ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2002
Seria: AUTOMATYKA z. 136 Nr kol. 1556

Marek KUBALE, Dariusz DERENIOWSKI
Politechnika Gdańska

UPORZĄDKOWANE KOLOROWANIE WIERZCHOŁKÓW GRAFÓW

Streszczenie. W pracy przedstawiamy stosunkowo nowy model kolorowania grafów, mianowicie kolorowanie uporządkowane. Po scharakteryzowaniu potencjalnych zastosowań tego modelu przedstawiamy liniowy algorytm kolorowania grafów w sposób przybliżony. Pokazujemy klasy grafów, które ten algorytm koloruje optymalnie i klasy grafów, dla których błąd pokolorowania może być dowolnie duży. Przedstawiamy również doświadczenia komputerowe zebrane w trakcie jego implementacji i testowania na grafach losowych.

ORDERED COLORING OF GRAPH VERTICES

Summary. We present a relatively new model of graph coloring, namely ordered (rank) coloring. After characterizing potential applications of this model, we give a linear-time algorithm KU for approximate graph coloring. We show graph classes that our algorithm colors optimally and graph classes that can be colored arbitrarily bad. Finally, we give results of computational experiments gained while testing algorithm KU on random graphs.

1. Wprowadzenie

Istnieje kilkanaście modeli kolorowania grafów mających zastosowania praktyczne. Jednym z nich jest tytułowy problem kolorowania uporządkowanego (ang. *rank/ordered coloring*). Funkcja $c : V \to \{1, 2, \ldots, k\}$ jest *uporządkowanym* k-pokolorowaniem grafu $G = (V, E)$, jeśli dla każdych dwóch wierzchołków o tym samym kolorze każda ścieżka pomiędzy nimi zawiera wierzchołek o kolorze wyższym. *Uporządkowana liczba chromatyczna* $\chi_u(G)$ to najmniejsza liczba k, dla której graf G ma uporządkowane k-kolorowanie.

Problem uporządkowanego kolorowania grafów znajduje zastosowanie w projektowaniu układów VLSI [10] i rozwiązywaniu układów równań liniowych metodą faktoryzacji [6]. Dalej przybliżamy to drugie zastosowanie.

Dla danej $n \times n$-wymiarowej, symetrycznej macierzy A obliczana jest macierz L taka, że $A = LL^T$. Uporządkowane pokolorowanie grafu skonstruowanego na podstawie A pozwala na równoległe obliczanie kolumn macierzy L. Rozwiązywanie liniowego układu równań postaci $Ax = b$ można zastąpić rozwiązaniem dwóch układów $L^T x = a$ i $La = b$. Teorię tę można wykorzystać w statyce konstrukcji. Wskutek odkształcalności ciała każdy jego punkt przemieszcza się pod wpływem przyłożonego obciążenia. Rozwiązanie zadania równowagi polega na wyznaczeniu tych przemieszczeń. Zadanie uprości się znacznie, jeżeli ograniczymy się do wyznaczenia przemieszczeń tylko skończonej liczby punktów zwanych węzłami. Przemieszczenia pozostałych punktów oblicza się na podstawie interpolacji. Znając macierz A sztywności elementu oraz siły P obciążające węzły można przystąpić do obliczenia układu równań $P = AU$, w którym niewiadomy wektor U opisuje przemieszczenia węzłów [12, 13].

Uporządkowane kolorowanie grafów jest problemem NP-trudnym już w przypadku grafów dwudzielnych [1], chociaż ma swoje nietrywialne zawężenia wielomianowe. Co ciekawe, problem pozostaje wielomianowy w przypadku grafów łuków (ang. *circular-arc graphs*), mimo że klasyczne kolorowanie takich grafów jest NP-trudne [3].

W niniejszej pracy przedstawimy algorytm przybliżony o złożoności $O(m + n)$ do uporządkowanego kolorowania dowolnych grafów, gdzie m jest liczbą krawędzi, a n jest liczbą wierzchołków grafu. Podamy jego funkcję dobroci i najmniejsze trudne do kolorowania grafy. Przedstawimy również jego działanie w przypadku przeciętnym na podstawie doświadczeń komputerowych zebranych w trakcie jego implementacji i testowania na grafach losowych.

2. Algorytm KU

Opiszemy algorytm KU, którego nazwa pochodzi od słów *kolorowanie uporządkowane*. Algorytm ten koloruje wierzchołki grafu w sposób uporządkowany, nie gwarantując jednak użycia minimalnej możliwej liczby kolorów.

```
procedure KU(G);
begin
  wstaw wierzchołki na listę L;
```

```
posortuj niemalejąco wg stopni elementy listy L;
zainicjuj puste zbiory S oraz Q;
while L niepusta do begin
  while (L niepusta) and (stopień(L[1]) ≤ 1) do begin
    v := UsuńPierwszy(L);
    S := S ∪ {v};
  end;
  if L niepusta then begin
    v := UsuńOstatni(L);
    Q := Q ∪ {v};
  end;
end;
Pokoloruj graf indukowany przez wierzchołki z S algorytmem kolorującym drzewa;
while Q niepusty do begin
  v := UsuńZeZbioru(Q);
  c(v) := nowy kolor;
end;
end;
```

2.1. Objaśnienia

Lista L jest używana w powyższym algorytmie wyłącznie do celów optymalizacyjnych – gwarantuje znalezienie wierzchołka o maksymalnym (oraz minimalnym) stopniu w stałym czasie. Instrukcja v:=UsuńOstatni(L) oznacza wyszukanie wierzchołka o maksymalnym stopniu w grafie indukowanym przez wierzchołki ze zbioru $V(G) - (S \cup Q)$, natomiast instrukcja v:=UsuńPierwszy(L) – minimalnym.

Główna pętla **while** w pseudokodzie powyżej dokonuje podziału elementów $V(G)$ na dwa zbiory: S oraz Q. Zbiór S zawiera wierzchołki, które indukują las, więc gdy lista L jest pusta (tzn. każdy wierzchołek w $V(G)$ został dodany do S lub Q), to są one kolorowane przy wykorzystaniu dowolnego optymalnego algorytmu kolorującego drzewa. W skład zbioru Q natomiast wchodzą wierzchołki, które ostatecznie otrzymują dodatkowe

etykiety, tzn. parami różne oraz nieużyte wcześniej, tj.

$$c(v) > max\{c(u) : u \in S\}, i\ c(v) \neq c(v'), v, v' \in Q, v \neq v',$$

dlatego kolejność kolorowania grafów indukowanych przez S i Q jest istotna.

Funkcja stopień(v) zwraca wierzchołek sąsiedni do v w grafie indukowanym przez zbiór $V(G) - (S \cup Q)$. Podczas wykonywania operacji $S := S \cup \{v\}$, $Q := Q \cup \{v\}$ należy zadbać o to, aby lista L była nadal posortowana względem stopni wierzchołków w tym grafie.

2.2. Poprawność i efektywność algorytmu

Lemat 1. *Graf indukowany przez wierzchołki ze zbioru S nie zawiera cykli.*

Dowód. Przez $S_i \in S$ oznaczmy podzbiór pierwszych i wierzchołków przeniesionych do zbioru S w wyniku działania omawianego algorytmu. Udowodnimy przez indukcję, że

$$\forall_{i=1,\ldots,|S|}\, G[S_i] \text{ nie zawiera cykli.}$$

Dla $i = 1$ własność jest prawdziwa, więc załóżmy, że zachodzi również dla pewnego $i > 1$. W $(i+1)$-szej operacji przenosimy wierzchołek v do zbioru S. Załóżmy, że lemat nie zachodzi, tzn. w grafie S_{i+1} powstał cykl złożony z wierzchołków $v, u_1, \ldots, u_k$. Aby przenieść dowolny wierzchołek p do S, musi być spełniony warunek $\deg(p) \leq 1$ (w grafie $G[V(G) - (S \cup Q)]$), gdzie $\deg(v)$ oznacza stopień wierzchołka v. Wynika stąd, że operacja przenoszenia dowolnego spośród wierzchołków $v_1, \ldots, v_k$ powiększała zbiór $E(G[S_{i+1}])$ co najwyżej o jeden. Zatem graf rozpięty na wierzchołkach $v, u_1, \ldots, u_k$ zawiera najwyżej k krawędzi, co prowadzi do sprzeczności z założeniem, że tworzą one cykl. □

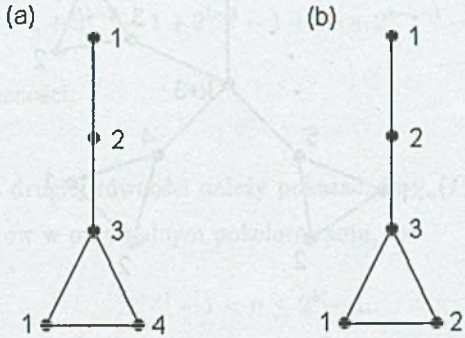
Z powyższego lematu wynika, że próba kolorowania grafu $G[S]$ algorytmem kolorującym drzewa doprowadzi do poprawnego wyniku. Przydzielenie nowych kolorów dla elementów Q nie prowadzi do pokolorowania niepoprawnego.

Czas potrzebny na wykonanie części inicjalizacyjnej jest zdominawany przez operację sortowania, którą można przeprowadzić w czasie $O(m+n)$. Czynność usuwania krawędzi z grafu wymaga uaktualnienia porządku wierzchołków na liście L, co można zrealizować w stałym czasie. Przeniesienie wierzchołka do zbiorów Q lub S wymaga więc czasu proporcjonalnego do jego stopnia (w zredukowanym grafie G). Pierwsza pętla **while** wykona więc $O(m+n)$ operacji. Pokolorowanie $G[S]$ może odbyć sie w czasie liniowym. Dlatego złożoność calego algorytmu można oszacować przez $O(m+n)$.

3. Najmniejsze grafy trudne dla KU

Poniżej zostały przedstawione przykłady grafów, dla których opisany algorytm nie generuje optymalnego pokolorowania.

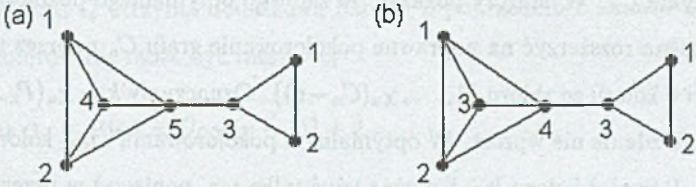
Piewszym przykładem jest graf dość trudny do kolorowania, dla którego algorytm może zużyć $\chi_u(G)+1$ kolorów. Taki przypadek pokazuje rysunek 1(a), rysunek 1(b) natomiast przedstawia pokolorowanie optymalne.



Rys. 1. Najmniejszy graf „dość trudny" do kolorowania dla KU: (a) pokolorowanie suboptymalne, (b) pokolorowanie optymalne

Fig. 1. The smallest SHC graph for algorithm KU: (a) suboptimal coloring, (b) optimal coloring

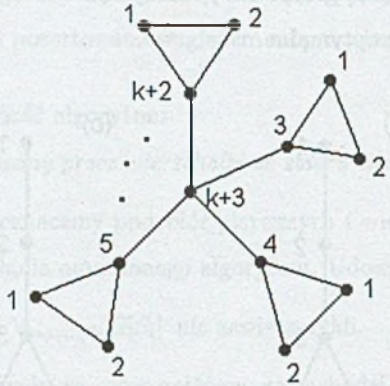
Kolejny rysunek przedstawia graf G_2, dla którego algorytm zawsze zwróci pokolorowanie nieoptymalne. Dzieje się tak dlatego, że należy usunąć 3 wierzchołki w celu wyeliminowania wszystkich cykli, co daje pokolorowanie 5 barwami, natomiast $\chi_u(G_2)=4$.



Rys. 2. Graf „trudny" do kolorowania dla KU: (a) pokolorowanie suboptymalne, (b) pokolorowanie optymalne

Fig. 2. HC graph for algorithm KU: (a) suboptimal coloring, (b) optimal coloring

Ostatni przykład dowodzi, że funkcja dobroci dla rozważanego algorytmu jest liniowa, ponieważ KU koloruje graf, pokazany na rys. 3, $k+3$ barwami dla każdego $k > 3$, natomiast liczba uporządkowana wynosi 4. Zatem $\mathrm{KU}(G_k) = k+3$, $\chi_u(G_k) = 4$, a ponieważ liczba wierzchołków $n(G_k) = 3k+1$, więc funkcja dobroci $\mathrm{KU}(n) = \Theta(n)$.



Rys. 3. Graf G_k, dla którego $\chi_u(G_k) = 4$
Fig. 3. Graph G_k for which $\chi_u(G_k) = 4$

4. Klasy grafów kolorowane optymalnie przez algorytm KU

Lemat 2. $\chi_u(C_n) = \chi_u(P_{n-1}) + 1 = \lceil log_2 n \rceil + 1$

Dowód. Korzystając z zasady indukcji matematycznej można wykazać, że k kolorami można pokolorować ścieżkę o długości co najwyżej $2^k - 1$. Niech v będzie dowolnym wierzchołkiem w cyklu C_n. Wystarczy pokazać, że żadnego optymalnego pokolorowania grafu $C_n - v$ nie można rozszerzyć na poprawne pokolorowanie grafu C_n poprzez przydzielenie wierzchołkowi v koloru ze zbioru $\{1, \ldots, \chi_u(C_n - v)\}$. Oznaczymy $k = \chi_u(P_{n-1})$ i dowodzimy poprzednie zdanie nie wprost. W optymalnym pokolorowaniu P_{n-1} kolor k jest użyty jednokrotnie. Również koloru $k-1$ można użyć tylko raz, ponieważ w przeciwnym przypadku wierzchołek v musiałby otrzymać barwę wiekszą niż k. Kolory k i $k-1$ zą zawsze zabronione dla wierzchołka v, ale użycie $k-2$ byłoby możliwe, o ile ten kolor zostałby użyty do pokolorowania P_{n-1} tylko raz i występowałby pomiędzy etykietami k oraz $k-1$.

Korzystając ze stwierdzenia na początku tego dowodu można obliczyć długość ścieżki:

$$n - 1 \leq 2(2^{k-3} - 1) + 2 + 2^{k-2} - 1 = 2^{k-1} - 1,$$

a ścieżkę o takiej długości można pokolorować za pomocą $k - 1$ barw, co oznacza, że skonstruowane pokolorowanie nie jest optymalne. Należy jeszcze zauważyć, że próba utworzenia pokolorowania, w którym wierzchołkowi v będzie można przydzielić etykietę $k - s, s > 2$, tworzy ścieżkę o długości:

$$n - 1 = 2^{k-s-1} - 1 + 2^{k-2} - 1 + 2^{k-3} - 1 + \cdots + 2^{k-s-1} - 1 + s = 2^{k-1} - 1,$$

co prowadzi do sprzeczności.

W celu wykazania drugiej równości należy pokazać, że $\chi_u(P_n) = \lceil log_2(n+1) \rceil$. Jeśli k oznacza liczbę kolorów w optymalnym pokolorowaniu, to

$$2^{k-1} - 1 < n \leq 2^k - 1.$$

Po wykonaniu przekształceń otrzymujemy:

$$log_2(n+1) \leq k < log_2(n+1) + 1$$

$$k = \lceil log_2(n+1) \rceil. \ \square$$

Twierdzenie 1. *Algorytm KU koloruje optymalnie cykle.*

Dowód. Algorytm KU, otrzymując na wejściu cykl, umieści jeden wierzchołek w zbiorze Q, a następnie tak powstały podgraf (ścieżka) pokoloruje optymalnie i ostatecznie wierzchołek $v \in Q$ otrzyma dodatkową barwę. Z poprzedniego lematu wynika, że liczba użytych kolorów nie może być niższa. $\square$

Lemat 3. $\chi_u(W_n) = \lceil log_2(n-1) \rceil + 2$

Dowód. Wierzchołek o stopniu $n-1$, który oznaczymy przez v, w każdym pokolorowaniu otrzymuje dodatkowy kolor. Zatem optymalne pokolorowanie koła W_n otrzymujemy kolorując optymalnie cykl $W_n - v$, co wymaga użycia $\lceil log_2(n-1) \rceil + 1$ barw. $\square$

Twierdzenie 2. *Algorytm KU koloruje optymalnie koła.*

Dowód. Zbiór Q po zakończeniu obliczeń w głównej pętli będzie zawierał dwa elementy: wierzchołek o stopniu $n-1$ oraz dowolny spośród pozostałych. Powyższe dwa wierzchołki otrzymają dodatkowe etykiety i jest to rozwiązanie optymalne, co wynika z wcześniejszego lematu. □

Lemat 4. *Niech $G = K_{s_1,\ldots,s_k}$ będzie grafem pełnym k-dzielnym. Wówczas $\chi_u(G) = n - max\{s_1,\ldots,s_k\} + 1$.*

Dowód. Teza wynika stąd, że jeśli w jednym z podzbiorów zawierających wierzchołki niesąsiednie w G dwa wierzchołki mają przydzielony jednakowy kolor, to wszystkie wierzchołki z nimi sąsiednie muszą mieć etykiety wyższe i parami różne. □

Twierdzenie 3. *Algorytm KU koloruje optymalnie grafy pełne k-dzielne.*

Dowód. W grafie $K_{s_1,\ldots s_k}$ jest s_i wierzchołków (oznaczmy ich zbiór przez D) o maksymalnym stopniu, gdzie $s_i \leq s_j$ dla $j = 1,\ldots,k$. Algorytm w pierwszym kroku zdecyduje przenieść wierzchołek ze zbioru D do zbioru Q, co spowoduje zmniejszenie stopni wierzchołków, które nie należą do D, nie zmieniając jednocześnie stopni tych z D. Po s_i iteracjach głównej pętli graf wejściowy zostanie zredukowany do $K_{s_1,\ldots,s_{i-1},s_{i+1},\ldots,s_k}$. Algorytm będzie przenosić kolejne wierzchołki do zbioru Q, aż rozpatrywany graf zredukuje się do $K_{1,max\{s_1,\ldots,s_k\}}$. Wówczas liście otrzymają najniższy kolor. Dla wierzchołka centralnego należy wprowadzić nowy kolor, po czym wszystkie pozostałe (usunięte wcześniej) wierzchołki otrzymają oddzielne etykiety, co jest rozwiązaniem optymalnym na mocy poprzedniego lematu. □

Twierdzenie 4. *Algorytm KU koloruje optymalnie drzewa oraz grafy pełne.*

Dowód. KU otrzymując na wejściu drzewo umieści wszystkie wierzchołki w zbiorze S, które następnie zostaną zaetykietowane optymalnie w podprogramie kolorującym drzewa. Z poprawności algorytmu wynika, że koloruje optymalnie grafy pełne, ponieważ $\chi_u(K_n) = n$. □

5. Doświadczenia komputerowe

W niniejszym punkcie podajemy wyniki doświadczeń komputerowych zebranych podczas testowania algorytmu KU. Mierzonymi wielkościami były: liczba zużywanych kolorów oraz czas obliczeń. Pojedyncza próba polegała na wygenerowaniu 10 grafów losowych, które podano na wejście algorytmu, a następnie uśredniano otrzymane na wyjściu wyniki. Obliczone liczby kolorów zostały zaokrąglone do liczb całkowitych. Tabela 1 zawiera tak obliczone wartości dla różnych rozmiarów $n = 100, 200, \ldots, 1000$ oraz gęstości grafów $g = 0.05, 0.1, 0.125, 0.25, 0.375, 0.5$. Litera l oznacza liczbę kolorów użytych przez algorytm KU.

Do optymalnego kolorowania drzew użyto algorytmu opisanego w pracy [2], o złożoności $O(nlogn)$.

Obliczenia były wykonywane na komputerze wyposażonym w procesor Intel Celeron 700 MHz. Algorytm zakodowano w języku C++. Strukturą danych przechowującą grafy w pamięci była macierz sąsiedztwa, co implikuje kwadratową złożoność algorytmu.

Tabela 1

Wyniki doświadczeń komputerowych

	gęstość g											
	0.05		0.1		0.125		0.25		0.375		0.5	
n	l	czas	l	czas	l	czas	l	czas	l	czas	l	czas
100	36	0.001	57	0.002	63	0.002	80	0.003	87	0.005	91	0.005
200	108	0.006	145	0.008	154	0.010	175	0.015	184	0.028	189	0.045
300	192	0.015	237	0.020	247	0.023	273	0.048	283	0.091	288	0.154
400	281	0.027	331	0.036	344	0.047	370	0.111	382	0.223	388	0.390
500	371	0.044	427	0.065	441	0.080	471	0.213	482	0.448	487	0.782
600	465	0.065	523	0.103	539	0.132	570	0.374	581	0.796	587	1.383
700	559	0.091	623	0.156	637	0.204	669	0.599	681	1.280	687	2.199
800	654	0.122	720	0.217	736	0.289	768	0.891	781	1.898	787	3.270
900	751	0.163	818	0.304	834	0.407	869	1.238	880	2.702	887	4.680
1000	848	0.207	917	0.399	934	0.544	968	1.734	979	3.693	987	6.368

6. Uwagi końcowe

Tabela 2 zawiera informacje na temat uporządkowanego kolorowania różnych klas grafów, w szczególności oszacowania liczby uporządkowanej $\chi_u(G)$ oraz złożoności dokładnych algorytmów kolorowania, o ile są znane. Dane zostały zebrane na podstawie informacji dostępnych w literaturze oraz w niniejszej pracy.

Na zakończenie odnotujemy, że problem uporządkowanego kolorowania krawędzi jest również NP-trudny [11] i pozostaje takim nawet dla grafów dwudzielnych.

Tabela 2

Złożoność obliczeniowa kolorowania uporządkowanego

Typy grafów	złożoność	$\chi_u(G)$	Literatura
dwudzielne	NPC	?	[1]
dopełnienia dwudzielnych	NPC	?	[1]
krawędziowe	NPC	?	[11]
planarne	?	$c\sqrt{n} \leq \chi_u(G) \leq 3(\sqrt{6}+2)\sqrt{n}$	[2]
P_n	$O(n)$	$\lceil \log_2(n+1) \rceil$	
drzewa	$O(n)$	$\chi_u(T) \leq 1 + \lfloor \log_2 n \rfloor$	[2, 4]
C_n	$O(n)$	$\lceil \log_2 n \rceil + 1$	
W_n	$O(n)$	$\lceil \log_2(n-1) \rceil + 2$	
$K_{s_1,\ldots,s_k}$	$O(n+m)$	$n - max\{s_1,\ldots,s_k\} + 1$	
"starlike"	$O(n+m)$	?	[9]
grafy d-trapezoidalne	$O(n^3\chi_u(G)^{3d+3})$	?	[3]
grafy przedziałowe	$O(n^3)$	?	[3]
grafy łuków	$O(n^3)$	?	[3]
permutacyjne	$O(n^6)$	?	[8]
drzewa - kol. krawędzi	$O(m)$	?	[7]

LITERATURA

1. Bodlaender H., Deogun J.S., Jansen K., Kloks T., Kratsch D., Müller H., Tuza Z.: Rankings of graphs, *SIAM J.Discrete Math.* 11 (1998) 168-181.

2. Katchalski M., McCaugh W., Seager S.: Ordered colourings, *Discrete Math.* 142 (1995) 141-154.
3. Deogun J.S., Kloks T., Kratsch D., Müller H.: On the vertex ranking problem for trapezoid, circular-arc and other graphs, *Discrete Appl. Math.* 98 (1999) 39-63.
4. Schäffer A.A.: Optimal node ranking of trees in linear time, *Infor. Process. Lett.* 33 (1989/90) 91-96.
5. Manne F.: Reducing the height of an elimination tree through local reorderings. *Tech. Report* 51, University of Bergen, Norway, 1991.
6. Liu J.W.H.: The role of elimination trees in sparse factorization, *SIAM J.Matrix Analysis and Appl.* 11 (1990) 134-172.
7. Tak Wah Lam, Fung Ling Yue, Optimal Edge Ranking of Trees in Linear Time, *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, (1998) 436-445.
8. Deogun J.S., Kloks T., Kratsch D., Müller H.: On vertex ranking for permutation and other graphs, Proc. of the 11th Annual Symposium on Theoretical Aspects of Computer Science, *Lecture Notes in Computer Science* 755 (1994) 747-758.
9. Sun-yuan Hsieh, On vertex ranking of a starlike graph, *Inform. Process. Lett.* 82 (2002) 131-135.
10. Leiserson C.E.: Area-efficient graph layout (for VLSI) *Proc. 21st Ann. IEEE Symp. on Foundations of Computer Science* (1980) 270-281.
11. Lam T.W., Yue F.L.: Edge ranking of graphs is hard, *Discrete Appl. Math.* 85 (1998) 71-86.
12. Szmelter J., Dacko M., Dobrociński S., Wieczorek M.: Metoda elementów skończonych w statyce konstrukcji, Arkady, Warszawa 1979.
13. Waszczyszyn Z., Cichoń Cz., Radwańska M.: Metoda elementów skończonych w stateczności konstrukcji, Arkady, Warszawa 1990.

Recenzent: Prof. dr hab. inż. Andrzej Świerniak

Abstract

There are about a dozen of graph coloring models which have practical applications. One of them is an ordered (rank) coloring of the vertices of G. A legal coloring of vertices is called ordered if any path joining the vertices of the same color goes via a vertex of higher color. The problem of ordered graph coloring can be applied in designing VLSI circuits and in computing Cholesky factorizations of matrices in parallel.

This problem is NP-hard even if the input graph is bipartite, though it has numerous easy instances. It is interesting that the problem remains polynomially solvable in the case of circular-arc graphs (the classical coloring of such graphs is intractable). In the paper we present a linear-time algorithm for approximate coloring of general graphs. We give graph classes that our algorithm colors optimally and graph classes that can be colored arbitrarily bad. Finally, we give results of computational experiments gained while testing our algorithm on random graphs.