ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2002
Seria: AUTOMATYKA z. 134 Nr kol. 1554

Krzysztof GIARO, Marek KUBALE
Politechnika Gdańska

PROGRAMOWANIE DYNAMICZNE W ROZWIĄZYWANIU PROBLEMÓW SZEREGOWANIA ZADAŃ W SYSTEMACH O ACYKLICZNEJ STRUKTURZE

Streszczenie. Rozważono rozrzedzone systemy niepodzielnych zadań dwuprocesorowych o jednostkowych długościach operacji oraz systemy maszyn dedykowanych (open shop, flow shop, mixed shop) o operacjach zero–jedynkowych. Przedstawiono rodzinę wielomianowych algorytmów opartych na programowaniu dynamicznym, pozwalających na znalezienie optymalnego uszeregowania względem szerokiej rodziny funkcji kryterialnych. Stopień rozrzedzenia systemu zdefiniowano posługując się jego modelem grafowym – w zakresie naszego zainteresowania leżą jedynie takie instancje problemów szeregowania, których modelami są grafy acykliczne.

DYNAMIC PROGRAMMING TO SOLVE SCHEDULING PROBLEMS IN SYSTEMS WITH ACYCLIC STRUCTURE

Summary. We consider systems with nonpreemptable 2-processor tasks of unit execution times as well as systems of dedicated processors with zero-unit execution times. We are interested in systems whose scheduling graphs are acyclic. For such graphs we give a family of polynomial algorithms based on dynamic programming which yield optimal solutions with respect to a broad family of criterion functions.

1. Wprowadzenie

Rozważmy zbiór niepodzielnych zadań o jednostkowych czasach wykonania, z których każde wykorzystuje dokładnie dwa przypisane mu procesory. Żadna maszyna nie może też pracować nad więcej niż jednym zadaniem równocześnie. Tego typu model nazywany jest w literaturze *systemem zadań dwuprocesorowych* i znajduje zastosowania wszędzie tam, gdzie poszczególne zadania wymagają współpracy dokładnie dwóch podmiotów, a więc np. przy przesyłaniu plików ze stacji źródłowych do docelowych lub przy

szeregowaniu testów komputerowych [6]. Pragniemy znaleźć wielomianowy algorytm optymalnie szeregujący takie zadania przy możliwie jak najszerszej klasie funkcji kryterialnych. Niestety, znane dotąd wyniki wskazują na NP-trudność zagadnienia przy bardzo prostych klasycznych kryteriach, np. powszechnie wiadomo, że obliczeniowo trudne są już problemy $P|fix_j=2,UET|C_{\max}$ i $P|fix_j=2,UET|\Sigma C_j$. Skoro szukamy efektywnych i dokładnych algorytmów – konieczne będzie narzucenie dalszych ograniczeń na strukturę rozważanego systemu zadań.

W dalszej analizie pomocny okaże się model grafowy zbioru zadań dwuprocesorowych. Możemy bowiem przyjąć, że procesory stanowią wierzchołki grafu, którego krawędziami są szeregowane zadania – każde z nich łączy ze sobą oba wykorzystywane procesory. Takie spojrzenie pozwala zdefiniować systemy *acykliczne* jako te, których grafy nie zawierają cykli, a więc są drzewami lub lasami. W dalszej części będziemy się zajmować jedynie układami zadań o strukturze acyklicznej. Wprowadzimy też do notacji trójpolowej oznaczenie $M=acyclic$ dla wyrażenia zdefiniowanej wyżej własności.

Samo założenie acykliczność nie wystarcza jednak, by problemy szeregowania dawały się rozwiązać w czasie wielomianowym. Warto przypomnieć:

Fakt 1.1. *Udowodniono, że:*

[4] *Problem* $P|fix_j=2,M=acyclic|C_{max}$ *jest silnie NP–trudny.*

[2] *Problem* $P|fix_j=2,M=acyclic|\Sigma C_{ij}$ *jest silnie NP–trudny.*

Jeżeli funkcje kryterialne o elementarnym charakterze prowadzą do NP-trudności, konieczne będzie dalsze zawężenie rozważanego modelu. Będziemy więc zakładać, że wszystkie zadania mają jednakowe (np. jednostkowe) czasy wykonania (symbolicznie *UET* od *Unit Execution Time*). Wówczas harmonogram może zostać przedstawiony jako przypisanie zadaniom jednostek czasu ponumerowanych kolejno liczbami naturalnymi, a wtedy zbiór poprawnych harmonogramów odpowiada legalnym pokolorowaniom krawędziowym grafu systemu.

Jak wspomnieliśmy, nasze grafy systemu są drzewami lub lasami. W teorii grafów drzewa ze względu na swą rekurencyjną strukturę okazują dużą podatność na obróbkę algorytmami opartymi na programowaniu dynamicznym. Wykażemy, że w przypadku zbioru jednostkowych zadań dwuprocesorowych o takiej strukturze możliwe jest skonstruowanie wielomianowych algorytmów minimalizujących maksymalny koszt uszeregowania zadania $P|fix_j=2,M=acyclic,UET|f_{\max}$ oraz łączny koszt po wszystkich zadaniach $P|fix_j=2,M=acyclic,UET|\Sigma f_j$, gdzie o funkcji kosztu $f(e,i)$ uszeregowania zadania e w chwili i

zakładamy jedynie, że jest **niemalejąca względem i oraz obliczalna w czasie $O(1)$**. Łatwo spostrzec, że takie funkcje kryterialne uogólniają znane klasyczne $C_{\max}$, $L_{\max}$, $T_{\max}$, $\Sigma w_j C_j$, $\Sigma w_j T_j$, $\Sigma w_j U_j$ i inne.

Podobnie rozważymy szeregowanie zadań na maszynach dedykowanych w systemach: otwartym, przepływowym, a nawet w uogólniającym je *systemie mieszanym* (*mixed shop*), kiedy to część zadań może wykonywać się w dowolnym porządku operacji, dla pozostałych zaś kolejność ich wykonania musi być zgodna z numeracją maszyn. Stosowny model grafowy uzyskamy biorąc procesory jako wierzchołki trafiające do jednej partycji, a zadania do drugiej oraz łącząc krawędzią zadanie z procesorem wtedy i tylko wtedy, gdy zawiera ono odpowiednią operację niezerową (a zatem graf systemu jest zawsze dwudzielny, a krawędzie odpowiadają w sposób wzajemnie jednoznaczny operacjom o niezerowej długości). Znów przez *acykliczny system mieszany* będziemy rozumieć taki system, którego graf nie zawiera cykli, czyli jest drzewem lub lasem. Niestety nadal mamy:

Fakt 1.2. *Udowodniono, że:*

[4] *Problem* $O|M=acyclic|C_{max}$ *jest silnie NP-trudny.*

[2] *Problem* $O|M=acyclic\ |\Sigma C_{ij}$ *jest silnie NP-trudny.*

[3] *Problem* $O2|M=acyclic\ |\Sigma C_{ij}$ *jest NP-trudny.*

Dlatego dalej ograniczymy się do acyklicznych systemów mieszanych o operacjach zero-jedynkowych (oznaczenie *ZUET*). Podobnie jak poprzednio możemy interpretować harmonogramy jak legalne pokolorowania krawędziowe grafu. Należy jednak zauważyć, że tym razem nie każde pokolorowanie powstałego grafu dwudzielnego odpowiada jakiemuś harmonogramowi – jedynie takie, które respektuje odpowiednie warunki kolejnościowe narzucane na kolory krawędzi przy wierzchołkach-zadaniach wykonujących się w reżimie flow-shopu (wzajemna jednoznaczność pomiędzy zbiorem harmonogramów i legalnych pokolorowań obowiązuje jedynie dla systemu otwartego). Teraz koszt uszeregowania zadania v, tak by się kończyło ono w chwili i, oznaczymy przez $f(v,i)$, która to funkcja ma spełniać założenia analogiczne jak w modelu zadań dwuprocesorowych. Wykażemy, że acykliczność systemu mieszanego typu *ZUET* pozwala na konstrukcję wielomianowych algorytmów rozwiązujących zarówno problem minimalizacji maksymalnego kosztu $X|M=acyclic,ZUET|f_{\max}$, jak i łącznego kosztu uszeregowania wszystkich zadań $X|M=acyclic,ZUET|\Sigma f_j$, które znów uogólniają wiele klasycznych funkcji kryterialnych.

Poczyńmy teraz pewne założenia odnośnie do używanych oznaczeń. Tradycyjnie $G(V,E)$ symbolizuje graf o zbiorze wierzchołków V i krawędzi E. Dla grafów dwudzielnych

posługiwać się będziemy też zapisem $G(V_1,V_2,E)$ czyniącym bardziej widocznym podział zbioru wierzchołków na dwie partycje. Standardowo n,m,Δ to odpowiednio liczba wierzchołków i krawędzi oraz maksymalny stopień w omawianym aktualnie grafie, liczba $\deg(v)$ to stopień wierzchołka v, zaś N oznacza zbiór liczb naturalnych, wreszcie $N_0=N\cup\{0\}$. Podgraf indukowany grafu $G(V,E)$ przez zbiór wierzchołków $V'\subseteq V$ oznaczamy jako $G[V']$. W przypadku modelowania systemu z procesorami dedykowanymi za pomocą grafu dwudzielnego $G(V_1,V_2,E)$ będziemy przyjmować, że pierwsza partycja stanowi zbiór maszyn, a druga – zadań. Do pełnego opisu mixed-shopu konieczne będzie ponadto podanie różnowartościowej funkcji *num*: $V_1\rightarrow N$ numerującej procesory, a także rozbicia zadań na dwa rozłączne podzbiory $V_2=V_O\cup V_F$, odpowiadające zadaniom o dowolnej kolejności wykonywania operacji V_O i tych z V_F, dla których musi być ona zgodna z funkcją *num*. W ten sposób system otwarty odpowiada sytuacji, gdy $V_F=\emptyset$, zaś przy $V_O=\emptyset$ mamy do czynienia z systemem przepływowym.

Definicja 1.1. Niech $G(V,E)$ będzie dowolnym grafem. Funkcję c: $E\rightarrow N$ nazywamy *pokolorowaniem krawędziowym* wtedy i tylko wtedy, gdy dla każdego wierzchołka v grafu G krawędzie incydentne do v uzyskują parami różne kolory.

W dalszej części będziemy omawiać metody rozwiązywania pewnych zagadnień optymalizacyjnych (np. „znajdź optymalne uszeregowanie", „znajdź największe skojarzenie w grafie"), choć prezentowane algorytmy będą jedynie odpowiadały na pytania o optymalną wartość kryterium („jaka jest minimalna wartość kryterium dla uszeregowania", „jaki jest rozmiar największego skojarzenia w grafie"). Czynimy tak dla zachowania klarowności opisu algorytmów – po ich analizie czytelnik bez trudu rozszerzy przedstawione procedury do pełnych wersji problemów optymalizacyjnych.

2. Wyszukiwanie skojarzeń w grafach

Definicja 2.1. W grafie $G(V,E)$ *skojarzeniem* M nazywamy dowolny podzbiór zbioru krawędzi, taki że żadne dwie krawędzie z M nie stykają się ze sobą.

Definicja 2.2. Jeżeli krawędzie w $G(V,E)$ są obciążone nieujemnymi wagami opisanymi funkcją w:$E\rightarrow N_0$, to *wagą skojarzenia* M jest liczba $\Sigma_{e\in M}\, w(e)$.

Definicja 2.3. Skojarzenie M w grafie $G(V,E)$ *nasyca* zbiór wierzchołków $U \subseteq V$ wtedy i tylko wtedy, gdy każdy wierzchołek z U jest incydentny z pewną krawędzią z M.

Przypomnijmy, że:

Fakt 2.1 [7]. *Największe skojarzenie (tj. o największej mocy) w danym grafie G można znaleźć w czasie* $O(mn^{1/2})$.

Standardowo problem najcięższego skojarzenia (tj. o największej wadze) rozwiązywano w czasie $O(n^3)$, istnieje jednak lepszy algorytm o złożoności $O(mn^{1/2}\log(Wn))$, gdzie W szacuje z góry wagi krawędzi. Wiadomo nawet więcej, mianowicie:

Fakt 2.2 [5]. *Istnieje algorytm, który dla danego grafu $G(V,E)$ obciążonego nieujemnymi wagami znajduje najcięższe skojarzenia we wszystkich podgrafach $G-v$ po $v \in V$, działający w czasie* $O(mn^{1/2}\log(Wn))$.

Oczywiście, możemy wykorzystać powyższy algorytm do znajdowania najlżejszego skojarzenia spośród wszystkich najliczniejszych – sprowadza się to do zmiany znaku wszystkich wag krawędzi oraz dodania do nich dostatecznie dużej stałej (np. $\lfloor Wn/2 \rfloor+1$).

Pomocne okażą się też skojarzenia z pewnymi dodatkowymi ograniczeniami.

Definicja 2.4. Niech dany będzie graf dwudzielny $G(V_1,V_2,E)$ i funkcja różnowartościowa num: $V_1 \cup V_2 \rightarrow N$ numerująca jego wierzchołki (będziemy mówili o *grafie z ponumerowanymi wierzchołkami*). Skojarzenie M w G nazwiemy *planarnym* wtedy i tylko wtedy, gdy dla dowolnych krawędzi ux i vy należących do M, gdzie $\{u,v\} \subseteq V_1$, zachodzi implikacja $num(u)>num(v) \Rightarrow num(x)>num(y)$.

Nazwa bierze się stąd, że jeśli potraktujemy wierzchołki z V_1 i V_2 jako punkty na dwóch równoległych i jednakowo skierowanych osiach liczbowych (wierzchołki na tych osiach będą leżały w kolejności zgodnej z ich numeracją), a następnie nakreślimy krawędzie z E jako prostoliniowe odcinki łączące odpowiednie wierzchołki – wówczas warunek planarności skojarzenia M jest równoważny temu, że odcinki-krawędzie z M nie przecinają się. Wiele własności skojarzeń planarnych w grafach można rozstrzygać w czasie wielomianowym, podobnie jak dla zwykłych skojarzeń.

Twierdzenie 2.3. *Niech dany będzie graf dwudzielny $G(V_1,V_2,E)$ o ponumerowanych wierzchołkach. Istnienie skojarzenia planarnego w G, nasycającego całą partycję V_2, można zweryfikować w czasie* $O(n+m)$.

Dowód. Przyjmijmy, że $V_1=\{v_1,..., v_{n_1}\}$ i $V_2=\{u_1,..., u_{n_2}\}$, przy czym oba ciągi wierzchołków wypisano w kolejności rosnących numerów. Dla liczby naturalnej $i \le n_2$ niech A_i będzie najmniejszym takim j, że w $G[\{v_1,..., v_j,u_1,..., u_i\}]$ istnieje skojarzenie planarne nasycające wierzchołki $\{u_1,..., u_i\}$. Łatwo zauważyć, że poprawna jest następująca formuła rekurencyjna:

$A_1 = \min \{x : u_1v_x \in E\}$,

$A_{i+1} = \min \{x > A_i : u_{i+1}v_x \in E\}$ dla $i>0$.

Odpowiedź na postawione pytanie jest TAK wtedy i tylko wtedy, gdy formuły powyższe pozwolą na obliczenie A_{n_2}, czyli gdy wszystkie zbiory pod operacją min() będą niepuste.

Twierdzenie 2.4. *Niech dany będzie pełny graf dwudzielny $G(V_1,V_2,E)$ (gdzie $|V_1| \ge |V_2|$) o ponumerowanych wierzchołkach i krawędziach obciążonych wagami w: $E \to N_0$. Wagę najlżejszego skojarzenia planarnego nasycającego V_2 można znaleźć w czasie $O(|V_1||V_2|)$.*

Dowód. Przyjmijmy oznaczenia wierzchołków z poprzedniego dowodu i określmy ciąg tablic o wartościach całkowitych $T_j[i]$ dla $j=1,...,n_2$, $i=j,...,n_1$. Wartością $T_j[i]$ ma być minimalna waga skojarzenia planarnego nasycającego drugą partycję grafu $G[\{v_1,..., v_i,u_1,..., u_j\}]$. Łatwo sprawdzić poprawność następujących rekursji:

$T_1[1]=w(u_1v_1)$,

$T_1[i+1]=\min \{T_1[i],w(u_1v_{i+1})\}$ dla $i>0$.

oraz

$T_{j+1}[j+1]=w(u_{j+1}v_{j+1})+T_j[j]$ dla $j>0$.

$T_{j+1}[i+1]=\min \{T_{j+1}[i],w(u_{j+1}v_{i+1})+T_j[i]\}$ dla $i>j>0$.

Można zauważyć, że wykonując powyższy algorytm równocześnie rozwiązujemy postawione w treści twierdzenia zagadnienie dla wszystkich podgrafów G postaci $G[\{v_1,...,v_i,u_1,..., u_j\}]$ przy $i \ge j$.

Twierdzenie 2.5. *Niech dany będzie graf dwudzielny $G(V_1,V_2,E)$ o ponumerowanych wierzchołkach i krawędziach obciążonych wagami w: $E \to N_0$. Wagę najcięższego skojarzenia planarnego można znaleźć w czasie $O(|V_1||V_2|)$.*

Dowód. Raz jeszcze powróćmy do poprzednich oznaczeń. Graf G możemy potraktować jak graf pełny dwudzielny (nowe krawędzie uzyskują wagi zerowe). Określmy ciąg tablic o wartościach całkowitych $T_j[i]$ dla $j=1,...,n_2$, $i=1,...,n_1$, przy czym wartością $T_j[i]$ jest maksymalna waga skojarzenia planarnego w grafie $G[\{v_1,...,v_i,u_1,..., u_j\}]$. Tym razem obowiązują równania:

$T_1[1]=w(u_1v_1)$,

$T_1[i+1]=\max\{T_1[i],w(u_1v_{i+1})\}$ dla $i>0$.

oraz

$T_{j+1}[1]=\max\{T_j[1],w(u_{j+1}v_1)\}$ dla $j>0$.

$T_{j+1}[i+1]=\max\{T_{j+1}[i], T_j[i+1],w(u_{j+1}v_{i+1})+T_j[i]\}$ dla dodatnich i,j.

Wniosek 2.6. *Niech dany będzie graf dwudzielny $G(V_1,V_2,E)$ o ponumerowanych wierzchołkach. Moc największego skojarzenia planarnego w G można znaleźć w czasie $O(|V_1||V_2|)$.*

Dowód. Problem sprowadza się do poprzedniego po nadaniu krawędziom wagi 1.

3. Wielomianowe algorytmy szeregujące

Wszystkie opisane tu algorytmy opierają się na modelowaniu uszeregowania za pomocą kolorowania krawędzi grafu przedstawiającego system. Pracują według następującego schematu. Możemy założyć, że model grafowy systemu jest drzewem, jeżeli bowiem stanowi on las, wówczas krawędzie z różnych składowych to zadania-operacje całkowicie niezależne i można szeregować każde drzewo osobno. Przedstawimy sposób na znalezienie wartości funkcji kryterialnej dla optymalnego uszeregowania systemu o grafie drzewa G. Znajdujemy liczbę L, dla której wiadomo, że istnieje uszeregowanie optymalne o długości nie przekraczającej L (sposób jej wyznaczania zależy od konkretnego modelu). Następnie dla dowolnej pary (v,H), gdzie H jest poddrzewem w G, a v wierzchołkiem stopnia 1 w H, przez TAB(v,H) oznaczamy tablicę L liczb całkowitych, taką że TAB$(v,H)[i]$ dla $i\in\{1,...,L\}$ jest równe minimalnej możliwej wartości kryterium dla uszeregowania podsystemu opisanego za pomocą grafu H w przedziale $1,...,L$, tak by krawędź przy v uzyskała kolor i (zakładamy przy tym, że podsystem H dziedziczy po G parametry, takie jak funkcję kosztów, numerację procesorów itd.). Celem naszym jest uzyskanie TAB(u,G), gdzie u jest dowolnym wierzchołkiem stopnia 1 w G – wówczas szukana optymalna wartość kryterium jest najmniejszym elementem tej tablicy. Algorytm jest rekurencyjny. W rozważanych modelach dla $H=K_2$ można wyliczyć TAB(v,H) w czasie $O(L)$. Niech teraz H będzie takim drzewem, że z wierzchołka w sąsiedniego do v w H wychodzi jeszcze $\deg(w)-1$ dodatkowych krawędzi (poza $\{v,w\}$), prowadzących do wierzchołków $v_1,...,v_{\deg(w)-1}$. Wtedy możemy rozbić resztę drzewa H pozostałą po usunięciu v na $\deg(w)-1$ krawędziowo

rozłącznych drzew $T_1,...,T_{\deg(w)-1}$, takich że w każdym z nich w jest wierzchołkiem stopnia 1. Załóżmy, że dysponując tabelami TAB(w,T_j) dla $j\in\{1,...,\deg(w)-1\}$ potrafimy w czasie wielomianowym wyliczyć TAB(v,H). Wówczas odpowiednio orientując drzewo G możemy obliczyć TAB(u,G) w czasie wielomianowym – wystarczy wyznaczyć co najwyżej $n-1$ tabel L-elementowych (po jednej dla każdej krawędzi drzewa).

3.1. Zagadnienie P|fixj=2,UET,M=acyclic|Σfj

Łatwo zauważyć, że istnieje optymalne uszeregowanie używające chwil czasu o numerach nie przekraczających $2\Delta(G)-1$, bowiem krawędź uv sąsiaduje z $\deg(u)+\deg(v)-2$ krawędziami i gdyby $c(uv)>\deg(u)+\deg(v)-1$, wówczas kolor ten można by zmniejszyć nie zwiększając też wartości kryterium dla całego pokolorowania. Przyjmujemy zatem $L=2\Delta(G)-1$. Warto wspomnieć, że jeśli funkcja kosztu $f(e,i)$ nie zależy od krawędzi e, wówczas możemy wziąć $L=\Delta(G)$ na mocy twierdzenia mówiącego, że każde optymalne kosztowe pokolorowanie krawędziowe grafu dwudzielnego (a takim jest drzewo) używa jedynie barw $1,..., \Delta(G)$ [9].

Przypadek $H=K_2$ ma trywialne rozwiązanie TAB(v,H)[i]=$f(vw,i)$ dla $i=1,...,L$. Zastanówmy się więc, jak obliczyć TAB(v,H)[i] na podstawie tabel TAB(w,T_j) dla $j\in\{1,...,\deg(w)-1\}$. Szukana wartość oznacza koszt poprzypisywania różnych kolorów ze zbioru $Kol=\{kol_1,...,kol_L\}$ do krawędzi wv_j oraz wv, przy czym wv ma zarezerwowaną barwę kol_i. Tworzymy zatem pomocniczy pełny graf dwudzielny K o pierwszej partycji w postaci zbioru barw Kol, a drugiej w postaci wierzchołków $\{v_1,...,v_{\deg(w)-1}\}$, przy czym krawędź kol_xv_y ma wagę TAB(v_y,T_y)[x]. Wówczas TAB(v,H)[i]=$f(vw,i)+M$, gdzie M jest wagą najlżejszego skojarzenia nasycającego drugą partycję w grafie $K-kol_i$. Operacja powyższa musi zostać powtórzona dla wszystkich i, co pozwala na zastosowanie dla grafu K procedury z faktu 2.2. Niech $F=\max_{e\in E,l<2\Delta} f(e,l)$ stanowi oszacowanie górne kosztu uszeregowania pojedynczej krawędzi. Wtedy złożoność procedury obliczenia wartości TAB(v,H) jest rzędu $O(L\deg(w)L^{0.5}\log(LnF))=O(\deg(w)\Delta^{1.5}\log(nF))$, a zatem czas pracy algorytmu dla całego grafu to $O(n\Delta^{1.5}\log(nF))$.

Warto wspomnieć, że możliwość wykorzystania algorytmu z faktu 2.2 w celu przyspieszenia procedury znajdującej optymalne kosztowe pokolorowanie krawędzi drzewa po raz pierwszy zauważono w [10].

3.2. Zagadnienie P|fix$_j$=2,UET,M=acyclic|f$_{max}$

Podobnie jak poprzednio pewne uszeregowanie optymalne nie używa barw większych od $L=2\Delta(G)-1$; przyjmujemy $F=\max_{e\in E, l<2\Delta} f(e,l)$. Dla $H=K_2$ znów zachodzi TAB(v,H)[i]=$f(vw,i)$. Rozważamy więc przypadek obliczania TAB(v,H)[i] przy znanych TAB(w,T_j). Oczywiście, wartość ta musi wynosić co najmniej $f(vw,i)$. Niech więc $f(vw,i)\leq C\leq F$ zastanówmy się, kiedy będzie możliwe pokolorowanie całego H tak, by żadna krawędź nie utworzyła kosztu większego od C? Najmniejsze takie C będzie wartością TAB(v,H)[i]. W tym celu przy danym C tworzymy graf pomocniczy K o zbiorach wierzchołków jak w poprzednim przypadku, tym razem jednak nie będzie to graf pełny dwudzielny, mianowicie $E(K)=\{kol_x v_y$: TAB(v_y,T_y)[x]$\leq C\}$. Odpowiedź na postawione wyżej pytanie dla wartości C jest TAK wtedy i tylko wtedy, gdy istnieje przypisanie odpowiednich barw do krawędzi wv_j, czyli gdy w $K-kol_i$ istnieje skojarzenie nasycające drugą partycję. W tym miejscu możemy się posłużyć algorytmem z faktu 2.1. Złożoność pojedynczego sprawdzenia to $O(\deg(w)L^{1.5})=O(\deg(w)\Delta^{1.5})$, zaś konieczność znajdowania optymalnego C metodą szukania binarnego daje czas wyliczenia całej TAB(v,H) w postaci $O(\deg(w)\Delta^{2.5}\log(F))$. Dla całego grafu złożoność algorytmu to $O(n\Delta^{2.5}\log(F))$.

3.3. Zagadnienie X|ZUET,M=acyclic|fmax

Czas zakończenia zadania jest równy maksymalnemu kolorowi krawędzi doń incydentnej, dlatego możemy wprowadzić koszt koloru na krawędzi (dla $u\in V_1$ i $v\in V_2$ przy $uv\in E$ mamy $f(uv,i)=f(v,i)$) i wyliczać funkcję kryterialną podobnie jak poprzednio. Problem nasz nie jest jednak tożsamy z poprzednim ze względu na obecność wierzchołków-zadań ze zbioru V_F. Dla krawędzi incydentych do takich wierzchołków w poprawnym harmonogramie muszą być spełnione dodatkowe warunki kolejnościowe związane z numeracją incydentnych doń procesorów. Pierwszą konsekwencją tego faktu jest zmiana zbioru barw zapewniających istnienie pokolorowania optymalnego $\{1,...,L\}$ – tym razem L nie może zostać wyrażone jako funkcja maksymalnego stopnia Δ. Możemy jednak przyjąć $L=m$, gdyż użycie na jakiejś krawędzi barwy większej od m oznaczałoby istnienie nieużywanego koloru i, takiego że pewne barwy większe od i pojawiałyby się na krawędziach. Wtedy jednak zmniejszenie wszystkich tych barw o 1 prowadziłoby do poprawnego harmonogramu o nie większej wartości funkcji kryterialnej. A zatem $F=\max_{e\in E, l\leq m} f(e,l)$.

Kolejna różnica pojawia się, gdy obliczamy TAB(v,H)[i] na podstawie TAB(w,T_j) w sytuacji, gdy wchodzą w grę ograniczenia kolejnościowe, tj. gdy $w \in V_F$. Jeżeli nie da się zapewnić warunków kolejnościowych dla kolorowania $c(vw)=i$, wówczas nadamy TAB(v,H)[i] $=2F$ jako „karę" za nielegalność harmonogramu – oczywiście uszeregowanie zawierające tak pokolorowaną krawędź nie może zostać zwrócone przez algorytm jako optymalne. Sposób postępowania w tym przypadku jest następujący. Załóżmy, że procesory incydentne do w, uporządkowane względem ich numeracji tworzą ciąg $v_1,\ldots,v_a,v,v_{a+1},\ldots,v_{\deg(w)-1}$. Dla danej wartości $f(vw,i) \le C \le 2F$ tworzymy dwa pomocnicze grafy dwudzielne: K_1 z pierwszą partycją kolejno ponumerowanych wierzchołków $kol_1,\ldots,kol_{i-1}$ i drugą $v_1,\ldots,v_a$ oraz K_2 z pierwszą partycją kolejno: $kol_{i+1},\ldots,kol_L$ i drugą $v_{a+1},\ldots,v_{\deg(w)-1}$. W obu grafach obowiązuje zasada wprowadzania krawędzi $kol_x v_y$, gdy TAB(v_y,T_y)[x]$\le C$. Łatwo sprawdzić, że przypisanie krawędziom barw o kosztach nie przekraczających C, spełniające warunki kolejnościowe, istnieje wtedy, gdy w K_1 i K_2 istnieją skojarzenia planarne nasycające drugą partycję (sprawdzalne w czasie $O(L\deg(w))$ na mocy twierdzenia 2.3). Oczywiście TAB(v,H)[i] jest równe najmniejszemu takiemu C, a gdy go brak – przypisujemy wartość $2F$.

Analiza złożoności całego algorytmu, przeprowadzona analogicznie do poprzedniego przypadku, daje wynik $O(n^{3.5}\log(F))$.

3.4. Zagadnienie X|ZUET,M=acyclic|Σfj

Jest to przypadek najbardziej skomplikowany ze względu na złożoną postać funkcji kryterialnej, która dla danego kolorowania krawędziowego c wynosi $\Sigma_{v \in V_2}\max\{f(v,c(e)): v \in e \in E\}$. Ograniczenie górne na ilość barw w pewnym pokolorowaniu optymalnym, podobnie jak w poprzednim przypadku, wynosi $L=m$, zatem $F=\max_{v \in V_2, l \le m} f(v,l)$. Przez TAB($v$,$H$)[$i$] rozumiemy minimalną możliwą wartość sumy z funkcji kryterialnej branej po wierzchołkach z $V_2 \cap V(H)-\{v\}$, przy czym rozważamy szeregowania podsystemu H o długości nie przekraczającej L, w których krawędź vw uzyskuje barwę i. Optymalna wartość funkcji kryterialnej dla całego systemu jest więc równa $\min_{i \le L}$ TAB(u,G)[i], gdy $u \in V_1$, zaś $\min_{i \le L}$ TAB(u,G)[i]$+f(u,i)$ dla $u \in V_2$, gdzie u jest wierzchołkiem wiszącym w opisującym cały system drzewie G.

Zgodnie z definicją dla $H=K_2$ mamy TAB(v,H)[i]=0 dla $v \in V_2$ i TAB(v,H)[i]$=f(w,i)$ dla $v \in V_1$. Przeanalizujmy sposób obliczania TAB(v,H)[i] przy znanych TAB(w,T_j). Możliwe są trzy przypadki.

- Wierzchołek $w \in V_1$. Kosztem przypisania odpowiednich kolorów do krawędzi wv_j jest suma odpowiednich kosztów po drzewach T_j. Budujemy pomocniczy pełny graf dwudzielny K o partycjach $\{kol_1,...,kol_L\}$ i $\{v_1,...,v_{\deg(w)-1}\}$, gdzie wagą krawędzi $kol_x v_y$ jest $\mathrm{TAB}(v_y, T_y)[x]$. Wówczas $\mathrm{TAB}(v,H)[i]$ jest wagą najlżejszego skojarzenia nasycającego drugą partycję w grafie $K-kol_i$. Wyliczenie całej tablicy $\mathrm{TAB}(v,H)$ można przeprowadzić za pomocą algorytmu z faktu 2.2 zastosowanego dla K, co ma złożoność $O(\deg(w)n^{1.5}\log(nF))$.
- Wierzchołek $w \in V_O$. Teraz do kosztów pokolorowania drzew T_j należy dodać koszt zakończenia zadania w w pewnej chwili $b \in \{\max\{\deg(w),i\},...,L\}$. A zatem dla każdego $i \in \{1,...,L\}$ oraz b jak wyżej liczymy $T_b[i]=f(w,b)+M$, gdzie M jest wagą najlżejszego skojarzenia nasycającego drugą partycję w grafie $K-\{kol_{b+1},..., kol_L, kol_i\}$ (graf obciążony K budujemy jak poprzednio). Następnie przyjmujemy $\mathrm{TAB}(v,H)[i]=\min_{b \in \{\max\{\deg(w),i\},...,L\}} T_b[i]$. Ze względu na konieczność powtarzania obliczeń dla wszystkich b złożoność uzyskania $\mathrm{TAB}(v,H)$ jest $O(\deg(w)n^{2.5}\log(nF))$ – zakładamy, że algorytm z faktu 2.2 wywołujemy dla każdego z grafów $K-\{kol_{b+1}, ..., kol_L\}$, $b \geq \deg(w)$.
- Wierzchołek $w \in V_F$. Niech procesory incydentne do w, uporządkowane względem ich numeracji tworzą ciąg $v_1,...,v_a,v,v_{a+1},...,v_{\deg(w)-1}$. Jest to przypadek zbliżony do poprzedniego, przy czym dodatkowo kolorowanie krawędzi przy w ma spełniać ograniczenia kolejnościowe. Dlatego obliczając $\mathrm{TAB}(v,H)[i]$ dla każdego możliwego terminu zakończenia $b \in \{\max\{\deg(w),i\},...,L\}$ zadania w tworzymy dwa pełne grafy dwudzielne o ponumerowanych wierzchołkach jak następuje: $K_1(b,i)$ ma w pierwszej partycji wierzchołki $kol_1,...,kol_{i-1}$, a w drugiej $v_1,...,v_a$, zaś $K_2(b,i)$ ma pierwszą partycję postaci $kol_{i+1},...,kol_b$, a drugą: $v_{a+1},...,v_{\deg(w)-1}$. W obu grafach wagą dla $kol_x v_y$ jest $\mathrm{TAB}(v_y, T_y)[x]$. Następnie $T_b[i]=f(w,b)+M_1(b,i)+M_2(b,i)$, gdzie $M_j(b,i)$ dla $j=1,2$ jest wagą najlżejszego skojarzenia planarnego nasycającego drugą partycję w $K_j(b,i)$ (w przypadku gdy któreś ze skojarzeń nie istnieje, przyjmujemy $T_b[i]=2nF$ jako „karę" za niespełnienie warunków kolejnościowych). Wreszcie mamy $\mathrm{TAB}(v,H)[i]=\min_{b \in \{\max\{\deg(w),i\},...,L\}} T_b[i]$. Oszacujmy teraz złożoność wyliczenia $\mathrm{TAB}(v,H)$. Policzenie $M_1(b,i)$ na mocy twierdzenia 2.4 trwa $O(\deg(w)n)$, co dla wszystkich badanych par liczb b i i daje $O(\deg(w)n^2)$, jako że $M_1(b,i)$ nie zależy od b. Podobnie, aby poznać wszystkie wielkości $M_2(b,i)$, na mocy uwagi pod twierdzeniem 2.4 wystarczy policzyć dla wszystkich i wartości $M_2(L,i)$, co znów wymaga $O(\deg(w)n^2)$ operacji.

Najbardziej czasochłonny okazuje się być przypadek drugi, dlatego złożoność obliczeniowa naszej procedury dla całego grafu wyniesie $O(n^{3.5}\log(nF))$.

LITERATURA

1. Giaro K., Kubale M.: Edge–chromatic sum of trees and bounded cyclicity graphs. Inf. Process. Lett. 75 (2000), 65 – 69.
2. Giaro K.: Szeregowanie zadań w acyklicznym systemie otwartym. Zesz. Nauk. AGH, Ser. Autom. 5 (2001), 213 – 220.
3. Giaro K., Kubale M., Małafiejski M., Piwakowski K.: Dedicated scheduling of biprocessor tasks to minimize mean flow time. LNCS (to appear).
4. Gonzalez T., Sahni S.: Open shop scheduling to minimize finish time. J. Assoc. Comput. Mach. 23 (1976), 665 – 679.
5. Kao M., Lam T. Sung W.: Ting H., All–cavity maximum matchings. in: Proc. ISAAC'97, LNCS 1350 (1997), 364 – 373.
6. Krawczyk H., Kubale M.: An approximation algorithm for diagnostic test scheduling in multicomputer systems. IEEE Trans. Comput. C–34 (1985), 869–872.
7. Micali S., Vazirani V.: An $O(mn^{1/2})$ algorithm for finding maximum matching in general graphs. Proc. 21st Ann. IEEE Symp. on Found. of Comput. Sci. (1980), 17 – 27
8. Mitchem J., Morriss P.: On the cost–chromatic number of graphs. Disc. Math. 171 (1997), 201–211.
9. Mitchem J., Morriss P., Schmeichel E.: On the cost chromatic number of outerplanar, planar and line graphs. Discussiones Mathematicae – Graph Theory 17 (1997) 1–13.
10. Zhou X., Nishizeki T.: Algorithm for the cost edge-coloring of trees. LNCS 2108 (2001), 288-297.

Recenzent: Prof. dr hab. inż. Jerzy Klamka

Abstract

We consider sparse systems of nonpreemptive 2-processor tasks with unit processing times as well as systems of dedicated machines (open shop, flow shop, mixed shop) with zero-unit processing times. We give a family of polynomial-time algorithms based on dynamic programming which yield solutions optimal with respect to a broad collection of criterion fun-ctions. The degree of sparseness of the system is determined by the density of the corres-ponding graph of the model and we are interested in systems whose scheduling graphs are acyclic. The main part of the algorithms are some procedures for finding special matchings in such graphs.