Marek KUBALE
Politechnika Gdańska

SZEREGOWANIE IDENTYCZNYCH ZADAŃ NA CZTERECH PROCESORACH JEDNORODNYCH Z DWUDZIELNYMI GRAFAMI KONFLIKTÓW

Streszczenie: W pracy rozważamy problem szeregowania n zadań jednostkowych na 4 procesorach jednorodnych o szybkościach $s_1 \geq s_2 \geq s_3 \geq s_4$. Celem szeregowania jest utworzenie najkrótszego możliwego harmonogramu. Zadania w naszym modelu podlegają ograniczeniom mówiącym, że niektóre pary zadań nie mogą być wykonywane na tym samym procesorze. Rozważamy dwa typy grafów niezgodności podyktowanych tymi ograniczeniami: dwudzielne grafy kubiczne (regularne stopnia 3) i dwudzielne grafy stopnia 4. Dla pierwszego przypadku podajemy algorytm dokładny, który rozwiązuje problem w czasie $O(n)$. W drugim przypadku podajemy algorytm przybliżony o złożoności $O(n^{1,5})$, który znajduje rozwiązanie optymalne, o ile $s_1 \geq 12s_2$ i $s_2 = s_3 = s_4$.

SCHEDULING OF IDENTICAL JOBS WITH BIPARTITE INCOMPATIBILITY GRAPH ON FOUR UNIFORM MACHINES

Summary: In the paper we consider the problem of scheduling n identical jobs on 4 uniform machines with speeds $s_1 \geq s_2 \geq s_3 \geq s_4$, respectively. Our aim is to find a schedule with minimum possible length. We assume that jobs are subject to some kind of mutual exclusion constraints modeled by a bipartite incompatibility graph of degree Δ, where two incompatible jobs cannot be processed on the same machine. We consider two kinds of incompatibility graphs: bipartite cubic graphs (i.e. regular of degree 3) and bipartite graphs of degree 4. In the first case we give a linear-time algorithm that solves the problem to optimality. In the second case we give an approximation algorithm which runs in $O(n^{1.5})$. This algorithm solves the problem to optimality if $s_1 \geq 12s_2$ and $s_2 = s_3 = s_4$.

1. Wprowadzenie

Wyobraźmy sobie, że jesteśmy organizatorami uroczystego bankietu dla 40 osób i mamy do dyspozycji 4 stoły o różnych rozmiarach, z których największy może pomieścić 16 osób. Wiemy, że wśród naszych gości są wegetarianie i nie wegetarianie. Co więcej, każdy wegetarianin jest w złych relacjach z co najwyżej 4 nie wegetarianami i na odwrót. Naszym zadaniem jest przydzielenie osób do stołów w taki sposób, by przy żadnym ze stołów nie było osób będących ze sobą w złych

stosunkach. W niniejszym artykule pokażemy, w jaki sposób można rozwiązać ten problem i inne podobne problemy metodami chromatycznego szeregowania zadań.

Nasz problem może być opisany w języku szeregowania zadań w następujący sposób. Przypuśćmy, że mamy n identycznych zadań $j_1,...,j_n$, które możemy opisać jako zadania o czasie trwania równym jednej jednostce czasu, symbolicznie p_i=1. Zadania te muszą być wykonane na czterech jednorodnych procesorach M_1, M_2, M_3 i M_4. Procesory te pracują z różną wydajnością, charakteryzowaną przez ich szybkość s_i, tzn. M_1 pracuje z szybkością s_1 itd. Bez straty ogólności zakładamy, że $s_1 \geq s_2 \geq s_3 \geq s_4$. Co ważne, procesory są jednorodne w tym sensie, że czas wykonania dowolnego zadania jednostkowego na procesorze M_i trwa $1/s_i$. Odpowiada to sytuacji, gdy nasze procesory są różnych generacji.

Tak zarysowany problem szeregowania byłby trywialny, gdyby nie było konfliktów między zadaniami. W związku z tym przyjmujemy, że pewne zadania nie mogą wystąpić na tym samym procesorze z pewnych powodów technologicznych. Ściślej, zakładamy, że każde zadanie może być w konflikcie z co najwyżej 4 innymi. Ponadto, bez straty ogólności zakładamy, że każde zadanie musi być w konflikcie z co najmniej jednym innym, gdyż zadania bezkonfliktowe mogą być łatwo przydzielone zachłannie do maszyn po ułożeniu podstawowego harmonogramu. Nasze zadania wraz z relacją konfliktów wyznaczają pewien graf G. Przyjmujemy, że graf ten jest dwudzielny, zaś jego stopień $\Delta(G)$, tj. maksymalny stopień wierzchołka, nie przekracza 4. Na przykład, wszystkie grafy pojawiające się na naszych rysunkach są dwudzielne. Zauważmy, że dwa zadania będące w konflikcie mogą być wykonane w przecinających się przedziałach czasowych. Grupa k zadań przydzielona procesorowi M_i będzie wymagała czasu k/s_i. Ponadto, będziemy używali zapisu $C(M_i)$ na oznaczenie czasu zakończenia pracy przez procesor M_i. Na mocy definicji, problem nasz można przedstawić jako podział grafu G na 4 zbiory niezależne I_1, I_2, I_3 i I_4. Dlatego w dalszym ciągu będziemy używali zamiennie terminów zadanie/wierzchołek oraz kolor/zbiór niezależny. Ponieważ wszystkie zadania muszą być wykonane, problem sprowadza się do takiego 4-pokolorowania grafu G, że długość uszeregowania $C_{\max} = \max\{|I_i|/s_i: i = 1,...,4\}$ jest zminimalizowana. Używając notacji 3-polowej nasz problem możemy zapisać jako $Q4|p_i$=1,G=*bipartite*$|C_{\max}$.

Ponieważ ten model szeregowania prowadzi do problemu NP-trudnego, w następnym punkcie rozważamy przypadek szczególny, gdy G jest kubiczny, tj. regularny stopnia 3. W tym przypadku udało się rozwiązać problem algorytmem liniowym. W kolejnym punkcie rozważamy grafy dwudzielne stopnia 4. Dla tych grafów podajemy algorytm przybliżony działający w czasie $O(n^{1,5})$. Algorytm ten staje się optymalny, gdy $s_1 \geq 12s_2$ i $s_2 = s_3 = s_4$. Na marginesie odnotujmy, że problem szeregowania grafów kubicznych na 3 procesorach był rozważany w [3].

2. Grafy kubiczne

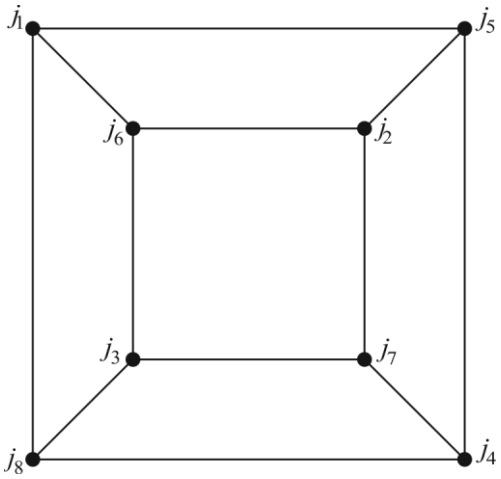
Najpierw wprowadzimy pojęcia podstawowe i oznaczenia. Grafem *kubicznym* nazywamy graf regularny stopnia 3. Niech będzie dany graf $G = (V,E)$; *k-kolorowanie* grafu G jest odwzorowaniem jego wierzchołków w zbiór $\{1,...,k\}$ takim, że końce każdej krawędzi mają różne kolory. Najmniejsze k, dla którego G jest *k-kolorowalny*, nazywamy *liczbą chromatyczną* grafu G i oznaczamy $\chi(G)$. Mówimy, że graf G =

(V,E) jest *sprawiedliwie* k-kolorowalny, jeżeli zbiór jego wierzchołków może być podzielony na k zbiorów niezależnych $V_1,...,V_k \subset V$ (niektóre mogą być puste) w taki sposób, iż $||V_i|-|V_j|| \leq 1$ dla wszystkich $i,j = 1,...,k$. Najmniejsze k pozwalające na takie pomalowanie wierzchołków nosi nazwę *sprawiedliwej liczby chromatycznej* $\chi_=(G)$. Mówimy, że graf G ma k-pokolorowanie *półsprawiedliwe* ($k \geq 3$), jeżeli istnieje podział jego wierzchołków na k zbiorów niezależnych $V_1,...,V_k \subset V$ taki, że jeden z tych podzbiorów, powiedzmy V_i, jest rozmiaru różnego od $\lfloor n/k \rfloor$ i $\lceil n/k \rceil$, zaś pozostały podgraf $G-V_i$ jest sprawiedliwie (k–1)-kolorowalny. W dalszym ciągu powiemy, że G ma ($V_1,...,V_k$)-pokolorowanie, gdy będziemy chcieli w sposób jawny wyrazić podział zbioru V na k zbiorów niezależnych. Jednakże, gdy wystarczy nam jedynie podkreślenie mocy zbiorów tego podziału, to użyjemy zapisu [$|V_1|,...,|V_k|$]. Na przykład graf z rysunku 3 ma jedno 2-pokolorowanie sprawiedliwe typu [4,4], kilka 4-pokolorowań sprawiedliwych typu [2,2,2,2], kilka 3-pokolorowań półsprawiedliwych typu [4,3,1] itd.

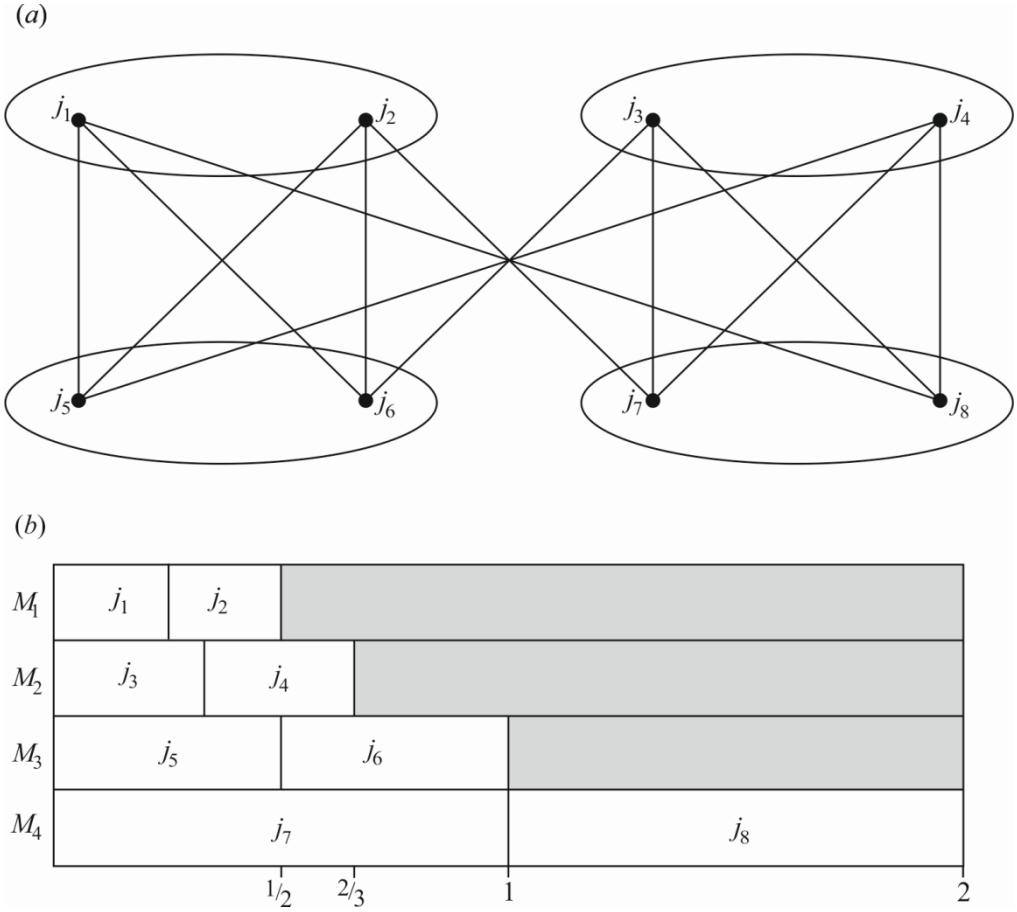
W dalszym ciągu skupimy się na własnościach chromatycznych grafów kubicznych. Przede wszystkim zauważmy, że grafy kubiczne mają parzystą liczbę wierzchołków oraz, że

$$2 \leq \chi_=(G) = \chi(G) \leq 4 \qquad (1)$$

dla dowolnego grafu kubicznego. Jednakże tutaj interesuje nas wyłącznie przypadek grafów *bikubicznych*, których liczba chromatyczna wynosi 2. Grafy te mają tylko jedno pokolorowanie typu [$n/2$, $n/2$], co oznacza, że mają również 4-pokolorowania sprawiedliwe typu [$\lceil n/4 \rceil$, $\lceil n/4 \rceil$, $\lfloor n/4 \rfloor$, $\lfloor n/4 \rfloor$]. Pokolorowanie to możemy uzyskać w dwóch krokach: najpierw w liniowym czasie otrzymujemy (A,B)-pokolorowanie przy okazji trawersowania grafu G metodą DFS, a następnie dzielimy oba zbiory A i B w miarę równo.



Rys. 1. Graf Q_3 jako kostka 3-wymiarowa



Rys. 2. (*a*) Sprawiedliwe 4-pokolorowanie Q_3; (*b*) odpowiadający mu harmonogram nieoptymalny, gdy $s_1 = 4$, $s_i = s_{i-1}-1$

Niech prędkości maszyn będą $s_1 \geq s_2 \geq s_3 \geq s_4$. Wówczas pokolorowanie optymalne nie musi być sprawiedliwe. Ogólnie, ładunek na każdej maszynie winien być proporcjonalny do jej szybkości, przy czym ładunek na żadnej maszynie nie może przekroczyć $n/2$ zadań. Ale z drugiej strony nasze zadania są niepodzielne, więc liczba zadań na każdej maszynie musi być całkowita. Dlatego, gdy $s_1 \geq s_2+s_3+s_4$, to na M_1 winniśmy położyć dokładnie $n/2$ zadań, a pozostałe procesory obciążyć proporcjonalnie do ich względnych prędkości, zaokrąglając ładunek na nich w górę lub w dół w zależności od tego, które rozwiązanie daje krótszy harmonogram. Jeżeli jednak nie ma tak wyraźnej dominacji M_1 nad pozostałymi maszynami, to wszystkie maszyny staramy się obciążyć proporcjonalnie do ich prędkości czyniąc odpowiednie zaokrąglenia w ramach posiadanych możliwości, tj. pamiętając, że spośród czterech maszyn muszą być dwie, takie że ich łączny ładunek wynosi dokładnie $n/2$ (a wówczas zachodzi to też dla pozostałych). Prowadzi to nas do algorytmu 1 dla szeregowania grafów bikubicznych.

Algorytm 1. Szeregowanie dla grafów bikubicznych.

Wejście: Graf bikubiczny G i 4 maszyny o szybkościach $s_1 \geq s_2 \geq s_3 \geq s_4$.
Wyjście: Uszeregowanie optymalne.
1. Znajdź (I,J)-pokolorowanie grafu G.

2. Jeśli $s_1 \geq s_2 + s_3 + s_4$, to idź do punktu 7.
3. Oblicz $n_1 = \frac{1}{2}ns_1/(s_1+s_2)$, $n_2 = \frac{1}{2}ns_2/(s_1+s_2)$, $n_3 = \frac{1}{2}ns_3/(s_3+s_4)$ i $n_4 = \frac{1}{2}ns_4/(s_3+s_4)$.
4. Sprawdź, które z poniższych rozwiązań częściowych gwarantuje lepsze rozwiązanie dla M_1 i M_2: $[\lceil n_1 \rceil, n/2-\lceil n_1 \rceil]$ czy $[\lfloor n_1 \rfloor, n/2-\lfloor n_1 \rfloor]$ i nazwij go OPT(1,2).
5. Sprawdź, które z poniższych rozwiązań częściowych gwarantuje lepsze rozwiązanie dla M_3 i M_4: $[\lceil n_3 \rceil, n/2-\lceil n_3 \rceil]$ czy $[\lfloor n_3 \rfloor, n/2-\lfloor n_3 \rfloor]$ i nazwij go OPT(3,4).
6. Przydziel $M_1 \leftarrow A$, $M_2 \leftarrow B$, $M_3 \leftarrow C$, $M_4 \leftarrow D$, gdzie A,B są zbiorami niezależnymi realizującymi OPT(1,2), zaś C,D są zbiorami niezależnymi realizującymi OPT(3,4) i stop.
7. Oblicz $n_2 = \frac{1}{2}ns_2/s$, $n_3 = \frac{1}{2}ns_3/s$, gdzie $s = s_2+s_3+s_4$.
8. Sprawdź, które z poniższych rozwiązań częściowych gwarantuje lepsze rozwiązanie dla M_2, M_3 i M_4:
   $[\lceil n_2 \rceil, \lceil n_3 \rceil, n/2-\lceil n_2 \rceil-\lceil n_3 \rceil]$, $[\lceil n_2 \rceil, \lfloor n_3 \rfloor, n/2-\lceil n_2 \rceil-\lfloor n_3 \rfloor]$,
   $[\lfloor n_2 \rfloor, \lceil n_3 \rceil, n/2-\lfloor n_2 \rfloor-\lceil n_3 \rceil]$, $[\lfloor n_2 \rfloor, \lfloor n_3 \rfloor, n/2-\lfloor n_2 \rfloor-\lfloor n_3 \rfloor]$
   i nazwij je OPT(2,3,4).
9. Przydziel $M_1 \leftarrow I$, $M_2 \leftarrow A$, $M_3 \leftarrow B$, $M_4 \leftarrow C$, gdzie A,B,C są zbiorami niezależnymi realizującymi OPT(2,3,4).

Powyższe rozważania prowadzą nas do następującego twierdzenia.

Twierdzenie 1. Algorytm 1 *tworzy rozwiązanie optymalne w czasie* $O(n)$. □

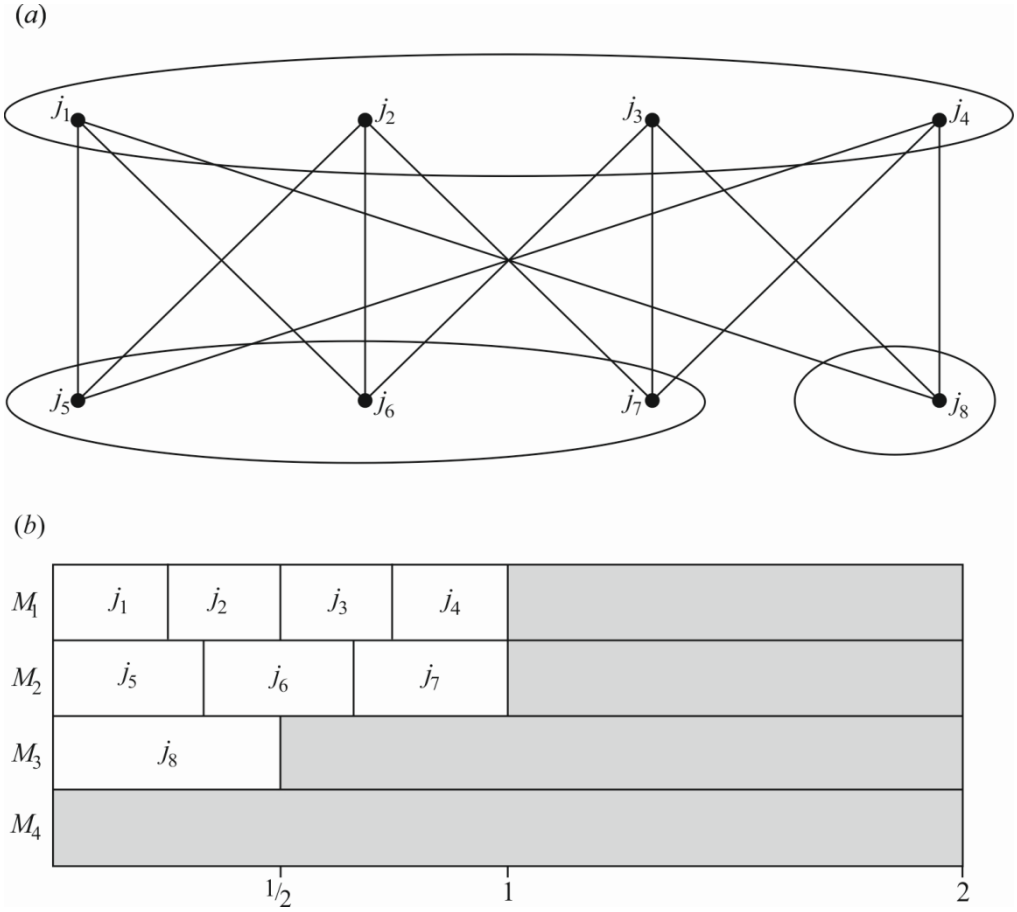
Przykład użycia algorytmu 1 dla grafu kostki 3-wymiarowej z rysunku 1 podajemy na rysunkach 2 i 3.

3. Grafy podbikwartyczne

Pod pojęciem grafu *podbikwartycznego* rozumieć będziemy graf dwudzielny stopnia $\Delta = 4$. Chen i Yan [2] udowodnili, że grafy te mają 4-pokolorowanie sprawiedliwe. W dalszym ciągu przez $\alpha(G)$ oznaczymy rozmiar największego zbioru niezależnego w G. W przypadku grafu dwudzielnego mamy $\alpha(G) \geq n/2$. Z drugiej strony maksymalna luka pomiędzy rozmiarami zbiorów niezależnych występuje dla $K_{1,\Delta}$. Zatem $\alpha(G) \leq n\Delta/(\Delta+1)$, zaś w naszym przypadku

$$\alpha(G) \leq 4n/5 \tag{2}$$

Zauważmy, że zbiór niezależny o rozmiarze $\alpha(G)$ może być znaleziony w grafie dwudzielnym w czasie $O(n^{1.5})$ poprzez wyznaczenie w kroku pośrednim maksymalnego skojarzenia w G [5].



Rys. 3. (*a*) Niesprawiedliwe 4-pokolorowanie Q_3;
(*b*) odpowiadający mu harmonogram optymalny, gdy $s_1 = 4$, $s_i = s_{i-1}-1$

Problem szeregowania zadań na 4 maszynach jednorodnych jest w przypadku ogólnym NP-trudny, o czym mówi

Twierdzenie 2 [4]. *Problem* $Q4|p_i=1,G=bipartite|C_{\max}$ *jest* NP-*trudny nawet wówczas, gdy* $s_1 = s_2 = s_3$. □

Dowód polega na redukcji z problemu podziału na ograniczone zbiory niezależne [1].

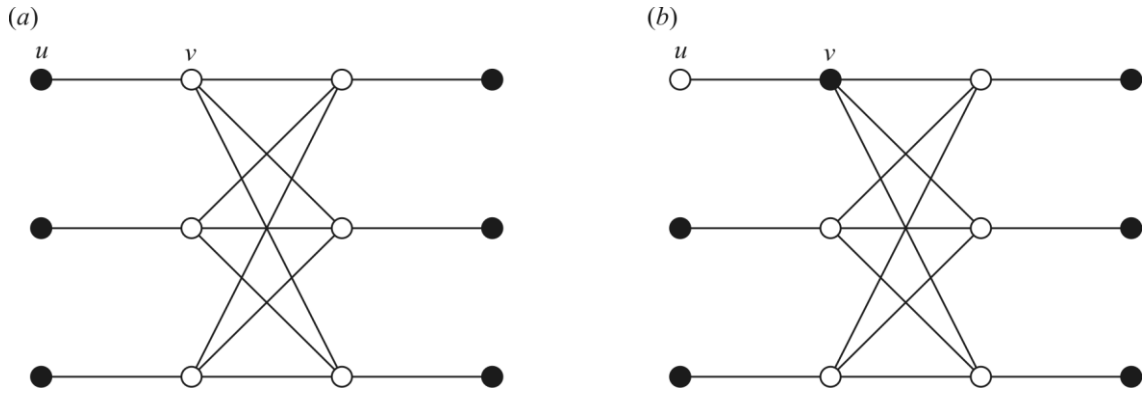
Ponieważ nie jesteśmy w stanie podać efektywnego algorytmu dokładnego, który byłby uniwersalny, podamy efektywny algorytm przybliżony. W tym celu przyjmiemy następującą definicję uszeregowania idealnego, które jest oszacowaniem dolnym uszeregowania optymalnego. Uszeregowanie, w którym wszystkie procesory kończą pracę w tej samej chwili czasu, nazywamy *idealnym*. Zauważmy, że długość uszeregowania idealnego wynosi n/s, gdzie $s = s_1+s_2+s_3+s_4$. Ponieważ liczba zadań na każdej maszynie musi być całkowita, harmonogram optymalny nie musi być idealny.

Niech $s_1 > s_2 = s_3 = s_4$. Wówczas jest rzeczą naturalną, by największy zbiór niezależny przydzielić procesorowi M_1, zaś pozostałe zadania rozdzielić równo pomiędzy pozostałe maszyny. Ideę tę realizuje algorytm 2.

Algorytm 2. Szeregowanie dla grafów podbikwartycznych.

Wejście: Graf podbikwartyczny G o $\Delta(G) \leq 4$ oraz 4 maszyny o szybkościach $s_1 > s_2 = s_3 = s_4$.
Wyjście: Uszeregowanie optymalne/suboptymalne.

1. Znajdź zbiór niezależny I_1 o mocy $\alpha(G)$.
2. Jeśli $K_{3,3}$ jest podgrafem G i zawiera 6 wierzchołków z sąsiedztwa $K_{3,3}$, to dokonaj zamiany $I_1 = I_1 \cup \{v\} - \{u\}$ jak na rys. 4.
3. Przydziel $M_1 \leftarrow I_1$.
4. Niech $G_1, G_2, ..., G_k$ będzie zbiorem składowych spójności grafu $G - I_1$.
5. Dla każdego $i = 1,2,...,k$ znajdź sprawiedliwe pokolorowanie $(A_i,B_i,C_i,)$ grafu G_i, gdzie
   $|A_i| \leq |B_i| \leq |C_i| \leq |A_i|+1$.
6. Połącz odpowiednie zbiory niezależne grafów G_i tak, aby otrzymać sprawiedliwe pokolorowanie (X, B, Y) grafu $G - I_1$, gdzie $B = B_1 \cup B_2 \cup ... \cup B_k$.
7. Przydziel $M_2 \leftarrow X$, $M_3 \leftarrow B$, $M_4 \leftarrow Y$.



Rys. 4. Graf $K_{3,3}$ i jego sąsiedztwo (czarne wierzchołki):
(a) przed zmianą; (b) po zmianie

Twierdzenie 3. *Jeżeli* $s_1 \geq 12s_2$, *to* algorytm 2 *gwarantuje rozwiązanie optymalne*.
Dowód. Niech I_1 będzie zbiorem niezależnym wyznaczonym algorytmem 2. Pokażemy, że $\Delta(G-I_1) \leq 3$. Jeśli G był stopnia co najwyżej 3, to nie ma czego dowodzić. A więc załóżmy, że $\Delta(G) = 4$ i niech v będzie dowolnym wierzchołkiem stopnia 4. Jeśli dla każdego takiego v mamy $v \in I_1$, to oczywiście $\Delta(G-I_1) \leq 3$. Jeśli zaś dla pewnego v mamy $v \notin I_1$, to co najmniej jeden z jego sąsiadów, powiedzmy u, należy do I_1, bo inaczej krawędź $\{u,v\}$ nie byłaby pokryta przez minimalne pokrycie wierzchołkowe i, na mocy twierdzenia Koniga [6], zbiór I_1 nie byłby maksymalny.

Załóżmy, bez straty ogólności, że $s_1 = 12s_2$. Wówczas długość uszeregowania idealnego jest $n/s = n/(15s_2)$. Na mocy nierówności (2) mamy $|I_1| \leq 4n/5$. Zatem $C(M_1) \leq 4n/5/12s_2 = n/(15s_2)$, co jest równe długości uszeregowania idealnego. Z drugiej strony algorytm 2 znajduje optymalne uszeregowanie na M_2, M_3 i M_4 dzięki sprawiedliwemu 3-kolorowaniu pozostałych wierzchołków. Pokolorowanie takie

wyznacza optymalne rozwiązanie całości, ponieważ nie ma możliwości przeniesienia jakiegokolwiek zadania z procesora M_i, $i = 1,2,3$ na M_1, skoro I_1 jest maksymalny. □

Twierdzenie 4. *Jeśli* $s_1 \geq s_2 \geq s_3 \geq s_4$, *to* algorytm 2 *nie jest c-aproksymacyjny dla żadnego skończonego c.*

Dowód. Pokażemy, że dla każdego $c > 0$ istnieje instancja problemu szeregowania, dla której $C_{max}/C^*_{max} > c$, gdzie C^*_{max} jest długością rozwiązania optymalnego.

Niech c będzie dowolną stałą i niech $s_1 = s_2 = s_3 = n/3$, $s_4 = 1$, gdzie n jest rzędem grafu G. Bez straty ogólności możemy przyjąć, że n jest wielokrotnością 6. Wówczas optymalne pokolorowanie G jest typu $[n/3, n/3, n/3, 0]$. Prowadzi to do uszeregowania o długości 1. Jednakże algorytm 2 w najgorszym przypadku może zwrócić pokolorowanie typu $[n/2, n/6, n/6, n/6]$, co prowadzi do harmonogramu o długości $C_{max} = C(M_4) = n/6$. Zatem $C_{max}/C^*_{max} = n/6$, co jest większe od c, gdy $n > 6c$. □

W podobny sposób można udowodnić następujące

Twierdzenie 5 [4]. *Jeżeli* $s_1 \geq 2s_2$ i $s_2 = s_3 = s_4$, *to* algorytm 2 *zwraca rozwiązanie o długości nie większej niż* $2C^*_{max}$. □

Podziękowanie

Artykuł był częściowo finansowany w ramach grantu NCN 2011/02/A/ST6/00201.

LITERATURA

1. Bodlaender H.L., Jansen K.: On the complexity of scheduling incompatible jobs with unit-times, LNCS 711, 1993, p. 291–300.
2. Chen B.-L., Yen C.-H.: Equitable Δ-coloring of graphs, Disc. Math. 312, 2012, p. 1512–1517.
3. Furmańczyk H., Kubale M.: Scheduling of unit-length jobs with cubic incompatibility graphs on three uniform machines, Disc. Appl. Math., w druku.
4. Furmańczyk H., Kubale M.: Scheduling of unit-length jobs with bipartite incompatibility graphs on four uniform machines, Bull. Polish Acad. Sci. - Tech. Sci., w druku.
5. Hopcroft J.E., Karp R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, SIAM J. Comput. 2, 1973, p. 225–231.
6. König D.: Gráfok és mátrixok (po węgiersku), Matematikai és Fizikai Lapok 38, 1931, p. 116–119.