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

Hanna FURMAŃCZYK, Paweł ŻYLIŃSKI
Uniwersytet Gdański

WSADOWE SZEREGOWANIE ZADAŃ JEDNOSTKOWYCH NA POJEDYNCZYM PROCESORZE

Streszczenie. Problem wsadowego szeregowania zadań jednostkowych na jednej maszynie polega na przydzieleniu zadań kompatybilnych do wsadów, które będą wykonywane w oddzielnych chwilach czasu. W pracy zajmiemy się przypadkiem, w którym graf modelujący problem jest grafem porównywalnym, split grafem albo kografem, a wielkość wsadu jest ograniczona przez stałą p. Naszym celem jest takie uszeregowanie zadań, aby łączny czas ich wykonywania był jak najmniejszy. Na koniec omówimy zastosowanie heurystyki wyżarzania symulowanego dla rozważanego problemu w przypadku ogólnym.

SCHEDULING UNIT TIME JOBS ON A SINGLE BATCH PROCESSING MACHINE

Summary. Problem of scheduling on a single batch processing machine reduces to attaching compatible jobs to batches. At most one batch can be treated at a time. In this paper we consider cases which are modeled by a graph which is a comparability graph, split graph or a cograph and the capacity of a batch is bounded by a constant p. Our aim is to schedule the jobs in such way that the makespan is as small as possible. In the last section we attemp to solve this problem in general case with the simulated annealing algorithm.

1. Wprowadzenie

W pracy będziemy rozważać problem wsadowego szeregowania n zadań jednostkowych $J_1,...,J_n$ na jednej maszynie B. Model ten jest rozszerzeniem klasycznego modelu szeregowania zadań, w którym na jednej maszynie w danej chwili czasu może być wykonywane co najwyżej jedno zadanie. Tutaj zakładamy, że w danej chwili może być wykonywany co najwyżej jeden wsad zadań. Grupę zadań, które mogą być wykonywane jednocześnie, tworzą zadania kompatybilne, np. podobne towary, które mogą być razem transportowane. Relację tę

przedstawiamy za pomocą grafu kompatybilności $G(V,E)$, w którym wierzchołki reprezentują zadania, a dwa wierzchołki są połączone krawędzią wtedy, gdy są kompatybilne, czyli mogą być wykonywane jednocześnie. Wielkość wsadu jest ograniczona przez stałą p. Naszym celem jest takie uszeregowanie zadań, aby łączny czas ich wykonywania był jak najmniejszy. Możemy zatem problem ten opisać następująco: $B|\ G = graph, p|\ C_{max}$, gdzie G oznacza graf kompatybilności, a *graph* jest konkretną klasą grafów rozważanego problemu. W pracy zajmiemy się przypadkiem zadań jednostkowych ($B|\ G = graph, p,\ p_i = 1|\ C_{max}$), zatem czas wykonywania wsadu jest również jednostkowy. Będziemy rozważać zagadnienie dla niektórych grafów *doskonałych*, mianowicie dla split grafów oraz dla grafów porównywalnych, w tym również dla kografów. Ponieważ Boudhar i Finke [2] pokazali, że problem $B|G=(V,E),p=2|C_{max}$ może być rozwiązany w czasie $O(n^{2.5})$, przyjmiemy, że $p \geq 3$.

Problem wsadowego szeregowania zadań jednostkowych na pojedynczym procesorze jest równoważny problemowi PODZIAŁU NA OGRANICZONE KLIKI grafu kompatybilności. Zagadnienie to definiujemy następująco:

Dane: Nieskierowany graf $G(V, E)$ oraz $k, p \in N$.

Pytanie: Czy istnieje podział V na kliki $K_1, K_2, ..., K_k$ takie, że $|K_i| \leq p$ dla $1 \leq i \leq k$?

W przypadku ogólnym problem podziału na ograniczone kliki jest NP-zupełny, gdyż zawiera problem PODZIAŁU NA KLIKI [6]. Zatem i problem wsadowego szeregowania zadań w przypadku ogólnym jest problemem NP-zupełnym. W ostatnim paragrafie omówimy zastosowanie heurystyki wyżarzania symulowanego właśnie dla przypadku ogólnego.

Szeregowanie wsadowe ma zastosowanie wszędzie tam, gdzie istnieje możliwość grupowania zadań, między którymi jest określona pewna relacja. Na przykład, wspomniany już transport różnych towarów czy malowanie pewnych elementów za pomocą maszyny, która może przyjmować po kilka części jednocześnie oraz daje możliwość użycia różnych kolorów farb w kolejnych jednostkach czasu. Wówczas naszym zadaniem jest taki podział towarów czy elementów, aby łączny czas wykonywania danej operacji był jak najkrótszy.

2. Grafy porównywalne

Grafy porównywalne są grafami odpowiadającymi częściowym porządkom [10]. Są to grafy nieskierowane $G(V,E)$ takie, że możliwe jest przyporządkowanie krawędziom kierunków w taki sposób, że jeśli (a,b) oraz (b,c) są łukami, to również (a,c) jest łukiem. Lonc [8]

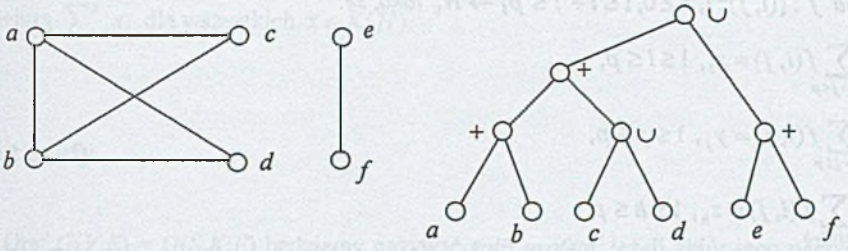
pokazał, że wsadowe szeregowanie dla tych grafów jest problemem NP-zupełnym. Jeżeli jednak ograniczymy się do węższej klasy grafów – do kografów, to problem staje się wielomianowy.

2.1. Kografy

Zaczniemy od zdefiniowania tej klasy grafów.

Definicja 1 [11]. *Kografy* stanowią najmniejszą rodzinę grafów zawierającą pojedynczy wierzchołek, zamkniętą na operacje sumy i zespolenia.

Przykładami kografów mogą być grafy puste, pełne, czy pełne r-dzielne. Kografy mają strukturę rekurencyjną. Możemy więc z każdym kografem G stowarzyszyć ukorzenione drzewo binarne, nazywane *kodrzewem* grafu G. Wierzchołki kodrzewa będziemy nazywali *węzłami*. Każdy węzeł nie będący liściem jest etykietowany albo "$\cup$" (jest węzłem sumy) albo "+" (jest węzłem zespolenia) i każdy z nich ma dokładnie dwóch potomków. Każdy węzeł kodrzewa odpowiada pewnemu kografowi, a węzeł-liść odpowiada pojedynczemu wierzchołkowi grafu G. Na rysunku 1 przedstawiony jest kograf G z odpowiadającym mu kodrzewem.



Rys.1. Kograf G oraz odpowiadające mu kodrzewo
Fig.1. A cograph G and its cotree

Istnieje liniowy algorytm ($O(n+m)$), podany przez Corneila i in. [4], sprawdzający, czy dany graf jest kografem i jeśli tak, wyznaczający odpowiadające mu kodrzewo.

Kografy mają wiele ciekawych własności. O jednej z nich mówi

Fakt 1. Dopełnienie kografu jest również kografem.

Nietrudno zauważyć jak będzie wyglądało kodrzewo dopełnienia kografu G. Strukturalnie będzie to takie samo kodrzewo z tym, że węzły sumy staną się węzłami zespolenia, a węzły

zespolenia – węzłami sumy. Nietrudno zatem przełożyć wyniki zagadnienia PODZIAŁU NA OGRANICZONE KLIKI na wyniki problemu PODZIAŁU NA OGRANICZONE ZBIORY NIEZALEŻNE. Przypadek ogólny pozostaje NP-zupełny dla tej klasy grafów.

Twierdzenie 1 [1].*Problem podziału na ograniczone kliki* (*i ograniczone zbiory niezależne*) *pozostaje NP-zupełny dla kografów.*

Jeśli jednak ustalimy ograniczenie p, to złożoność tych problemów będzie wielomianowa. Wykorzystując rekurencyjną strukturę kografu, wygenerujemy zbiór $K(G)$ zawierający ciągi odpowiadające wszystkim możliwym podziałom G na kliki o rozmiarze nieprzekraczającym p. Wektor $x = (x_1,...,x_p)$ należy do zbioru $K(G)$, jeżeli G można podzielić na kliki w taki sposób, że x_i klik ma rozmiar i dla $1 \le i \le p$. Wówczas wektor x będziemy nazywać *dopuszczalnym* dla grafu G.

Jeżeli $V = \{v\}$, to $K(H)$ składa się z jednego wektora (1,0,0,...,0). Dla większych kografów zbiór $K(H)$ konstruujemy rekurencyjnie, wykorzystując poniższy lemat.

Lemat 1 [1].*Niech* $G = (V,E)$ *będzie kografem.*

Jeśli $G = G_1 \cup G_2$, *to* $K(G) = \{x + y : x \in K(G_1), y \in K(G_2)\}$.

Jeśli $G = G_1 + G_2$, *to* $z \in K(G)$ *wtedy i tylko wtedy, gdy istnieją* $x \in K(G_1), y \in K(G_2)$ *i funkcja* $f : \{(i,j) : i,j \ge 0, 1 \le i + j \le p\} \to N_0$ *taka, że:*

1) $\sum_{j:i+j\le p} f(i,j) = x_i,\ 1 \le i \le p,$

2) $\sum_{i:i+j\le p} f(i,j) = y_j,\ 1 \le j \le p,$

3) $\sum_{i,j:i+j=h} f(i,j) = z_h,\ 1 \le h \le p.$

Dowód. Dla sumy $G = G_1 \cup G_2$ twierdzenie jest oczywiste. Jeśli $G = G_1 + G_2$, to niech $x \in K(G_1)$ i $y \in K(G_2)$ oraz niech $K_1, K_2, ..., K_k$ będzie podziałem na kliki o rozmiarze co najwyżej p. Klika o rozmiarze p występująca w grafie G jest dana albo przez klikę wielkości p w grafie G_1 albo G_2, albo też przez klikę wielkości i ($1 \le i < p$) w jednym z tych grafów i klikę rozmiaru $p - i$ w drugim. Dla kliki o rozmiarze mniejszym niż p mamy podobne przedstawienie. Używając x_i dla liczby klik rozmiaru i możemy opisać podział na kliki w G_1, G_2 przez takie odwzorowanie f.

Jako że ostatni wyraz wektora x jest jednoznacznie wyznaczony przez liczbę wierzchołków oraz pozostałe wyrazy tego wektora, to dla każdego kografu możliwych jest co najwyżej $O(n^{p-1})$ wektorów, co jest wartością wielomianową dla ustalonego p. Dla kografu $G = G_1 \cup G_2$ z rysunku 1 i $p = 3$ $K(G_1) = \{(4,0,0),(0,2,0),(2,1,0),(1,0,1)\}$, natomiast $K(G_2) = \{(0,1,0),\ (2,0,0)\}$. Zatem $K(G) = \{x + y : x \in K(G_1), y \in K(G_2)\} = \{(4,1,0),(6,0,0),$ $(0,3,0),(2,2,0),(4,1,0),(1,1,1),(3,0,1)\}$.

Twierdzenie 2 [1]. *Problem $B|G = kograf, p, p_i = 1|C_{max}$ może być rozwiązany w czasie wielomianowym.*

Dowód. Rozważmy podział na ograniczone kliki. Dla każdego węzła kodrzewa znajdujemy zbiór $K(H)$, gdzie H jest kografem związanym z tym węzłem. Określenie zbioru $K(G)$, gdy G jest sumą G_1 i G_2, może być wykonane w $O(n^{2(p-1)})$ krokach, przy danych zbiorach $K(G_1)$ i $K(G_2)$. Skoro zbiór $\{(i,j) : i,j \geq 0, 1 \leq i+j \leq p\}$ ma jedynie stałą liczbę $(p \cdot p + 3p)/2 \leq p \cdot p + 1$ elementów $(p \geq 2)$, to dla każdej pary wektorów x, y istnieje co najwyżej $O(n^{p \cdot p+1})$ możliwych odwzorowań f. Zatem, jeśli graf G jest zespoleniem G_1 i G_2, to $K(G)$ może być wyznaczone w czasie wielomianowym – w $O(n^{p(p+2)-1})$ krokach. W ten sposób możemy znaleźć $K(H)$ dla każdego kografu H przypisanego do danego węzła kodrzewa, a następnie wybrać wektor $x \in K(H)$ z minimalną liczbą klik, to znaczy z najmniejszą $\sum_{i=1}^{p} x_i$ dla wszystkich $x \in K(H)$.

3. Split grafy

Graf $G(V,E) = G(S,K;E)$ będziemy nazywać *split grafem*, jeżeli zbiór jego wierzchołków V możemy podzielić na zbiór niezależny S oraz klikę K takie, że $V = S \cup K$. Krawędzie w split grafie pomiędzy wierzchołkami z S i K mogą występować dowolnie. Podobnie jak kografy, split grafy posiadają tę własność, że dopełnienie split grafu jest również split grafem. Zatem problem podziału na ograniczone kliki jest równoważny problemowi podziału na ograniczone zbiory niezależne. Tym ostatnim problemem zajmował się Chen i in. w [3]. My przedstawimy wielomianowy algorytm optymalnie grupujący zadania w kliki.

Twierdzenie 3. *Problem $B|G$=split graf, $p, p_i = 1|C_{max}$ może być rozwiązany w czasie wielomianowym.*

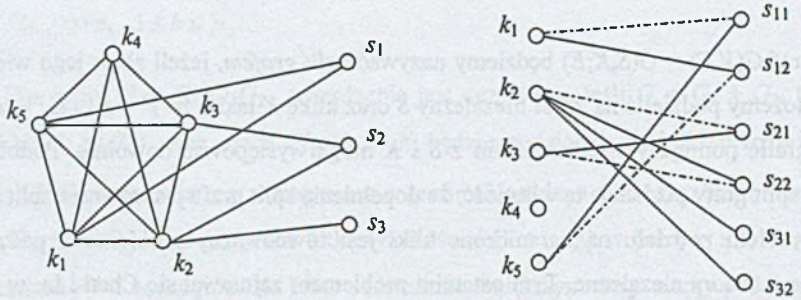
Dowód. Prawdziwość twierdzenia dla $p = 2$ pokazali Boudhar i Finke w [2]. Możemy zatem przyjąć, że $p \geq 3$. Wówczas dla split grafu $G(K,S;E)$, gdzie $K = \{k_1, k_2, ..., k_t\}$ indukuje klikę i $S = \{s_1, s_2, ..., s_r\}$ jest niezależnym zbiorem wierzchołków, konstruujemy graf dwudzielny $BG(p\text{-}1)$, którego zbiorem wierzchołków jest zbiór $K \cup \{s_{ij} : 1 \leq i \leq r \text{ i } 1 \leq j \leq p-1\}$ oraz $\{k_i, s_{ij}\}$ jest krawędzią wtedy i tylko wtedy, gdy $\{k_i, s_i\} \in E(G)$.

Dla tak skonstruowanego grafu dwudzielnego szukamy maksymalnego skojarzenia. Jego moc oznaczymy przez $\alpha'(BG(p-1))$. Wsad B_l będą tworzyć zadania: s_l wraz z zadaniami k_i takimi, że krawędź $\{k_i, s_{lq}\}$ należy do maksymalnego skojarzenia $BG(p-1)$ dla pewnego q, $1 \leq q \leq p-1$. W ten sposób utworzymy r grup zadań. Pozostałe, nieuszeregowane zadania z kliki K przydzielimy do następnych wsadów, co najwyżej p w każdym. Zatem optymalny czas wykonania wszystkich zadań możemy określić wzorem:

$$\min C_{\max} = r + \left\lceil \frac{t - \alpha'(BG(p-1))}{p} \right\rceil .$$

Złożoność algorytmu szeregowania zadań dla grafu kompatybilności będącego split grafem zależy głównie od złożoności szukania maksymalnego skojarzenia grafu dwudzielnego. Jeżeli zastosujemy algorytm, którego autorami są Micali i Vazirani [9], to algorytm ten będzie wykonywał $O(mn^{0.5}p^{1.5})$ kroków, gdzie n oznacza liczbę wierzchołków, a m liczbę krawędzi split grafu.

Na rysunku 2 mamy przykład split grafu oraz odpowiadającego mu grafu dwudzielnego $BG(2)$, czyli $p = 3$. Krawędzie należące do maksymalnego skojarzenia zostały zaznaczone linią przerywaną. Zatem wsady będą utworzone następująco: $B_1 = \{s_1,k_1,k_5\}$, $B_2 = \{s_2,k_2,k_3\}$, $B_3 = \{s_3\}$, $B_4 = \{k_4\}$.



Rys.2. Graf $G(K,S;E)$ oraz odpowiadający mu graf $BG(2)$

Fig.2. A graph $G(K,S;E)$ and its coresponding graph $BG(2)$

4. Symulowane wyżarzanie

W paragrafie tym opiszemy zastosowanie heurystyki symulowanego wyżarzania (ang. *simulated annealing*) w problemie znajdowania minimalnego podziału na ograniczone kliki.

Przypomnijmy, że w algorytmie symulowanego wyżarzania z otoczenia $N(x_0)$ bieżącego rozwiązania x_0 wybiera się dowolne rozwiązanie x i następnie, jeżeli wartość minimalizowanej (lub maksymalizowanej) funkcji celu F jest mniejsza, tzn. $F(x)<F(x_0)$, to rozwiązanie x przyjmowane jest jako najlepsze znalezione do tego czasu rozwiązanie ($x_0:=x$); jeżeli natomiast $F(x)>F(x_0)$, to rozwiązanie x przyjmowane jest jako najlepsze z prawdopodobieństwem P zależnym od tego, o ile rozwiązanie x jest gorsze od x_0. W typowym algorytmie prawdopodobieństwo P określone jest wzorem $P = P(x_0, x, T) = \exp^{-(F(x)-F(x_0))/t}$, gdzie parametr t (zwany temperaturą) stabilizuje algorytm, tzn. im temperatura jest niższa, tym prawdopodobieństwo P jest mniejsze; parametr t jest zmienny w czasie: na początku ma wysoką wartość, w miarę upływu czasu dąży do zera.

Tutaj jako rozwiązanie początkowe x_0 przyjmujemy albo podział na podzbiory złożony z pojedynczych wierzchołków, albo podział złożony z całego zbioru wierzchołków, w zależności od tego, jak bardzo graf jest gęsty.

Relacja sąsiedztwa określona jest następująco: niech $x_1=\{V_1,...,V_k\}$ będzie podziałem zbioru V na rozłączne podzbiory. Wówczas sąsiedztwo $N(x_1)$ tworzą te wszystkie podziały x, które mogą być otrzymane z podziału x_1 poprzez usunięcie dowolnego wierzchołka v z dowolnego podzbioru V_i i wstawienie go do innego podzbioru $V_{j\neq i}$, bądź też utworzenie nowego jednoelementowego podzbioru $V_{k+1}=\{v\}$; np. do sąsiedztwa $N(x_1)$ podziału $x_1=\{\{1,2\},\{3,4\},\{5,6\}\}$ należy podział $x_2=\{\{1\},\{2,3,4\},\{5,6\}\}$, jak i podział $x_3=\{\{1,2\},\{3,4\},\{5\},\{6\}\}$, natomiast $x_3\notin N(x_2)$.

Przy tak określonej relacji sąsiedztwa wygenerowanie losowego rozwiązania sąsiedniego polega na wylosowaniu podzbioru V_i (czyli liczby i, gdzie $1\leq i\leq k$), następnie wylosowaniu elementu $v\in V_i$ (czyli liczby j, gdzie $1\leq j\leq|V_i|$) oraz wylosowaniu podzbioru, do którego będziemy wstawiać element (czyli znowu liczby l, gdzie $1\leq l\leq k$) przy założeniu jednakże, że jeśli wylosowano ten sam zbiór, z którego usuwamy element, wówczas tworzymy nowy jednoelementowy zbiór $V_{k+1}=\{v\}$ złożony z elementu v. Oczywiście, może się zdarzyć, że będziemy usuwać element z jednoelementowego podzbioru i tworzyć z niego znowu jednoelementowy zbiór, ale tę sytuację łatwo wyeliminować; są to już szczegóły techniczne.

Niech p będzie ograniczeniem na rozmiar kliki. Dla dowolnego podziału x_0 funkcję celu określamy następująco:

$$F(x_0)=|x_0|+2e(x_0)+p(x_0)$$

gdzie:

$|x_0|$ – liczba podzbiorów; tak naprawdę pseudoklik, bo przy tak określonej funkcji celu kolejne rozwiązania nie muszą być do końca dobre w takim sensie, że podzbiory wierzchołków nie będą tworzyć klik;

$e(x_0)=\sum_i e(V_i)$, gdzie $e(V_i)$ jest liczbą brakujących krawędzi, aby podzbiór V_i tworzył klikę;

$p(x_0)$ – ma związek z ograniczeniem na rozmiar klik, i analogicznie jak wyżej, $p(x_0)=\sum_{i=1}^{k} p(V_i)$, gdzie $p(V_i)=\lceil(|V_i|-p)\operatorname{div} p\rceil$; takie określenie $p(V_i)$ wynika z tego, że podzbiory o mocach $jp+1$, $jp+2$,..., $jp+p$ (j dowolne naturalne) można podzielić na dokładnie $j+1$ podzbiorów o mocy nie przekraczającej p.

Jak łatwo zauważyć, zminimalizowanie tej funkcji równoważne jest znalezieniu minimalnego podziału zbioru wierzchołków na kliki z ograniczeniem p.

Obliczenie tak określonej funkcji celu dla rozważanego podziału wiąże się z przeglądnięciem tablicy sąsiedztwa, czyli wymaga czasu rzędu $\Theta(n^2)$. Ale funkcja celu musi być obliczana w jak najkrótszym czasie. Jeśli struktura przechowująca podział wierzchołków na podzbiory dla każdego z podzbiorów V_i będzie przechowywać opisaną wyżej przy definicji funkcji celu wartość $e(V_i)$, wówczas wyliczenie funkcji wymagać będzie czasu rzędu $O(n)$.

Przechowywanie dla każdego z podzbiorów V_i wartości $e(V_i)$ wymaga modyfikacji procedury losującej: oprócz wylosowania „skąd-co-dokąd", po wygenerowaniu nowego podziału należy uaktualnić wartości funkcji $e(\cdot)$, ale tylko dla podzbiorów „skąd" i „dokąd", a to, znając indeks usuwanego/wstawianego wierzchołka, można zrobić w czasie $O(n)$. Oczywiście przez to wzrasta koszt wygenerowania rozwiązania sąsiedniego, ale przy takim modelu czas obiegu najbardziej wewnętrznej pętli algorytmu SA będzie rzędu $O(n)$, a nie $\Theta(n^2)$.

W algorytmie SA temperatura t_0 początkowa powinna być na tyle wysoka, aby na początku działania algorytmu większość gorszych rozwiązań mogła być akceptowana z dość dużym prawdopodobieństwem. Tutaj, doświadczalnie, przyjęliśmy $t_0=1$. Istnieje wiele

sposobów aktualizacji temperatury t: przyjęliśmy model geometryczny, tzn. nową temperaturę t uzyskuje się ze wzoru $t = \alpha * t$, gdzie $\alpha = 0{,}9$.

Sprawdzanie, czy układ jest w równowadze, polega na utrzymywaniu temperatury t przez

$$\begin{cases} 10n & \text{prób} \\ n & \text{prób udanych } (F(x) < F(x_0)) \end{cases},$$

gdzie n jest liczbą wierzchołków grafu. Kryterium stopu jest osiągnięte po obniżeniu temperatury n razy. Ostatecznie daje to czas działania rozważanego algorytmu SA rzędu $O(n^3)$.

Jednakże jak wiadomo, algorytmy SA nie zawsze dają optymalne rozwiązanie. Przy tak określonej funkcji celu może się zdarzyć, że w otrzymanym podziale na podzbiory będą istniały podzbiory, które nie tworzą klik. W takiej sytuacji należy te podzbiory znowu podzielić na minimalne kliki z ograniczeniem p, używając ponownie algorytmu SA. Prowadzi to do algorytmu GSA, dla którego otrzymujemy następujące równanie rekurencyjne, przy założeniu że $T(n)$ opisuje czas działania dla danych rozmiaru n:

$$T(n)=T(n_1)+T(n_2)+...T(n_k)+O(n^3), \text{ gdzie } n_1+n_2+...+n_k=n$$

Powyższe równanie prowadzi do oszacowania $T(n)=O(n^4)$: najgorsza z możliwych sytuacji to ta, kiedy kolejne wywołania wykonywane będą dla podziałów $n_1=n-1$, $n_2=1$.

```
algorytm GSA(V).;
begin
wygeneruj podział y algorytmem SA(V);
   for all Vi ∈ y do begin
      if e(Vi)=0 then przepisz Vi do y* dzieląc Vi na podzbiory o mocy ≤ p
          else begin
          utwórz podział x zbioru Vi algorytmem GSA(Vi);
          przepisz podział x do y* dzieląc go na podzbiory o mocy ≤ p;
          end;
      end;
y* jest rozwiązaniem końcowym;
end;
```

We wstępnych badaniach grafów o liczbie wierzchołków n=10, 20, 50, 100 i ograniczeniu p=3 otrzymano, że np. dla ścieżek P_n na n prób maksymalny błąd względny był odpowiednio równy (0,4;0,1;0,12;0,06), średni błąd względny: (0,18;0,04;0,04;0,03). Dla kół W_n odpowiednio: (0,2;0,1;0,04;0,04) i (0,06;0,01;0,1;0,02). Dla niespójnych grafów składających się z losowej liczby grafów pełnych odpowiednio: (0,25;0;0;0) i (0,08;0;0;0). Dla pewnej klasy losowych kografów: (0,25;0,14;0,11;0,07) i (0,13;0,03;0,02;0,01).

LITERATURA

1. Bodleander H.L., Jansen K.: Restrictions of graph partition problems, Theoretical Computer Science 148 (1995), s. 93-109.
2. Boudhar M., Finke G.: Scheduling on a batch machine with job compatibilities. Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL) (2001), to appear.
3. Chen B.L., Ko M.T., Lih K.W.: Equitable and m-bounded coloring of split graphs, Lecture Notes in Computer Science 1120 (1996), s. 1-5.
4. Corneil D.G., Perl Y., Stewart L.K.: A linear recognition algorithm for cographs, SIAM J. Comput. 4 (1985), s. 926-934.
5. Johnson D.S.: The NP-completness column: an ongoing guide. Journal of Algorithms, 6 (1985), s. 434-451.
6. Karp R.M.: Reducibility among combinatorial problems. W: Complexity of Computer Computations, Plenum Press, Oxford (1972), s. 85-104.
7. Kirkpatrick S., Gelatt C.D. Jr., Vecchi M.P.: Optimization by Simulated Annealing, Science, 220, 4598, s. 671-680, 1983.
8. Lonc Z.: On complexity of some chain and antichain partition problems. Graph Theoretical Concepts in Computer Science, WG' 91, Lecture Notes in Computer Science 570, s. 97-104.
9. Micali S., Vazirani V.V.: An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs, Proc. 21st Ann. IEEE Symp. on Foundations of Computer Science (1980), s. 17-27.
10. Möhring R.H.: Computationally tractable classes of ordered sets. In: Algorithms and Order (ed. I. Rival), Reidel Publishing Co., Dordrecht 1989, s. 105-194.
11. Seinsche D.: On a property of the class of n-colorable graphs, J. Combin. Theory, Ser. B 16 (1974), s. 191-193.

Recenzent: Prof. dr hab. inż. Jerzy Józefczyk

Abstract

In this paper we consider a problem of scheduling unit time jobs on a single batch processing machine. In this model at most one batch can be treated at a time. A batch consists of compatible jobs and its capacity is bounded by a constant p. In terms of graph theory it forms a clique. Our aim is to schedule jobs in such way that the makespan is as small as possible. The problem is modeled by a comparability graph, in which vertices stand for jobs and two vertices are adjacent if they correspond to compatible jobs. In general, the problem is NP-complete. We show some special cases, for which this model of scheduling is polynomially solvable. These include the following classes of comparability graphs: comparability, split graph and a cograph. In the last section we attemp to solve the problem in general case with the simulated annealing algorithm.