A better practical algorithm for distributed graph coloring

Marek Kubale, Łukasz Kuszner
Technical University of Gdańsk, Foundations of Informatics Department
Narutowicza 11/12, 80-690 Gdańsk, POLAND
fax: +(58)3471766 email: kubale@eti.pg.gda.pl

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 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 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].

2 Previous work

In [3] an algorithm for $(\Delta + 1)$-coloring of graphs was given, where Δ is the largest vertex degree in the graph. Also, 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]. In that paper the new algorithm which is able to compute a coloring with $O(\Delta/\log\Delta)$ colors was given. It is much better but works only on triangle-free graphs.

How to use fewer colors in a distributed setting? As we know, it is better to color vertices with larger degree first. This observation is used in some sequential heuristics like largest-first (LF), smallest-last (SL) or saturation largest-first (SLF)(see eg.[6]). Our idea in this paper is to use this observation in a distributed coloring.

3 Algorithm

Each vertex has three parameters:

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

Parameters: $\deg(v)$ and $\mathrm{rndvalue}(v)$ determine the order of coloring. Let $v_1, v_2 \in V$. We say that v_1 has a higher priority then v_2 if:

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

or

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

During each round every uncolored vertex v executes the following five steps:

1. Choose parameter $\mathrm{rndvalue}(v) \in [0..1]$.
2. Send to all neighbors the following parameters: $\deg(v)$, $\mathrm{rndvalue}(v)$, 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 $\mathrm{usedcolor}(v)$.

We shall refer to the new algorithm as a *distributed largest first (DLF) algorithm.*

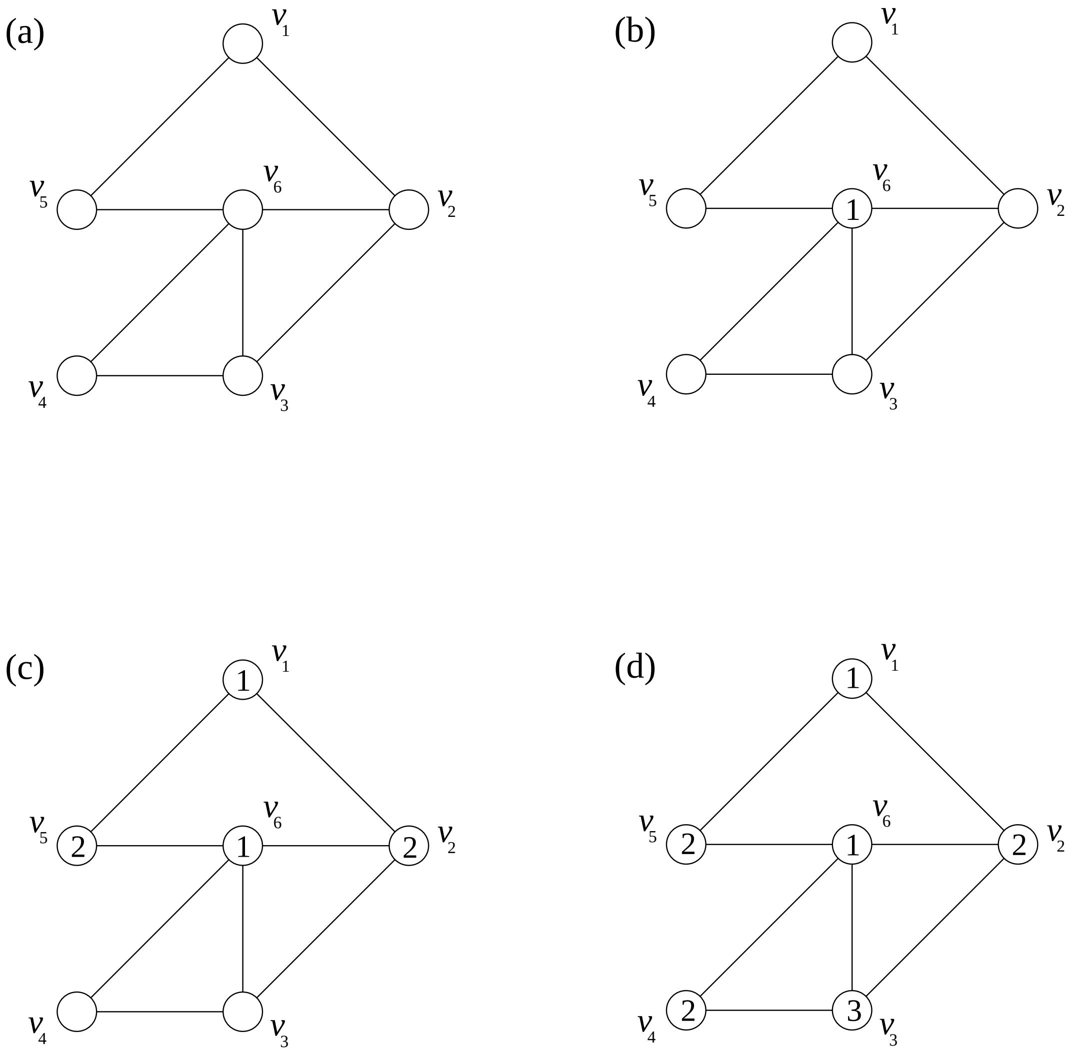

Figure 1: Example of coloring with the *DLF* algorithm.

3.1 Example

Let us consider an example shown in Figure 1(a). In this graph vertex v_6 has 4 neighbors. All other nodes have smaller degree so v_6 gets a color in the first round. Simultaneously v_1 could get a color (only v_1 is not connected with v_6), but its neighbor v_2 has larger degree. The situation after the first round is shown in Fig. 1(b).

Now only v_1 has free color 1, so it can get that color without conflicts. Other vertices can get the color 2. There are conflicts between v_2, v_3 and v_3, v_4. Vertex v_4 has to loose ($\deg(v_4) < \deg(v_3)$), but v_2 and v_3 have the same degree so the parameter *rndvalue* breaks the tie. We can see in Figure 1(c) that v_2 randomized larger number and this vertex has been colored. We have only v_3 and v_4 left. There is no conflict between them because v_4 has 2 as its first legal color and vertex v_3 needs the third color. The final assignment after the third round is shown in Figure 1(d). In this case this is an optimal coloring. The graph contains triangles (v_3, v_4, v_6 and v_2, v_3, v_6) so at least three colors are needed.

4 Analysis

Fact 1 *In each round at least one vertex gets a color.*

Proof: No vertex can get a color unless there is a vertex with locally highest priority. In this case at least two vertices with maximum degree have equal priority. It is possible to neglect the probability of this situation by increasing the precision of tossing the parameter *rndvalue*. $\Box$

Fact 2 *Algorithm DLF never uses more then $\Delta + 1$ colors.*

Proof: Let us consider an uncolored vertex u in the round when it gets a color. In the worst case all its neighbors are colored with different colors. Thus u gets a color of value $\deg(u) + 1 \leq \Delta + 1$. $\Box$

Fact 3 *In each round, in whole graph at most one new color can be assigned.*

Proof: It easily follows from the definition of the algorithm. $\Box$

5 Efficiency

5.1 Colors

Of course it may happen that the DLF algorithm produces an optimal coloring. What about expected number of colors? Some experiments have been done in random graph spaces. In comparison to the sequential LF algorithm DLF seems to work quite satisfactory. In fact, in Table 1 we give average

n	$trivial$	DLF	LF
10	6,2	3,9	3,9
20	11,6	6,3	6,2
30	16,9	8,0	8,1
50	27,0	12,0	12,1
70	37,2	15,1	15,1
100	52,2	20,2	19,9
120	61,8	22,3	22,8
150	78,3	26,9	26,8
170	86,9	29,9	29,8
200	101,3	33,6	33,5

Table 1: Average number of colors used by distributed LF and sequential LF algorithms as compared to the trivial distributed algorithm for random graphs $G_{n,1/2}$.

numbers of colors used by sequential LF and distributed LF algorithms as compared to the trivial distributed algorithm while working on the same graphs.

What about the worst case behavior of the DLF algorithm? Johnson [4] gave a family of graphs $J_1, J_2, \ldots, J_k, \ldots$ for which LF is extremely poor. The construction of this family is as follows: J_k is a bipartite graph with vertices $u_1, u_2, \ldots, u_k$ and $v_1, v_2, \ldots, v_k$ in two subsets and with all possible edges $\{u_i, v_j\}$ except $i = j$. Unfortunately the DLF algorithm is able to color the J_k graph with k colors. This is possible when during round i vertices u_i and v_i obtain both color i. Thus the performance guarantee of DLF is $\Theta(n)$.

5.2 Speed

Let us consider complete graph K_n. To color this graph we need n colors. It means that at least n rounds will be needed to deal with K_n (see Fact 3).

We can not expect logarithmic time complexity in all cases, but what happens if the number of colors is bounded? To answer this question we performed a number of computational experiments with the DLF algorithm on graphs with small degree. Let us have a look at results of DLF for regular graphs shown in Figure 3. The diagram in this figure is in the logarithmic scale. Our experiments allow us to formulate the following:

Fact 4 *The algorithm runs in $O(\log n)$ rounds for regular graphs with constant degree.*

Proof: Let G be a r-regular graph and v an uncolored vertex at the beginning of some round k. Suppose that v generated *rndvalue* equal to x. Probability

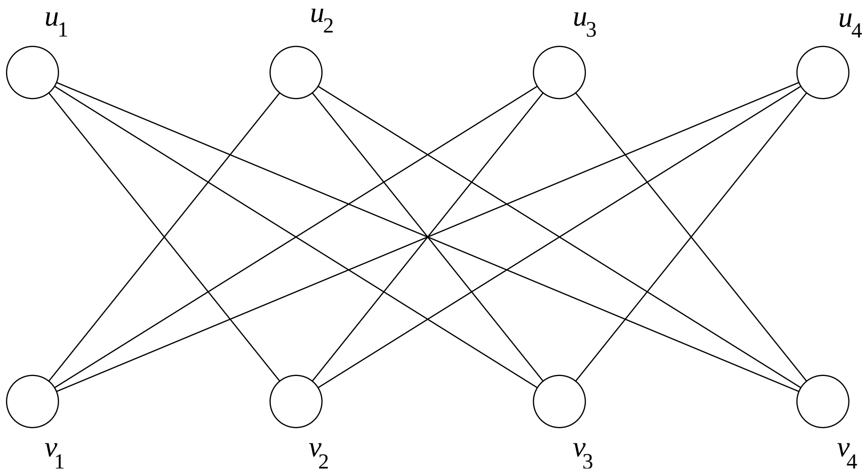

Figure 2: Johnson's graph J_4.

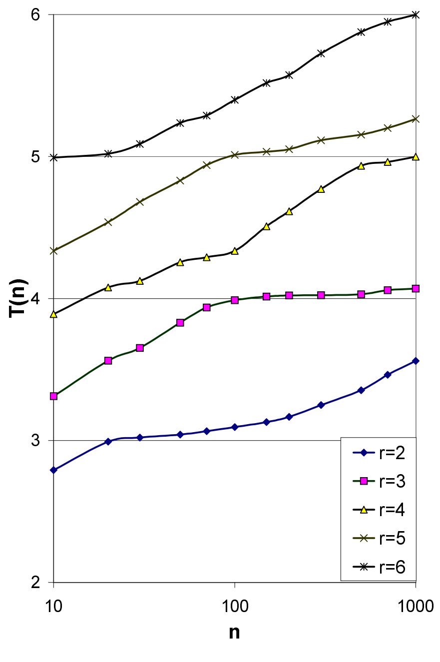

Figure 3: Number of rounds used by DLF to color r-regular graphs with n vertices.

that v gets a color during this round $Pr(v)$ depends on x and the number of uncolored neighbors of v, say $UN(v)$. $Pr(v) = x^{UN(v)} > x^r$. Summing over all x we have:

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

In virtue of Theorem 1.1 in [5] we have:

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

6 Variations of graph coloring

Algorithm DLF can easily be adjusted to some other coloring extensions. Below we give just two of them.

6.1 T-coloring

Given a set T of nonnegative integers including 0, a *T-coloring* of G is an assignment of a positive integer $c(v)$ to each vertex v of G so that if $\{u, v\} \in E$ then $|c(u) - c(v)| \notin T$ [6].

The only modification to DLF is that each vertex v knows set T and the list *usedcolors* contains not only colors used by the neighbors of v but all forbidden colors as well. For example: if $T = \{0, 2\}$ and vertex v has two colored neighbors: u_1 with color 1 and u_2 with color 3, then *usedcolors* for v is $\{1, 3, 5\}$.

6.2 Multicoloring

An (f, k)-coloring of a graph G, $1 \leq f \leq k$ is an assignment of f colors from a k-element palette to each vertex of G in such a way that no two adjacent vertices have the same color in common.

The only modification to DLF is that each vertex knows f, and stops after finding f proper colors.

References

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

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

[3] Ö. Johansson, *Simple distributed* $\Delta{+}1$ — *coloring of graphs*, Information Processing Letters **70** (1999), 229–232.

[4] D. S. Johnson, *Worst case behavior of graph coloring algorithms*, Proc. 5-th Southeastern Conf. on Combinatorics, Graph Theory and Computing, 1974, pp. 513–527.

[5] R. M. Karp, *Probabilistic recurence relations*, Journal of ACM **41** (1994), 1136–1150.

[6] M. Kubale, *Introduction to Computational Complexity and Algorithmic Graph Coloring*, GTN, Gdańsk, 1998.