwości przerywania wykonywania operacji może być opuszczone (nie wpływa ono na dolne oszacowanie z lematu 1, a zatem algorytm znajduje rozwiązanie optymalne).

Podziękowania

Autor pragnie podziękować Profesorowi Markowi Kubale za szereg cennych, krytycznych uwag, które w istotny sposób przyczyniły się do poprawy merytorycznej zawartości powyższego referatu oraz dr. Oldze Choreń za pomoc w redakcji językowej.

LITERATURA

[1] Błażewicz J., Cellary W., Słowiński R., Węglarz J.: *Badania Operacyjne dla Informatyków*. WNT, Warszawa 1983, pp. 161-227.

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

[3] de Werra D., Solot Ph.: Some Graph-Theoretical Models for Scheduling in Automated Production Systems. *Networks, vol.* 23, 1993, pp. 651-660.

Recenzent: Dr inż. Stanisław Zdrzałka

Wpłynęło do Redakcji do 30.04.1994 r.

Abstract

The Open-Shop Scheduling (OSS) problem in a general case is described as follows: There are given two collections: a collection of processors $M = \{M_1, M_2,..., M_m\}$ and a collection of jobs $J = \{J_1, J_2,..., J_n\}$. Each job J_i consists of m tasks $o_{i,1}, o_{i,2},...,o_{i,m}$, where $o_{i,j}$ has to be processed on the processor M_j. The processing time of each task is known. A processor cannot work on pair of tasks at a time and no two tasks of the same job can be processed simultaneously. Furthermore, in nonpreemptive OSS no interruption is allowed during processing of a task. A solution of this problem is the schedule with a minimum completion time.

In general case this problem is NP-hard. In this paper we present a polynomial $O(mn)$ exact algorithm for particular case of the nonpreemptive OSS, where the processing time of each task depends only on the type of the processor which executes it and is equal to one of two values: 1 (for fast processors) and L (for slower ones), where $L \in N$. An example for this model can be a concurrent execution of similar programs on a computer with two types of processors (e.g. numerical and communicative). We also present easy formulas which allow to find immediately the minimum completion time. Finally, we discuss the possibility to generalize our model so that presented algorithm can still be useful.