ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 123 Nr kol. 1389

Hanna DAMRATH, Marek KUBALE
Uniwersytet Gdański, Politechnika Gdańska

ALGORYTMY SPRAWIEDLIWEGO KOLOROWANIA GRAFÓW

Streszczenie. Problemy kolorowania grafów należą do najtrudniejszych problemów optymalizacji kombinatorycznej z punktu widzenia złożoności obliczeniowej. W niniejszej pracy rozważono zagadnienie sprawiedliwego kolorowania wierzchołków grafu, tj. takiego kolorowania, że krotności użycia dowolnych kolorów różnią się najwyżej o 1. Pokazano, że problem ten jest NP-trudny i sformułowano dwa algorytmy heurystyczne dla usprawiedliwiania rozwiązań uzyskanych innymi algorytmami kolorowania. Podano doświadczenia komputerowe z implementacji i testowania tych algorytmów na identycznych seriach grafów pseudolosowych.

ALGORITHMS FOR EQUITABLE GRAPH COLORING

Summary. Graph coloring problems belong to the hardest combinatorial optimization problems with respect to the computational complexity. In this paper we consider the problem of equitable coloring the vertices of a graph, i.e. such a coloring that the sizes of color classes differ by at most 1. We show that the problem is NP-hard and give two heuristic algorithms for transforming colorings obtained by other standard algorithms into equitable solutions. General considerations are supported by computational experience gained with implementation and profiling of these algorithms on identical series of pseudorandom graphs.

1. Wprowadzenie

W niektórych dyskretnych systemach przemysłowych spotykamy się z problemem optymalnego podziału zbioru zawierającego konflikty na równoliczne podzbiory bezkonfliktowe. Sytuacje takie mogą być modelowane za pomocą sprawiedliwego kolorowania grafów, tj. takiego kolorowania wierzchołków, aby krotności użycia kolorów różniły się co najwyżej o 1. Na przykład w problemie oczyszczania miasta wierzchołki grafu reprezentują trasy wywozu śmieci, zaś dwa wierzchołki połączone są krawędzią, gdy odpowiadające im trasy nie mogą być obsłużone tego samego dnia. Problem przydziału jednego z 6 dni pracy do każdej trasy sprowadza się do pokolorowania odpowiedniego grafu sześcioma kolorami. W praktyce, z uwagi na ograniczenia taboru, chcielibyśmy, aby w każdym

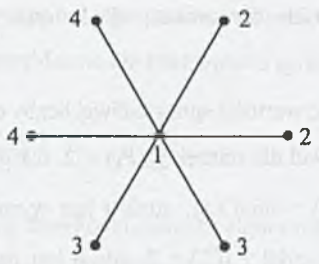
dniu obsłużyć możliwie taką samą liczbę tras. Należy więc pokolorować nasz graf sprawiedliwie sześcioma kolorami.

Problem optymalnego kolorowania sprawiedliwego grafów ogólnych jest NP-trudny. Oznacza to, że w praktyce zmuszeni jesteśmy poszukiwać uproszczonych struktur grafów dopuszczających konstrukcję wielomianowych algorytmów dokładnych bądź też zmuszeni jesteśmy do stosowania uniwersalnych algorytmów przybliżonych.

Niech $\chi_{=}(G)$ oznacza sprawiedliwą liczbę chromatyczną grafu G, tj. najmniejszą liczbę kolorów dopuszczającą takie pokolorowanie wierzchołków. W niniejszej pracy podamy klasy grafów, dla których liczba ta może być ustalona w czasie wielomianowym. Do grafów tych należą m.in. drzewa, koła, hiperkostki i grafy kubiczne. Z drugiej strony wiadomo, że $\chi_{=}(G) \leq \Delta+1$, gdzie Δ jest maksymalnym stopniem grafu. Istnieje hipoteza *ECC* (ang. *Equitable Coloring Conjecture*) mówiąca, że jeżeli G nie jest ani cyklem nieparzystym, ani grafem pełnym, to jedynka po prawej stronie nierówności może być pominięta. Hipoteza ta została udowodniona, poza wyżej wymienionymi, dla wszystkich grafów dwudzielnych oraz trójdzielnych grafów kubicznych. Jednakże nawet oszacowanie $\chi_{=}(G) \leq \Delta$ nie jest zbyt dokładne i w praktyce stosuje się algorytmy heurystyczne dla sprawiedliwego kolorowania grafów. W niniejszej pracy podamy dwa takie algorytmy wielomianowe. Oba algorytmy zostały zaimplementowane na komputerze osobistym i przetestowane na serii grafów pseudo-losowych o gęstości 0.1 i 0.5. W artykule podamy doświadczenia komputerowe związane z implementacją owych algorytmów oraz wnioski wynikające z tych eksperymentów.

2. Najważniejsze wyniki teoretyczne

Niech będzie dany graf spójny $G = (V,E)$ o $|V| = n$ wierzchołkach i $|E| = m$ krawędziach. Przez $\Delta = \Delta(G)$ oznaczymy maksymalny stopień wierzchołka grafu G, zaś przez $\alpha = \alpha(G)$ *liczbę stabilności* grafu G, tj. maksymalny rozmiar podgrafu pustego zawartego w G. Liczbę chromatyczną grafu G oznaczymy symbolem $\chi(G)$. Jeżeli wierzchołki grafu G można podzielić na k klas $V_1, V_2,..., V_k$ takich , że każde V_i jest zbiorem wierzchołków niezależnych i $||V_i| - |V_j|| \leq 1$ dla wszystkich $i \neq j$, to mówimy, że graf G jest *sprawiedliwie kolorowalny k* kolorami. Najmniejsze k, dla którego G może być sprawiedliwie pokolorowany k kolorami, nazywamy *sprawiedliwą liczbą chromatyczną* grafu G i oznaczamy przez $\chi_{=}(G)$. Przykład sprawiedliwego pokolorowania gwiazdy podano na rys. 1.



Rys.1. Sprawiedliwe pokolorowanie gwiazdy $K_{1,6}$
Fig.1. Equitable coloring of star $K_{1,6}$

Pojęcie sprawiedliwego kolorowania grafu i związaną z nim hipotezę *ECC* wprowadził Meyer [7]. Jednakże, jeszcze 3 lata wcześniej Hajnal i Szemeredi [3] pokazali, że każdy graf G stopnia Δ jest sprawiedliwie k-kolorowalny, jeśli $k > \Delta$. Meyer wysunął przypuszczenie, że $\chi_=(G) \leq \Delta$ dla wszystkich G z wyjątkiem pełnych K_n i cykli C_{2n+1}. Hipoteza ta została potwierdzona dla grafów o $n \leq 6$, grafów dwudzielnych [5], grafów podkubicznych z wyjątkiem K_4 [2] oraz kół.

W następnym punkcie pokażemy, że optymalne kolorowanie sprawiedliwe jest problemem NP-trudnym. Zatem w praktyce zmuszeni jesteśmy do korzystania z oszacowań i algorytmów heurystycznych. Poniżej podamy najważniejsze oszacowania sprawiedliwej liczby chromatycznej z dołu i z góry. Nasz przegląd zaczynamy od oczywistego oszacowania dolnego

$$\chi(G) \leq \chi_=(G). \tag{1}$$

Oszacowanie to jest dość kiepskie dla gwiazd, gdyż różnica $\chi_=(K_{1,n+1})-\chi(K_{1,n+1}) = \lceil n/2 \rceil+1-2 = \lceil n/2 \rceil-1$ może być dowolnie duża, gdy $n\to\infty$. Obecnie przypuśćmy, że G jest sprawiedliwie pokolorowany i że wierzchołek v ma kolor 1. Łączna liczba wierzchołków pokolorowanych jedynką nie przekracza $\alpha(G-N[v])+1$, gdzie $N[v]$ jest zamkniętym sąsiedztwem wierzchołka v. Ponieważ mamy kolorowanie sprawiedliwe, więc krotność użycia każdego innego koloru nie przekracza $\alpha(G-N[v])+2$. Stąd

$$\left\lceil \frac{n+1}{\alpha(G-N(v))+2} \right\rceil \leq \chi_=(G). \tag{2}$$

Jeśli natomiast chodzi o ograniczenie górne, to wspomniane wyżej mówi, że

$$\chi_=(G) \leq \Delta + 1. \tag{3}$$

Jak poprzednio, jest ono kiepskie dla gwiazd, dla których $\Delta(K_{1,n+1})+1-\chi_=(K_{1,n+1}) = n+1-(\lceil n/2\rceil+1) = \lfloor n/2\rfloor$.

Ponadto znamy dokładne wartości sprawiedliwej liczby chromatycznej grafów o bardzo regularnej strukturze. Na przykład dla ścieżek $\chi_=(P_n) = 2$, dla gwiazd $\chi_=(K_{1,n+1}) = \lceil n/2\rceil+1$, dla pełnych drzew binarnych $\chi_=(T_t) = \min\{3,t\}$, gdzie t jest wysokością drzewa. Następnie, dla grafów pełnych $\chi_=(K_n) = n$, dla cykli $\chi_=(C_n) = 2$, gdy n jest parzyste i $\chi_=(C_n) = 3$, gdy n jest nieparzyste. Wreszcie dla kół $\chi_=(W_n) = \lceil (n+1)/2\rceil$, zaś dla hiperkostek $\chi_=(Q_d) = 2$, gdzie d jest wymiarem hiperkostki. Ogólnie, Chen i Lih [1] pokazali, że dla wszystkich drzew $T = T(X,Y)$

$$\chi_=(T) = \begin{cases} 2 & \text{gdy} \big||X|-|Y|\big| \le 1 \\ \min\limits_{w\in X\cup Y} \max\{3,\lceil (n+1)/(\alpha(T-N(v))+2)\rceil\} & \text{gdy} \big||X|-|Y|\big| > 1 \end{cases} \tag{4}$$

3. Dowód NP-zupełności

W dalszym ciągu potrzebne nam będzie pojęcie grafu krawędziowego. Niech będzie dany graf $G = (V,E)$. *Grafem krawędziowym* $L(G)$ grafu G nazywamy graf m-wierzchołkowy, w którym każdy wierzchołek odpowiada pewnej krawędzi grafu G, zaś dwa wierzchołki są sąsiednie, gdy odpowiadające im krawędzie w G stykają się w wierzchołku $v\in V$. Problem kolorowania krawędzi grafu kubicznego jest NP-trudny [4]. Fakt ten wykorzystamy w dowodzie następującego twierdzenia.

Twierdzenie. *Problem odpowiedzi na pytanie, czy* $\chi_=(G) \le 3$ *jest NP-zupełny również, gdy G jest grafem krawędziowym grafu kubicznego.*

Dowód. Niech G_3 będzie grafem kubicznym grafu krawędziowego G, tzn. $G = L(G_3)$. Wiadomo, że indeks chromatyczny $\chi'(G_3) = 3$ lub 4, przy czym $\chi'(G_3) = 3$ wtedy i tylko wtedy, gdy każdy kolor pokrywa dokładnie $|E(G_3)|/3 = |V(G_3)|/2$ krawędzi. Z drugiej strony wiadomo, że kolorowanie krawędzi grafu G_3 odpowiada kolorowaniu wierzchołków grafu $L(G_3)$, czyli G. Zatem $\chi_=(G) \le 3$ wtedy i tylko wtedy, gdy $\chi'(G_3) \le 3$. Skoro ten drugi problem jest NP-zupełny, zaś weryfikacja sprawiedliwości 3-pokolorowania jest w NP, więc teza twierdzenia została udowodniona.

Skoro problem sprawiedliwego kolorowania wierzchołków jest NP-trudny dla grafów krawędziowych, to jest on NP-trudny w przypadku ogólnym. Co więcej, problem sprawiedliwego kolorowania krawędzi jest również NP-trudny. Zatem w praktyce oba

problemy nie mogą być rozwiązane algorytmem wielomianowym. Dlatego w następnym punkcie podajemy algorytmy przybliżone dla kolorowania sprawiedliwego.

4. Dwa algorytmy heurystyczne

W punkcie tym opiszemy dwa konkurencyjne algorytmy dla przybliżonego kolorowania sprawiedliwego dowolnego grafu G. Oba opierają się na algorytmie *SL* (ang. *Smallest Last*). Metoda ta, podana przez Matulę i in. [6], polega na tym, że porządkujemy wierzchołki $v_1,v_2,...,v_n$ tak, aby dla każdego $i = 1,...,n$ wierzchołek v_i miał minimalny stopień w podgrafie indukowanym przez $v_1,...,v_i$. Następnie kolorujemy je zachłannie w otrzymanym porządku. Algorytm *SL* można zrealizować w czasie $O(m+n)$.

Pierwszy algorytm, o nazwie *NAIWNY*, polega na zamianie kolorów wierzchołków pomalowanych kolorami występującymi najczęściej i najrzadziej. Jeśli takie działanie nie spowoduje usprawiedliwienia pokolorowania, to wprowadzamy nowy kolor.

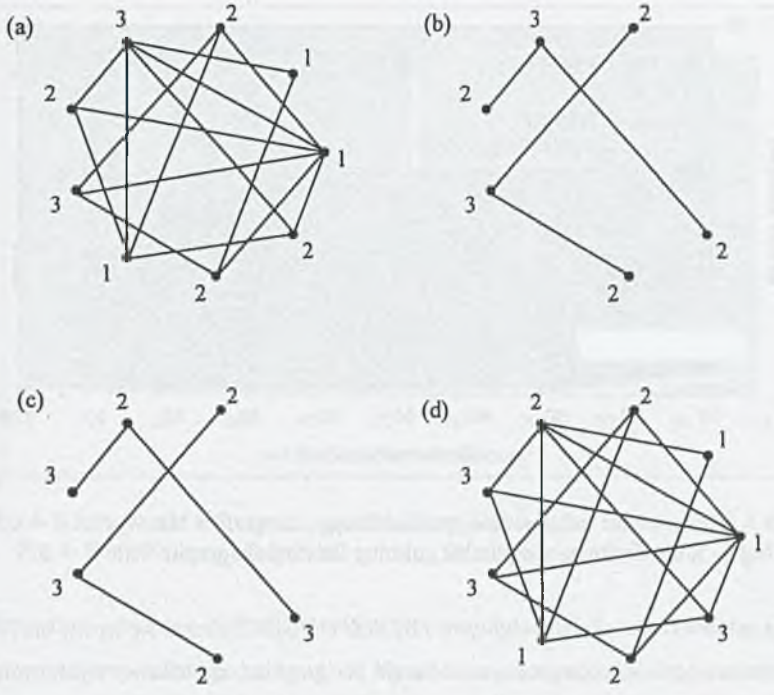
algorytm *NAIWNY*(*G*);
begin
  koloruj *G* algorytmem *SL*;
  while pokolorowanie niesprawiedliwe **do begin**
    colmin := numer koloru występującego najmniej razy;
    colmax := numer koloru występującego najwięcej razy;
    znajdź wierzchołki pomalowane tymi kolorami;
    if można zmienić kolor wierzchołka pomalowanego kolorem *colmax* na *colmin*
    then zrób to **else** dowolnemu wierzchołkowi z *colmax* daj nowy kolor
  end
end.

W praktyce próbując usprawiedliwić pokolorowanie sprawdzamy wszystkie możliwe pary *colmin* i *colmax*, a w ich ramach wszystkie wierzchołki z *colmax*. Dlatego pesymistyczna złożoność obliczeniowa algorytmu *NAIWNY* jest $O(n^4)$. Natomiast wynik działania algorytmu *NAIWNY* uzależniony jest od kolorowania, jakie uzyskamy po zastosowaniu algorytmu *SL*. Nie można jednak powiedzieć, że usprawiedliwienie będzie optymalne, jeśli tylko *SL* da pokolorowanie optymalne. Kolejny algorytm polega na zamianie kolorów w całych

podgrafach utworzonych z wierzchołków pomalowanych kolorami występującymi najmniej i najwięcej razy.

algorytm *TWORZPODGRAF*(*G*);
begin
min := 0; {zmienna, pod którą pamiętamy wartość *colmin*}
max := 0; {zmienna, pod którą pamiętamy wartość *colmax*}
p := 0; {wskaźnik tablicy COLZAM}
koloruj *G* algorytmem *SL*;
while pokolorowanie niesprawiedliwe **do begin**
colmin := numer koloru występującego najmniej razy;
colmax := numer koloru występującego najwięcej razy;
if ((*colmin*=*min*)**and**(*colmax*=*max*)) **or** ((*colmin*=*max*)**and**(*colmax*=*min*)) **then** *p* := *p*+1;
else begin
p := 1;
utwórz podgraf generowany wierzchołkami o kolorze *colmin* i *colmax*;
ponumeruj jego składowe spójności liczbami 1,...,*ilpodgr*;
dla każdej z nich oblicz różnicę pomiędzy krotnością występowania koloru *colmax* i *colmin* i ustaw je w tablicy COLZAM w kolejności malejących różnic;
end;
if (*p*≤*ilpodgr*) **and** (COLZAM[*p*]>0) **then** zamień kolory w *p*-tej składowej
else dowolnemu wierzchołkowi z *colmax* daj nowy kolor;
max := *colmax*; *min* := *colmin*
end
end.

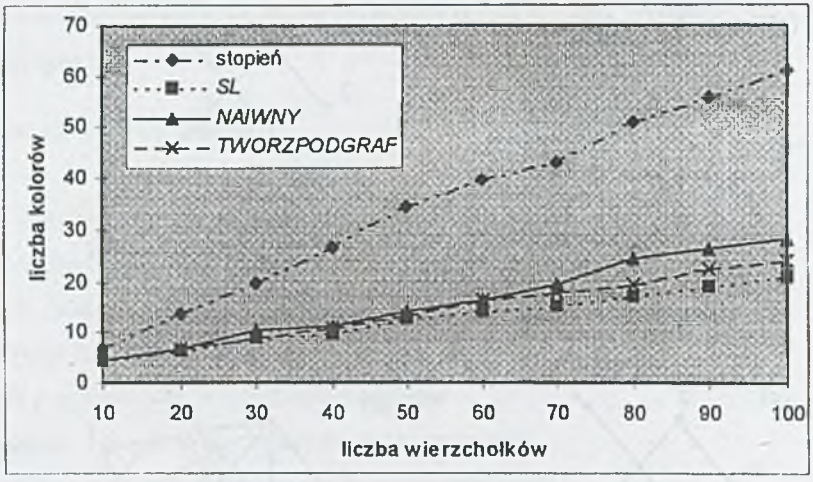
Algorytm *TWORZPODGRAF* ma także złożoność $O(n^4)$, choć ze znacznie większą stałą proporcjonalności. Przykład działania tego algorytmu podano na rys. 2. Zauważmy, że graf z rys. 2(a) nie może być optymalnie przemalowany algorytmem *NAIWNY*.



Rys.2. Przykład dla algorytmu *TWORZPODGRAF*: (a) graf G; (b) podgrafy 2-chromatyczne; (c) przekolorowanie; (d) kolorowanie sprawiedliwe
Fig.2. Example for algorithm *TWORZPODGRAF*: (a) graf G; (b) bichromatic subgraphs; (c) recoloring; (d) equitable coloring

5. Doświadczenia komputerowe

Dla oceny zachowania się obu algorytmów w przypadku średnim przeprowadzona została seria testów komputerowych polegająca na kolorowaniu grafów losowych o różnych współczynnikach gęstości. Rząd grafów wahał się od $n = 5$ co 5 do $n = 100$. Aby zredukować zależność uzyskanych rezultatów od wyników losowego wyboru grafów, dla każdego testowanego rozmiaru grafu wygenerowano 5 różnych grafów losowych. Grafy o jednakowym rozmiarze były kolorowane oboma algorytmami, a uzyskane wyniki uśredniano. Jednocześnie rejestrowano maksymalne stopnie generowanych grafów w celu weryfikacji hipotezy *ECC*. Doświadczenia objęły dwie serie grafów losowych o gęstości d wynoszącej 0.1 i 0.5, gdzie $d = 2m/(n(n-1))$. Eksperymenty obliczeniowe przeprowadzono na komputerze osobistym Pentium 120. Wyniki tych eksperymentów przedstawiamy na rys. 3 i 4.

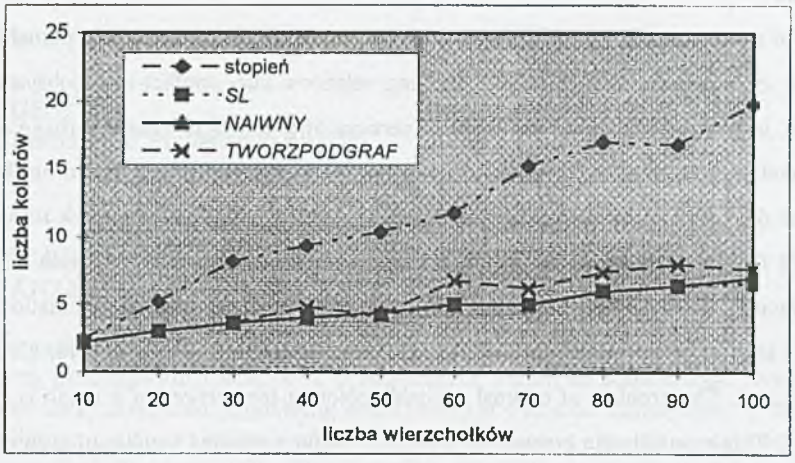


Rys.3. Efektywność kolorowania sprawiedliwego dla grafów losowych z $d = 0.5$
Fig.3. The efficiency of equitable coloring for random graphs with $d = 0.5$

Dla grafów o gęstości 0.5 algorytm *TWORZPODGRAF* okazał się lepszy niż *NAIWNY*, lecz kosztem znacznie dłuższego czasu obliczeń. Na przykład dla losowo wybranych grafów 100-wierzchołkowych czas działania wynosił średnio 4:46,41 wobec 0,01 sekundy dla algorytmu *NAIWNEGO*. W tym przypadku algorytm użył średnio o 4,2 kolorów mniej. Wśród badanych grafów były i takie, dla których różnica ilości użytych kolorów sięgnęła 10. Nie jest jednak prawdziwe ogólne stwierdzenie, że algorytm *TWORZPODGRAF* zawsze daje lepsze wyniki. W 18% przypadków algorytm *NAIWNY* dał lepsze wyniki, a w 35% oba algorytmy użyły tej samej liczby kolorów. Algorytm *TWORZPODGRAF* w 44 przypadkach na 100 usprawiedliwiał kolorowanie uzyskane algorytmem *SL*, nie dodając nowych kolorów. *NAIWNY* potrafił to zrobić jedynie w 32% przypadków.

Dla grafów rzadkich o gęstości 0.1 skuteczność omawianych algorytmów była inna. Dla tych grafów bardziej efektywny okazał się algorytm *NAIWNY*. Aż w 88% badanych grafów potrafił usprawiedliwić kolorowanie *SL* nie dodając nowych kolorów (*TWORZPODGRAF* potrafił to zrobić w 66%). Tylko w 8 przypadkach *TWORZPODGRAF* okazał się lepszy, zaś w 64% oba algorytmy użyły tej samej liczby kolorów.

W obu przypadkach, tzn. 0.1 i 0.5, liczba użytych kolorów była z reguły dużo mniejsza od Δ. Co więcej, w przypadku grafów rzadkich tylko 2 pokolorowania *SL* były sprawiedliwe. Wśród grafów gęstych było 7% takich od razu sprawiedliwych pokolorowań.



Rys.4. Efektywność kolorowania sprawiedliwego dla grafów losowych z $d = 0.1$
Fig.4. The efficiency of equitable coloring for random graphs with $d = 0.1$

Powyższe rezultaty sugerują następujący sposób postępowania w celu znalezienia pokolorowania sprawiedliwego: z uwagi na błyskawiczny czas działania algorytmu *NAIWNY* (na niewielkich grafach) najpierw należy użyć tego algorytmu; gdy liczba kolorów nie jest satysfakcjonująca, to należy zastosować algorytm *TWORZPODGRAF*.

LITERATURA

1. Chen B.L., Lih K.W.: Equitable coloring of trees. Journal of Combinatorial Theory, Ser. B, vol. 61, 1994, pp. 83-87.
2. Chen B.L., Lih K.W., Wu P.L.: Equitable coloring and the maximum degree. European Journal of Combinatorics, vol. 15, 1994, pp. 443-447.
3. Hajnal A., Szemeredi E.: Proof of a conjecture of Erdős. in: Combinatorial Theory and Its Applications II, North-Holland, Amsterdam, Colloq. Math. Soc. János Bolyai, vol. 4, 1970, pp. 601-623.
4. Holyer I.: The NP-completeness of edge-coloring. SIAM Journal on Computing, vol. 10, 1981, pp. 718-720.
5. Lih K.W., Wu P.L.: On equitable coloring of bipartite graphs. Discrete Mathematics, vol. 151, 1996, pp. 155-160.
6. Matula D.W., Marble D., Isaacson D.: Graph coloring algorithms. in: *Graph Theory and Computing*, Academic Press, New York, 1972, pp. 109-122.
7. Meyer W.: Equitable coloring. American Mathematical Monthly, vol. 80, 1973, pp. 920-922.

Recenzent: Prof.dr hab.inż. Jerzy Klamka

Abstract

In some discrete industrial systems we can encounter the problem of optimal equitable partition of a system with binary conflicting relations into conflict-free subsystems. For example, in the garbage collection problem vertices of graph G represent garbage collection routes and two such vertices are joined when the corresponding routes should not be run on the same day. The problem of assigning one of the six days of the work week to each route becomes that of 6-coloring of G. On practical grounds it might be desirable to have an approximately equal number of routes run on each day. We see that such situations can be modeled as an equitable graph coloring, i.e. such a coloring that the sizes of color classes differ by at most 1. The problem of optimal equitable coloring the vertices of a graph is generally NP-hard. This means that in practice we have to look for simplified families of graphs allowing polynomial-time algorithms or we have to use general-purpose algorithms for approximate graph coloring.

Let $\chi_=(G)$ be the equitable chromatic number of graph G. In the paper we give families of highly-structured graphs, e.g. trees, wheels, hypercubes and cubic graphs, for which $\chi_=(G)$ can be found in polynomial time. On the other hand, we consider bounds on $\chi_=(G)$ and mention the *Equitable Coloring Conjecture* that $\chi_=(G) \leq \Delta$ if G is neither a complete graph nor an odd cycle. Also, we give two approximation polynomial-time algorithms for equitable graph coloring. General considerations are supported by computational experience gained with implementation and profiling of these algorithms on identical series of pseudorandom graphs.