An experimental study of distributed algorithms for graph coloring

ROBERT JANCZEWSKI, ŁUKASZ KUSZNER, MICHAŁ MAŁAFIEJSKI, ADAM NADOLSKI
Gdańsk University of Technology
Narutowicza 11/12, 80-952 Gdańsk, POLAND,
e-mail: skalar, kuszner, mima, nadolski@eti.pg.gda.pl

Abstract: In the paper we present a new distributed algorithm for coloring the vertices of a graph. A practical simulation shows that this algorithm performs slightly better than a distributed largest-first algorithm known before.

Key words: Graph coloring, distributed algorithms.

1. INTRODUCTION

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 vertices of a graph G means to give different colors to each pair of adjacent vertices. If at most k colors are used, the result is called a *k-coloring*.

The model of distributed processing is as follows: 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 time efficiency.

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

2. PREVIOUS WORK

An algorithm for (Δ+l)-coloring of graphs was given in [3], where Δ is the largest vertex degree in a graph. Also, an analysis of its time complexity was presented. We shall refer to this as the trivial algorithm (same as in [2]). The trivial algorithm is extremely simple and fast, however not optimal. In fact, it uses the number of colors close to Δ even if the graph is bipartite. Nobody should be surprised with that fact because the algorithm has no mechanism to economize colors.
Further improvement was done in [2],where the authors proposed an algorithm using $O(\Delta / \log \Delta)$ colors. It is much better but works on triangle-free graphs only.
As we know, it is better to color vertices with largest degree first. This observation was applied in a distributed largest first algorithm (DLF) introduced in [4]. In this paper we show an innovation to DLF algorithm.

3. ALGORITHM

In the distributed largest first (DLF) algorithm each vertex has three parameters:

- degree: deg(v)
- random value: rndvalue(v)
- palette of forbidden colors, which were used by its neighbors: usedcolor(v) (initially empty).

Parameters: deg(v) and rndvalue(v) determine the order of coloring. Let $v_1, v_2 \in V$. We say that v_1 has a higher priority than v_2 if: deg(v_1) > deg(v_2) or (deg(v_1) = deg(v_2)) and (rndvalue(v_1) > rndvalue(v_2)).
During each round every uncolored vertex v executes the following five steps:

1. Choose parameter rndvalue(v) ∈ [0..1].
2. Send to all neighbors the following parameters: deg(v), rndvalue(v), and the first legal color (not on the list of forbidden colors).
3. Compare its own parameters with these received from the neighbors and check which vertex has the highest priority.
4. If vertex v has the highest priority, keep the proposed color and stop.
5. If not, update list usedcolor(v).

The philosophy behind this priority rules is to create, by each coloring of a vertex as few restrictions on later vertex colors as possible. Let us consider two adjacent vertices during the second round of the coloring process. Suppose that one of these, say v, has only one of its neighbors colored and the other one say u has colored all neighbors except v. In this case we have a conflict, because both u and v need a color number 2. It is clear that v should be colored first regardless of its degree.
Our improvement to DLF is to replace parameter deg with an actual number of uncolored neighbors (dynamic degree - *ddeg*). We shall refer to the new version of algorithm as a *dynamic distributed largest first* (DDLF) algorithm.

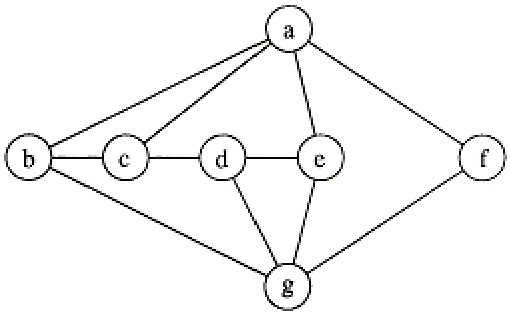

Figure 1: An example graph showing that DDLF may perform better than DLF

3.1 Example

Let us consider the example shown in Figure 1 with $\Delta = 4$. In this graph vertices *a* and *g* are not in a conflict and have four neighbors, hence they both will be colored in the first round with color 1. The is same for both algorithms, however in the second round we have a difference. In the DLF algorithm vertices *b*, *c*, *d* and *e* have the same degree and only random values determine which of them is colored first. Unfortunately, it can happen that *b* and *e* will be colored with 2 and in this case four colors are needed for the whole graph. Using the DDLF algorithm for this graph we always get an optimal solution. In the second round vertex *f* gets a color 2 without obstacles, but vertices *b* and *e* have to lose because their ddeg is smaller than that of their neighbors. We have a conflict between vertices *c* and *d*. Without loss of generality we can suppose that *c* wins and gets a color 2. In the third round we have only *b*, *d* and *e* left uncolored. There is no more conflict so we have: *b* colored with 3, *d* colored with 3, and *e* colored with 2.

4. EFFICIENCY

So, is it true that DDLF is better than DLF for any graph? An example shown in section 3.1 could suggest so, but we could show easily a counterexample that it is not.

We have done some computer experiments for random graphs. We used *n*-vertex random graph $G_{n,p}$ in which each edge appeared independently with probability *p*. In Table 1 we give average numbers of colors and rounds used by the DLF and DDLF algorithms running on the same graphs with $n = 24$. Each average number is taken over 100 iterations. We can observe that DDLF is better than DLF.

	colors C(p)		*rounds T(p)*	
P	DLF	DDLF	DLF	DDLF
0.05	2.52	2.52	2.67	2.67
0.10	3.09	3.10	3.24	3.27
0.15	3.69	3.73	3.83	3.80
0.20	4.16	4.20	4.27	4.28
0.25	4.82	4.73	4.85	4.80
0.30	5.22	5.13	5.29	5.24
0.35	5.60	5.52	5.69	5.57
0.40	6.19	6.08	6.31	6.15
0.45	6.54	6.50	6.57	6.56
0.50	7.21	6.97	7.28	7.04
0.55	7.79	7.56	7.83	7.61
0.60	8.31	8.13	8.37	8.21
0.65	9.16	8.84	9.21	8.88
0.70	9.70	9.55	9.73	9.57
0.75	10.49	10.20	10.51	10.25
0.80	11.32	11.02	11.33	11.03
0.85	12.37	12.09	12.38	12.11
0.90	14.03	13.87	14.05	13.89
0.95	16.88	16.82	16.88	16.82

Table 1: Average number of colors used by DLF and DDLF algorithms for random graphs $G_{24,p}$

All tests have been done in a parallel environment, where each vertex was a separate process. All processes were working independently, without central control. All communications were sent using MPI library. We could observe that if all processes are working then the algorithms never fail, but if something is wrong with at least one process due to allocation problem, incorrect input data or anything else, then whole process collapses. Thus, before practical implementation we need to add some features to assure failure resistance property of the algorithms.

5. ACKNOWLEDGMENT

The authors would like to acknowledge the support of the European Commission through grant number HPRI-CT-1999-00026 (the TRACS Programme at EPCC).

6. REFERENCES

[1] Battiti R., Bertossi A. A., and Bonuccelli M. A. 1999. 'Assigning codes in wireless networks'. *Wireless Networks* 5, pp. 195–209.

[2] Grable D. A. and Panconesi A. 2000. 'Fast distributed algorithms for Brooks-Vising colorings'. *J. Algorithms* 37 no. 1, pp. 85–120.

[3] Johansson Ö. 1999. 'Simple distributed $\Delta + 1$ - coloring of graphs'. *Information Processing Letters* 70, pp. 229–232.

[4] Kubale M. and Kuszner Ł. 2002. 'A better practical algorithm for distributed graph coloring'. *Proc. of IEEE International Conference on Parallel Computing in Electrical Engineering*, pp. 72–75.