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

Lukasz KUSZNER
Politechnika Gdańska

ROZPROSZONE KOLOROWANIE GRAFÓW

Streszczenie. W pracy zaprezentowano nowy rozproszony algorytm kolorowania grafów. Przeprowadzone eksperymenty pokazują, że daje on lepsze wyniki niż znany wcześniej algorytm trywialny.

DISTRIBUTED GRAPH COLORING

Summary. In the paper we present a new distributed algorithm for graph coloring. A practical simulation shows that the algorithm performs much better then a naive distributed algorithm.

1. Wstęp

Grafem G nazwiemy parę $G = (V, E)$, w której V jest skończonym zbiorem *wierzchołków*, a E rodziną *krawędzi*. Rodzina E składa się z nieuporządkowanych par $e = \{u, v\} \in E$, gdzie $u, v \in V$. O wierzchołkach u, v takich, że $\{u, v\} \in E$ powiemy, że są *sąsiednie*, a o wierzchołku u i krawędzi $\{u, v\}$, że są *incydentne*. Przez *stopień* wierzchołka v rozumiemy liczbę krawędzi z nim incydentnych i oznaczamy $deg(v)$. Największy stopień wierzchołka w grafie oznaczamy przez Δ.

Przyporządkowanie kolorów lub elementów innego zbioru wierzchołkom grafu G nazwiemy *kolorowaniem*, jeśli sąsiednim wierzchołkom przydzielono różne kolory. Kolorowanie przy użyciu k kolorów nazwiemy *k-kolorowaniem*. Jeśli dla grafu istnieje k-kolorowanie, to powiemy, że jest on *k-kolorowalny*, a jeśli dodatkowo nie jest on $(k-1)$-kolorowalny, to powiemy, że jego *liczba chromatyczna* wynosi k, co oznaczamy $\chi(G) = k$. Znamy wiele algorytmów kolorowania. Można je znaleźć w popularnych książkach, np. w [7].

Przez *graf losowy* $G_{n,p}$ rozumiemy graf o liczbie wierzchołków n, który powstał przez połączenie każdej pary wierzchołków krawędzią z prawdopodobieństwem p; losowania odbywają się w sposób niezależny. W ten sposób otrzymujemy rodzinę grafów nadającą się do testowania heurystycznych algorytmów grafowych. Były one używane między innymi w pracach [5], [9], [10]. Na grafach losowych można też badać zachowanie algorytmów w przypadku średnim.

2. Przyznawanie kodów w sieciach bezprzewodowych

Zajmiemy się teraz polem możliwych zastosowań dla kolorowania rozproszonego. Weźmy pod uwagę bezprzewodową sieć radiową. Zastosowanie sieci radiowych umożliwia [11]:

- komunikację pomiędzy stacjami będącymi w ruchu;
- wymianę informacji pomiędzy stacjami rozproszonymi na dużym obszarze niezależnie od sieci przewodowych;
- łatwą rozbudowę i rekonfigurację sieci w zależności od potrzeb.

Poszczególne stacje są wyposażone w nadajnik i odbiornik. Głównym zasobem sieci bezprzewodowej jest kanał komunikacyjny. Korzysta z niego wielu użytkowników sieci. Swobodne nadawanie pakietów danych prowadzi do nakładania się ich na siebie i powstawania kolizji. Zaistnienie kolizji powoduje utratę danych i konieczność retransmisji.

Kolizje można podzielić na: *bezpośrednie*, kiedy zakłócające stacje słyszą się lub *ukryte*, kiedy niesłyszące się stacje nadają do tego samego odbiorcy.

Konieczne jest zastosowanie protokołów umożliwiających współdzielenie pasma. W szczególności mamy do dyspozycji protokół wykorzystujący algorytm losowego przydziału zasobów: wielodostęp z podziałem kodowo-adresowym (*Code Division Multiple Access*).

W protokole CDMA możliwe jest wyeliminowanie kolizji dzięki użyciu całego spektrum technik oraz dzięki właściwemu przydziałowi *ortogonalnych kodów*. Zastosowanie kolorowania grafów do przydziału kodów dla protokołu CDMA zostało omówione w [1].

Sieć jest modelowana przez nieskierowany graf $G_{WN} = (V, E)$, w którym zbiór wierzchołków V oznacza stacje nadawczo-odbiorcze, a zbiór krawędzi kanały komunika-

cyjne pomiędzy parami stacji. Dokładniej dwa wierzchołki u, v są połączone krawędzią $(u, v) \in E$ wtedy i tylko wtedy, gdy odpowiadające im stacje słyszą się wzajemnie. W tym przypadku powiemy o wierzchołkach, że są *fizycznie sąsiednie*. Graf G_{WN} reprezentuje fizyczną topologię sieci. Wystąpienie kolizji bezpośredniej jest możliwe pomiędzy stacjami sąsiednimi w grafie G_{WN}. Kolizja ukryta może wystąpić między stacjami u i v, jeśli długość najkrótszej ścieżki pomiędzy nimi wynosi 2. Stacje u i v unikną kolizji używając różnych kodów. Przyporządkowując kodom kolory mamy problem kolorowania grafu w dwóch wersjach:

- Unikanie ukrytych kolizji

  Kolorowanie grafu $G_H = (V, E_H)$ stowarzyszonego z G_{WN}, w którym $(u, v) \in E_H$, jeśli najkrótsza ścieżka z u do v wynosi 2.

- Unikanie kolizji ukrytych i bezpośrednich

  Kolorowanie grafu $G_{HP} = (V, E_{HP})$ stowarzyszonego z G_{WN}, gdzie $(u, v) \in E_{HP}$, jeśli najkrótsza ścieżka z u do v nie przekracza 2.

W cytowanym wcześniej artykule [1] autorzy proponują do kolorowania tradycyjne algorytmy sekwencyjne. Co jednak w przypadku, gdy stacje przemieszczają się względem siebie i topologia sieci szybko się zmienia, albo jeśli sieć jest bardzo duża i gromadzenie danych o aktualnej konfiguracji nastręcza poważne trudności?

3. Kolorowanie rozproszone

W dalszym ciągu pracy nie będziemy koncentrować się na tym, czemu odpowiadają wierzchołki grafu. Nie będzie dla nas różnicy, czy mamy do czynienia ze stacjami nadawczo-odbiorczymi, czy też z czym innym. Przyjmiemy, że jest to coś, co potrafi liczyć, zapamiętywać dane i komunikować się z innymi podobnymi jednostkami. W dalszej części będziemy używali słowa *procesor*.

3.1. Model

Wygodnym narzędziem do analizy jest poniższy model systemu wieloprocesorowego. Sieć procesorów jest reprezentowana przez graf nieskierowany. Wierzchołki grafu

odpowiadają procesorom, a krawędzie dwukierunkowym, niezawodnym kanałom komunikacyjnym. Zwykle przez n oznacza się liczbę wierzchołków w grafie, czyli tutaj liczbę procesorów, przez m rozumiemy liczbę krawędzi – tutaj kanałów komunikacyjnych.

Obliczenia wykonywane są w rundach. Każda runda składa się z dwóch faz: fazy obliczeń i fazy komunikacji (wysyłania i odbierania komunikatów).

Przez *złożoność czasową* algorytmu rozumiemy wyłącznie liczbę rund potrzebnych do wykonania zadania. Jest to uzasadnione wysokimi nakładami na przesłanie komunikatu i stosunkowo niskimi na wykonanie obliczeń.

3.2. Oczekiwania

Po kolorowaniu rozproszonym nie należy się zbyt wiele spodziewać w stosunku do zwykłych algorytmów sekwencyjnych. W szczególności nie należy sądzić, że problem niewielomianowy da się rozwiązać wielomianowo w nowym modelu. Wynika to z następującego rozumowania: rozpatrzmy trudny problem niewielomianowy Π rozmiaru n. Gdyby można go było rozwiązać wielomianowo w środowisku rozproszonym na n procesorach, to również można byłoby go rozwiązać wielomianowo na jednym procesorze w czasie n razy dłuższym, czyli również wielomianowym.

Właściwie należy spodziewać się mniej niż po algorytmach równoległych. Proste przeniesienie algorytmów równoległych nie jest możliwe głównie z powodu braku pamięci współdzielonej. Z tego względu również przeniesienie algorytmów sekwencyjnych napotyka na poważne trudności.

4. Algorytm trywialny

Najprostszy algorytm to $(\Delta + 1)$-kolorowanie wierzchołkowe (*trivial algorithm*) [2], gdzie Δ jest największym stopniem wierzchołka w grafie. Takie pokolorowanie oczywiście istnieje, co wynika z następującego algorytmu sekwencyjnego: każdemu kolejnemu wierzchołkowi przypisz najmniejszy kolor nie przypisany jeszcze żadnemu sąsiadowi.

W wersji rozproszonej algorytmu każdy wierzchołek u przechowuje paletę barw nie przypisanych jeszcze przez żadnego z sąsiadów – inicjalnie rozmiaru $deg\,(u) + 1$. Każdy wierzchołek losuje kolor ze swojej palety i wysyła komunikat do swoich sąsiadów. Jeśli

żaden z nich nie wybrał tej barwy, u jest pokolorowany i informuje o tym swoich sąsiadów. Jeśli jakiś sąsiad wylosował taki sam kolor, u nie zatrzymuje go. Na początku następnej rundy u usuwa ze swojej palety kolory użyte przez jego sąsiadów. Johansson w [3] wykazał, że w algorytmie trywialnym każdy wierzchołek w każdej rundzie otrzyma kolor z prawdopodobieństwem $p \geq 1/4$. Stąd, wykorzystując wynik Karpa [6], zauważymy, że algorytm skończy się prawie zawsze po $O(\log n)$ rundach. Bardziej precyzyjnie: czas wykonania $T(n)$ dla n wierzchołkowego grafu dla każdego dodatniego t spełnia zależność:

$$\Pr\left\{T(n) \geq \lfloor \log_{4/3} n \rfloor + t + 1\right\} \leq \left(\frac{3}{4}\right)^{t-1} \tag{1}$$

Podobny algorytm został podany przez Luby'ego w [8]. W tym algorytmie dodatkowo w każdej rundzie wierzchołek podejmuje próbę kolorowania tylko z pewnym prawdopodobieństwem p.

5. Algorytm DLF

Opisany wcześniej algorytm trywialny ma istotną wadę, a mianowicie nie ma żadnego mechanizmu oszczędzania kolorów. Z tego względu zaproponujemy teraz inny rozproszony algorytm, który jest pod tym względem dużo lepszy.

Jak wiemy z analiz, opłaca się najpierw kolorować wierzchołki o większym stopniu. Wykorzystują to sekwencyjne algorytmy: *largest-first*(LF), *smallest-last*(SL), czy *saturation largest-first*(SLF). Spróbujmy zastosować tę obserwację w kolorowaniu rozproszonym.

Zakładamy model opisany we wstępie. Każdy wierzchołek zna porządek kolorów, listę swoich sąsiadów i potrafi wygenerować losową liczbę. Nie ma żadnych dodatkowych założeń o grafie, ani o jego liczbie chromatycznej.

Algorytm przebiega w rundach. W każdej rundzie są dwie fazy komunikacji. Pomiędzy nimi wierzchołki wykonują obliczenia.

Każdy wierzchołek v posiada trzy parametry:

- swój stopień: $deg(v)$,
- losowy parametr: $rndvalue(v)$,

- inicjalnie pustą paletę zabronionych kolorów, które zostały już zużyte przez jego sąsiadów: $usedcolors(v)$.

Parametry $deg(v)$ i $rndvalue(v)$ decydują o kolejności kolorowania. Niech v_1, v_2 będą wierzchołkami grafu. Mówimy, że v_1 ma *wyższy priorytet* niż v_2, jeśli:

$$deg(v_1) > deg(v_2)$$

lub

$$(deg(v_1) = deg(v_2)) \quad \wedge \quad (rndvalue(v_1) > rndvalue(v_2))$$

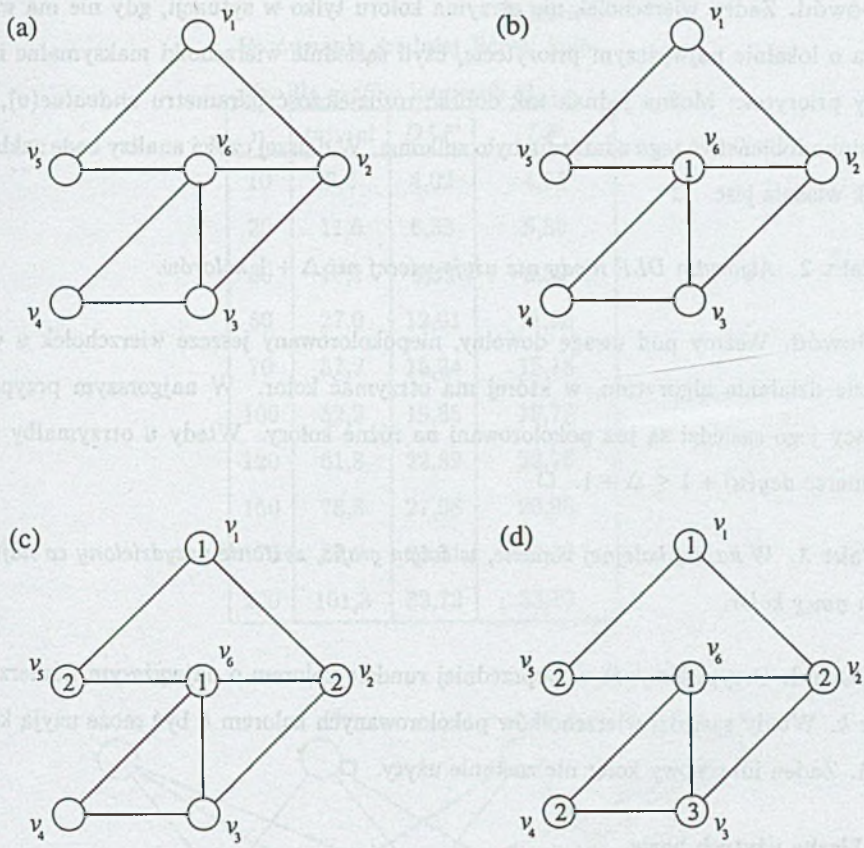
Pojedyncza runda dla każdego niepokolorowanego wierzchołka przebiega w pięciu krokach:

- Wylosuj parametr $rndvalue(v)$.
- Wyślij wszystkim sąsiadom swoje parametry: $deg(v)$, $rndvalue(v)$ i pierwszy kolor, którego nie ma na twojej liście $usedcolors$.
- Porównaj swoje parametry z otrzymanymi od sąsiadów i sprawdź, czy masz najwyższy priorytet spośród tych, którzy chcą otrzymać ten sam kolor.
- Jeśli tak, przydziel sobie kolor, poinformuj o tym sąsiadów i zakończ.
- Jeśli nie, zaktualizuj swoją listę $usedcolors$.

5.1. Przykład działania

Działanie algorytmu prześledzimy na przykładzie grafu znajdującego się na rys. 1(a). Największy stopień – 4 ma wierzchołek v_6. Zostanie on pomalowany w pierwszej kolejności. Równocześnie mógłby otrzymać kolor wierzchołek v_1 (tylko v_1 nie jest połączony z v_6), ale jego sąsiad v_2 ma większy stopień. Sytuacja po pierwszej rundzie jest przedstawiona na rys. 1(b).

Teraz v_1 jako jedyny ma jeszcze wolny kolor o numerze 1, bez przeszkód otrzyma go w rundzie drugiej. Pierwszym wolnym kolorem dla pozostałych wierzchołków (poza v_6, który już otrzymał kolor) jest barwa o numerze 2. Występują konflikty pomiędzy v_2 i v_3 oraz v_3 i v_4. Wierzchołek v_4 musi przegrać, gdyż $deg(v_4) < deg(v_3)$. Natomiast stopnie wierzchołków v_2 i v_3 są równe, w tym przypadku decyduje losowy parametr $rndvalue$. Jak widzimy na rys. 1(c), więcej szczęścia miał v_2 i ten właśnie wierzchołek został pomalowany.



Rys. 1. Przykład kolorowania algorytmem DLF

Fig. 1. Example of coloring with the DLF algorithm

Pozostały wierzchołki v_3 i v_4. Nie ma pomiędzy nimi konfliktu, gdyż v_4 ma jeszcze wolny kolor o numerze 2, a v_3 już nie. Konieczne jest użycie kolejnej barwy.

Na rysunku 1(d) widzimy ostateczny przydział barw po trzeciej rundzie działania algorytmu. W tym przypadku otrzymaliśmy optymalne pokolorowanie. Co najmniej trzy barwy musiały być użyte, bo graf zawiera trójkąty: v_3, v_4, v_6 oraz v_2, v_3, v_6.

5.2. Analiza

Fakt 1. *W każdej rundzie prawie zawsze co najmniej jeden wierzchołek zostanie pokolorowany*

Dowód. Żaden wierzchołek nie otrzyma koloru tylko w sytuacji, gdy nie ma wierzchołka o lokalnie najwyższym priorytecie, czyli sąsiednie wierzchołki maksymalne mają równy priorytet. Można jednak tak dobrać rozdzielczość parametru $rndvalue(v)$, aby prawdopodobieństwo tego zdarzenia było znikome. W dalszej części analizy będę zakładał, że tak właśnie jest. □

Fakt 2. *Algorytm DLF nigdy nie użyje więcej niż* $\Delta + 1$ *kolorów.*

Dowód. Weźmy pod uwagę dowolny, niepokolorowany jeszcze wierzcholek u w tej rundzie działania algorytmu, w której ma otrzymać kolor. W najgorszym przypadku wszyscy jego sąsiedzi są już pokolorowani na różne kolory. Wtedy u otrzymałby kolor o numerze $deg(u) + 1 \leq \Delta + 1$. □

Fakt 3. *W każdej kolejnej rundzie, w całym grafie, zostanie przydzielony co najwyżej jeden nowy kolor.*

Dowód. Przypuśćmy, że w poprzedniej rundzie kolorem o najwyższym numerze był kolor k. Wtedy sąsiedzi wierzchołków pokolorowanych kolorem k być może użyją koloru $k + 1$. Żaden inny nowy kolor nie zostanie użyty. □

5.3. Liczba użytych barw

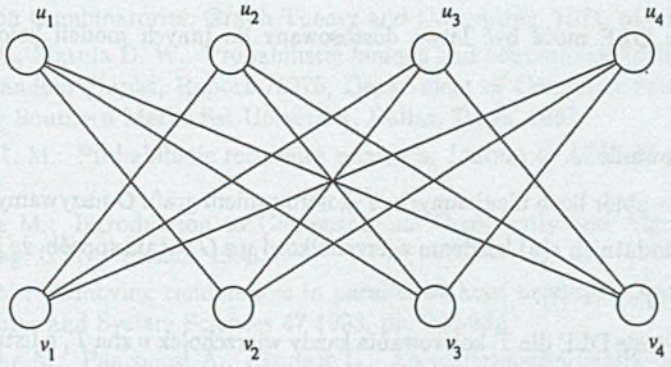
Może się zdażyć, że algorytm DLF, tak jak w poprzednim przykładzie, da optymalne pokolorowanie. Nie jest to jednak regułą. Johnson w [4] podał rodzinę grafów $J_1, J_2, \ldots, J_k, \ldots$, dla których algorytm sekwencyjny LF działa bardzo słabo. Graf J_k jest grafem dwudzielnym z wierzchołkami: $u_1, u_2, \ldots, u_k$ i $v_1, v_2, \ldots, v_k$ oraz wszystkimi krawędziami $\{u_i, v_j\}$ dla $i \neq j$. Przykładowy graf Johnsona znajduje się na rys. 2. Niestety, może się zdarzyć, że algorytm DLF użyje k kolorów do pokolorowania dwudzielnego grafu J_k. Będzie tak na przykład wtedy, gdy w czasie i-tej rundy wierzchołki u_i oraz v_i, mając wyższy priorytet niż pozostałe, otrzymają kolor i. Stąd funkcja dobroci algorytmu jest $\Theta(n)$.

Lepiej przedstawia się sytuacja dla średniej liczby kolorów. Przeprowadzone eksperymenty w przestrzeni grafów losowych wskazują, że jest ona podobna jak dla sekwencyjnego algorytmu LF. Odpowiednie zestawienie wyników znajduje się w tabl. 1.

Tablica 1

Porównanie średniej liczby kolorów dla grafów losowych $G_{n,1/2}$

n	*trivial*	*DLF*	*LF*
10	6,2	4,02	4,04
20	11,6	6,33	6,35
30	16,9	8,38	8,32
50	27,0	12,01	11,95
70	37,2	15,24	15,18
100	52,2	19,85	19,79
120	61,8	22,82	22,75
150	78,3	27,08	26,98
170	86,9	29,71	29,78
200	101,3	33,72	33,73



Rys. 2. Graf Johnsona J_4

Fig. 2. Johnson's graph J_4

5.4. Szybkość

Weźmy pod uwagę graf pełny K_n. Do pokolorowania tego grafu potrzeba n kolorów. To oznacza, że do pokolorowania grafu K_n potrzeba n rund (patrz fakt 3.).

Z powyższego widać, że nie możemy oczekiwać logarytmicznego czasu działania w

każdym wypadku. Jednak w przypadku grafów regularnych możemy sformułować co następuje:

Fakt 4. *Algorytm DLF prawie zawsze działa w czasie $O(\log n)$ rund na grafach regularnych.*

Dowód. Niech G będzie grafem d-regularnym, a v — niepokolorowanym wierzchołkiem na początku pewnej rundy k. Przypuśćmy, że v dla parametru *rndvalue* wylosował wartość x. Prawdopodobieństwo $Pr(v)$, że v otrzyma kolor w czasie tej rundy zależy od x oraz $UN(v)$ – liczby niepokolorowanych sąsiadów v. Możemy zapisać $Pr(v) = x^{UN(v)} > x^d$. Sumując po wszystkich x mamy:

$$Pr(v) = \int_0^1 x^d \mathrm{d}x = \frac{1}{d+1} x^{d+1}\Big|_0^1 = \frac{1}{d+1}$$

Korzystając z twierdzenia 1.1 z pracy [6] otrzymujemy:

$$Pr[T(n) \geq \lfloor \log_{\frac{d+1}{d}}(n) \rfloor + t + 1] \leq \left(\frac{d}{d+1}\right)^{t-1} \quad \square$$

6. Warianty kolorowania

Algorytm DLF może być łatwo dostosowany do innych modeli kolorowania wierzchołkowego.

6.1. T-kolorowanie

Mając T – zbiór liczb nieujemnych, T-kolorowaniem grafu G nazywamy przyporządkowanie liczb dodatnich $c(v)$ każdemu wierzchołkowi v z G w taki sposób, że jeśli $\{u, v\} \in E$, to $|c(u) - c(v)| \notin T$ [7].

W algorytmie DLF dla T-kolorowania każdy wierzchołek v zna T, a lista *usedcolors* zawiera wszystkie kolory zabronione dla v uwzględniając T i kolory wybrane przez sąsiadów.

Na przykład, jeśli $T = \{0, 2\}$, a wierzchołek v ma dwóch sąsiadów pokolorowanych kolorami 1 i 3 odpowiednio, to lista *usedcolors* dla v zawiera kolory $\{1, 3, 5\}$.

6.2. Multikolorowanie

Dla grafu G (f, k)-kolorowaniem, $1 \leq f \leq k$, nazwiemy takie przyporządkowanie f barw ze zbioru k każdemu wierzchołkowi v, że żadne dwa sąsiednie wierzchołki nie mają ani jednego wspólnego koloru.

Modyfikacja algorytmu DLF polega na tym, że każdy wierzchołek zna f i kończy działanie dopiero po znalezieniu wszystkich potrzebnych barw.

Podziękowania

Autor chciałby wyrazić wdzięczność Profesorowi Markowi Kubale oraz Adamowi Nadolskiemu za cenne uwagi i okazaną pomoc.

LITERATURA

1. Battiti R., Bertossi A. A., Bonuccelli M. A.: Assigning codes in wireless networks, Wireless Networks 5 1999, pp. 195-209.
2. Grable D. A., Panconesi A.: Fast distributed algorithms for Brooks-Vizing colourings, 9th Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco), ACM, New York, 1998, pp. 473-480.
3. Johansson Ö.: Simple distributed $\Delta+1$ – coloring of graphs, Information Processing Letters 70, 1999, pp. 229-232.
4. Johnson D. S.: Worst case behavior of graph algorithms, Proc. 5-th Southeastern Conf. on Combinatorics, Graph Theory and Computing, 1974, pp. 513-527.
5. Johri A., Matula D. W.: Probabilistic bounds and heuristic algorithms for coloring large random graphs, Raport 75275, Department of Computer Science and Engineering Southern Methodist University, Dallas, Texas, 1982.
6. Karp R. M.: Probabilistic recurence relations, Journal of ACM 41 1994, pp. 1136-1150.
7. Kubale M.: Introduction to Computational Complexity and Algorithmic Graph Coloring, GTN, Gdańsk, 1998.
8. Luby M.: Removing randomness in parallel without processor penalty, Journal of Computer and System Sciences 47 1993, pp. 250-286.
9. Marathe M., Panconesi A., Risinger L.: An experimental study of a simple, distributed edge coloring algorithm, Raport, Los Alamos National Labolatory, 1999.
10. Shamir E., Upfal E.: Sequential and distributed graph coloring algorithms with performance analysis in random graph spaces, Journal of Algorithms 5 1984, pp. 488-501.
11. Woźniak J., Nowicki K.: Sieci LAN, MAN i WAN - protokoły komunikacyjne. Wydawnicwto Fundacji Postępu Telekomunikacji, Kraków 1998.

Recenzent: Dr hab. inż. Konrad Wala, Prof. AGH

Abstract

We discuss the vertex coloring problem in a distributed network. Such a network consists of processors and bidirectional communication links between pairs of them. It can be modeled by a graph $G = (V, E)$. Set V denotes processors and E models links between them. To color the vertices of a graph G means to give each vertex a color in such a way that no two adjacent vertices get the same color. If at most k colors are used, the result is called a k-coloring.

We assume that there is no shared memory. Each processor knows its own links and its unique identifier. We want these units to compute a coloring of the associated graph without any other information about the structure of G. We will assume that the system is synchronized in *rounds*. The number of rounds will be our measure of efficiency.

Such a model of coloring can be used in a distributed multihop wireless networks to eliminate packet collisions by assigning orthogonal codes to radio stations [1].

In the paper we present a new distributed algorithm for graph coloring. A practical simulation shows that the algorithm performs much better then a naive distributed algorithm.