# Distributed Largest-First Algorithm for Graph Coloring

Jennie Hansen[1], Marek Kubale[2], Łukasz Kuszner[2], and Adam Nadolski[2]

[1] Department of Actuarial Mathematics and Statistics, Heriot-Watt University, Edinburgh EH14 4AS, UK
[2] Department of Algorithms and System Modeling, Gdańsk University of Technology, Narutowicza 11/12, 80-952 Gdańsk, POLAND*

**Abstract.** In the paper we present a distributed probabilistic algorithm for coloring the vertices of a graph. Since this algorithm resembles a largest-first strategy, we call it the distributed LF (DLF) algorithm. The coloring obtained by DLF is optimal or near optimal for numerous classes of graphs e.g. complete $k$-partite, caterpillars, crowns, bipartite wheels. We also show that DLF runs in $O(\Delta^2 \log n)$ rounds for an arbitrary graph, where $n$ is the number of vertices and $\Delta$ denotes the largest vertex degree.

## 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 $G = (V, E)$. The set of vertices $V$ corresponds to processors and the set $E$ of edges models links in the network. To color the vertices of $G$ 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 assume that the system is synchronized in *rounds*. The number of rounds will be our measure of the time complexity. 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].

In evaluating the performance of a random distributed coloring algorithm $\mathcal{A}$ on a graph $G$ there are at least two random variables of interest: $C_{\mathcal{A}}(G)$, the number of colors used by the algorithm to color graph $G$ and $T_{\mathcal{A}}(G)$, the number of rounds used to color $G$.

A good distributed algorithm is one where $C_{\mathcal{A}}(G)$ is close to $\chi(G)$, the chromatic number of $G$, and where $T_{\mathcal{A}}(G)$ is small relative to the number of vertices

* Supported in part by KBN grants: 4T11C04725 and 4T11C00324

in $G$. The difference $C_{\mathcal{A}}(G)-\chi(G)$ can be viewed as a measure of the effectiveness of the algorithm.

We note that it is not always easy to achieve both speed and effectiveness. In [3] a distributed algorithm for $(\Delta+1)$-coloring of graphs was given, where $\Delta$ is the largest vertex degree in a graph. Also, analysis of its time complexity was presented. It was proved to run in $O(\log n)$ time. 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, the number of colors used by the algorithm is close to $\Delta$ even if the graph is bipartite. This is not surprising as the trivial algorithm has no mechanism of economizing on colors. Further improvements to the trivial algorithm were proposed in [2]. In that paper a new algorithm which is able to compute a coloring using $O(\Delta/\log\Delta)$ colors was given, but it works on triangle-free graphs only and fails on some instances of the problem.

One way to improve the performance of a distributed algorithm is to introduce a strategy into the algorithm which is known to be effective in non-distributive coloring algorithms. For example, it is well known that it is much better to color the vertices in a largest degree first order as in some sequential heuristics like largest-first (LF), smallest-last (SL) and saturation largest-first (SLF) (see e.g. [5]). That is why this strategy is incorporated into the distributed largest-first (DLF) algorithm which was introduced in [6] and which is analyzed thoroughly in this paper.

The paper is organized as follows. In the next section the DLF algorithm is carefully defined. In Section 3 we investigate the time complexity of the DLF algorithm in a special case of regular graphs with fixed degree and in the general case, where no assumptions are made about the structure of $G$. In the final section we consider the effectiveness of DLF and we show that for some classes of graphs we have $C_{\mathrm{DLF}}(G)-\chi(G)\leq 1$. This holds even when $\chi(G)<<\Delta$. In particular, on these classes the DLF algorithm is better than the trivial algorithm in the worst case.

## 2 DLF Algorithm

In the DLF algorithm each vertex has three parameters:

- its degree in graph: $\deg(v)$,
- random value, which is generated locally and independently: $\mathrm{rndvalue}(v)$,
- palette of forbidden colors, which have already been used by its neighbors: $\mathrm{usedcolor}(v)$ (initially empty).

Within each round the vertices are trying to obtain their colors. If the same color is chosen by neighboring vertices then parameters: $\deg(v)$ and $\mathrm{rndvalue}(v)$ determine the precedence. Specifically, for neighboring vertices $v_1, v_2 \in V$, we say that the priority of $v_1$ is higher than that of $v_2$ if:

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

or

$$(\deg(v_1) = \deg(v_2)) \text{ and } (\mathrm{rndvalue}(v_1) > \mathrm{rndvalue}(v_2))$$

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

1. Choose parameter rndvalue($v$) uniformly distributed on [0..1].
2. Send to all neighbors the following parameters: deg($v$), rndvalue($v$), and the first legal color (not on vertex $v$'s list of forbidden colors).
3. Compare its own parameters with those received from its neighbors and check which vertex has the highest priority.
4. If vertex $v$'s proposed color does not clash with proposals from its neighbors or if $v$ has the highest priority amongst its neighbors, keep the proposed color, send message to neighbors and stop.
5. If not, update list usedcolor($v$).

## 3 Time Complexity

In this section we investigate the complexity of the DLF algorithm. Let $\mathcal{A}$ be a randomized algorithm working on a graph with $n$ vertices. We say that the complexity of $\mathcal{A}$ is $O(F(n))$ if there exist a constant $c > 0$ and $q(n) \in (0..1)$ such that $1/(1-q(n))$ is $O(F(n))$ and the probability that the processing time of $\mathcal{A}$ exceeds $cF(n)+t$ rounds is at most $q(n)^t$, i.e.

$$\Pr[T_{\mathcal{A}}(G) > cF(n) + t] \leq q(n)^t. \tag{1}$$

It is easy to see that if the complexity of $\mathcal{A}$ is $O(F(n))$, then the mean of $T_{\mathcal{A}}(G)$ is $O(F(n))$ as well.

**Proposition 1.** **[6]** *For any $r-$regular graph $G$ with $n$ vertices the algorithm DLF runs in $O(\log n)$ rounds.*

*Proof.* Let $G$ be an $r-$regular graph and let $v$ be an uncolored vertex at the beginning of some round $k$. Let $\Pr(v)$ denote the probability that $v$ gets a color during this round and let $UN(v)$ denote the number of uncolored neighbors of vertex $v$ at the start of round $k$. Suppose that rndvalue$(v) = x$, then the conditional probability that $v$ gets a color during the $k$-th round, $\Pr[v \,|\, \mathrm{rndvalue}(v) = x]$, is given by $\Pr[v \,|\, \mathrm{rndvalue}(v) = x] = x^{UN(v)} \geq x^r$. It follows that

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

Using this bound and applying established techniques for analyzing probabilistic recurrence relations (see [4], Theorem 1.1), we obtain

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

### 3.1 Time Complexity in a General Case

Now we establish a general upper bound on the complexity of the DLF algorithm.

**Theorem 1.** *For any $n$-vertex graph $G$ the DLF algorithm runs in $O(\Delta^2 \log n)$ rounds.*

*Proof.* Let us consider a graph $G$ with $n$ vertices. The sketch of our reasoning is as follows: we divide the coloring process of $G$ into several phases and estimate the number of rounds in each phase of the process. The first phase starts at the beginning of coloring process and terminates in the first round in which all vertices of degree $\Delta$ are colored. The second phase starts in the next round and is terminated in the round in which all vertices of degree $\Delta - 1$ are colored, and so on.

For $i = 1, \ldots, \Delta$, let $V_i$ denote the set of vertices in $G$ of degree $i$ and let $n_i = |V_i|$. For each $i = 1, \ldots, \Delta$, define the random variable $T_i$ to be equal to the first round in which all the vertices in $G$ of degree at least $i$ are colored, i.e. $T_i$ is the completion time of the $(\Delta + 1 - i)-$th phase of the coloring process. For notational convenience we put $T_{\Delta+1} = 0$.

Now, let $k$ be a positive integer such that $1 \leq k \leq \Delta$. We estimate the probability $\Pr[T_k > s_2 \,|\, T_{k+1} \leq s_1]$ for some positive integers $s_1 < s_2$, i.e. the conditional probability that the $(\Delta + 1 - k)$-th phase finishes after the $s_2$-th round given that the previous phase has been completed by the $s_1$-th round. Consider any uncolored vertex $v$ of degree $k$ in some round $l$, $s_1 < l \leq s_2$. Since $T_{k+1} \leq s_1$, $v$ has no uncolored neighbors of degree greater than $k$. Moreover, it has at most $k$ uncolored neighbors of degree $k$ demanding the same color as $v$, so the probability that $v$ will be colored in round $l$ is at least $1/(k+1)$ (see the proof of Fact 1). Therefore, since the number of uncolored vertices of degree $k$ is at most $n_k$ after the start of the $(s_1 + 1)$-st round, we can apply Theorem 1.1 from [4] again to obtain the following bound

$$\Pr[T_k > s_2 \,|\, T_{k+1} \leq s_1] \leq \left(\frac{k}{k+1}\right)^{t-1}, \tag{2}$$

where $s_2 = s_1 + \lfloor \log_{(k+1)/k} n_k \rfloor + 1 + t$ and integer $t \geq 1$.

Now, let $t_1 > \ldots > t_\Delta > t_{\Delta+1} = 0$ be a sequence of integers. We use the Bayes identity to obtain

$$\Pr[T_k \leq t_k] \geq \Pr[T_k \leq t_k \,|\, T_{k+1} \leq t_{k+1}] \cdot \Pr[T_{k+1} \leq t_{k+1}], \tag{3}$$

for $k = 1, \ldots, \Delta$. Iterating (3) and noting that $T_{\mathrm{DLF}}(G) = T_1$ we have

$$\Pr[T_{\mathrm{DLF}}(G) \leq t_1] \geq \prod_{k=1}^{\Delta} \Pr[T_k \leq t_k \,|\, T_{k+1} \leq t_{k+1}]. \tag{4}$$

Now fix $t \geq 1$ and for $k = 1, \ldots, \Delta$, set

$$t_k = \sum_{i=k}^{\Delta} \left( \lfloor \log_{1+1/i} n_i \rfloor + t + 1 \right).$$

From (2) and (4) it follows that

$$\Pr[T_{\mathrm{DLF}}(G) \le t_1] \ge \prod_{k=1}^{\Delta} \left(1 - \left(\frac{1}{1+1/k}\right)^{t-1}\right) \ge$$
$$\left(1 - \left(\frac{1}{1+1/\Delta}\right)^{t-1}\right)^{\Delta} \ge 1 - \Delta \left(\frac{1}{1+1/\Delta}\right)^{t-1} . \quad (5)$$

The last estimate follows from the Bernoulli inequality. The obvious bound $n_i \le n$ implies that $t_1 \le \Delta \log_{1+1/\Delta}(n) + (t+1)\Delta$. Therefore, (5) leads us to the following bound for all real $t \ge 1$,

$$\Pr[T_{\mathrm{DLF}}(G) > \Delta \log_{1+1/\Delta}(n) + (t+1)\Delta] \le \Delta \left(\frac{1}{1+1/\Delta}\right)^{t-2} . \quad (6)$$

After putting $u = t\Delta$ into (6) we obtain

$$\Pr[T_{\mathrm{DLF}}(G) > \Delta(\log(n)/\log(1+1/\Delta)+1)+u] \le 2(\Delta+1)\left(\frac{\Delta}{1+\Delta}\right)^{u/\Delta} . \quad (7)$$

Then for $s = u + \log_q(2(\Delta+1))$, where $q = (\Delta/(\Delta+1))^{1/\Delta}$, we obtain the following inequality

$$\Pr[T_{\mathrm{DLF}}(G) > \Delta(\log(2(\Delta+1)n)/\log(1+1/\Delta)+1)+s] \le q^s, \quad (8)$$

From (8) it follows that the algorithm runs in $O(\Delta^2 \log n)$ time. □

Notice that the estimates we have used are far from tight, and the bound for the complexity is not effective if $\Delta = \Theta(n)$. However, graphs which occur in applications of the model of distributed graph coloring are typically sparse. For example, if we consider graphs for which $\Delta = O(\log n)$ then the complexity of DLF is $O(\log^3 n)$. Further, if we cannot bound $\Delta$, but we have some information about the structure of graph, so that a better bound on $t_1$ can be derived, then using inequality (5) we can obtain a tighter bound for the complexity of the algorithm.

In the next section we show that the DLF algorithm works particularly well on certain graph classes and for some of these classes it can even be shown that the number of rounds is bounded by a fixed constant independent of the number of vertices in the graph.

## 4 Effectiveness

In this section we consider the accuracy of the DLF algorithm and compare its effectiveness to the effectiveness of the trivial algorithm.

Much work has been done on characterizing the performance of the sequential LF coloring algorithm (complexity, accuracy, hard to color graphs, etc.). So it

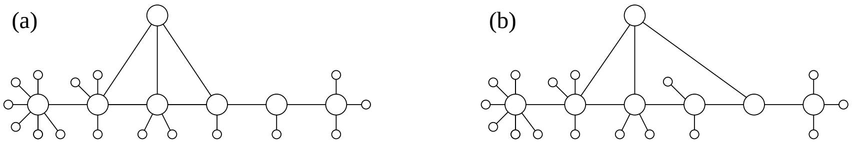


**Fig. 1.** Example graphs

would be useful if known performance results for the LF algorithm could provide bounds on the effectiveness of the DLF algorithm since the latter incorporates the largest-first heuristics.

Unfortunately, this is not the case. For example, consider the three-colorable graphs in Figure 1. It is easy to check that graph (a) is colored optimally by the LF algorithm, whereas it is colored with 4 colors by the DLF algorithm. On the other hand, graph (b) is colored optimally by the DLF algorithm and with one additional color by the LF algorithm.

So what can be said about the effectiveness of the DLF algorithm? Recall that $C_{\mathrm{DLF}}(G)$ denotes the number of colors used by the DLF algorithm to color $G$. It is clear that for any graph $G$ we have the following trivial bounds

$$\chi(G) \leq C_{\mathrm{DLF}}(G) \leq \Delta + 1,$$

but these bounds are useless in assessing the effectiveness of the DLF algorithm when $\chi(G) << \Delta$. On the other hand, in [6] experimental results are obtained which indicate that the expected number of colors used by DLF is much smaller than $\Delta$ on random graphs. For some classes of graphs it is even possible to show that the DLF algorithm is always "nearly optimal". In fact, the examples given below show that there are infinite classes of graphs $\mathcal{G}$ such that for all $G \in \mathcal{G}$

$$C_{\mathrm{DLF}}(G) - \chi(G) \leq 1$$

**Proposition 2.** *The DLF algorithm produces an optimal coloring for complete $k$-partite graphs.*

*Proof.* When $k = 1$ the fact is trivial. Suppose that Fact 2 is true for $i = 1, 2, \ldots k-1$. We will show it to be true for $i = k$. Let $V = P_1 \cup P_2 \cup \ldots \cup P_k$, where $P_i$ are independent sets in $G$. Let $v \in P_i$ be a vertex colored during the first round. Observe that the first color can be assigned to vertices from $P_i$ only. So after the first round only a number of vertices from $P_i$ are colored. In the second round, vertices from $\overline{P_i} = V \backslash P_i$ have color number 1 added to their lists of forbidden colors. So, vertices in $P_i$ which were not colored in the first round obtain color 1 without conflicts during round 2. Now, we can consider vertices from $\overline{P_i}$ separately and $\overline{P_i}$ forms a complete $(k-1)$-partite graph which will be colored with $k-1$ colors. $\square$

**Corollary 1.** [**6**] *For all $p, q \geq 1$, $C_{\mathrm{DLF}}(K_{p,q}) = 2$, where $K_{p,q}$ is a complete bipartite graph.*

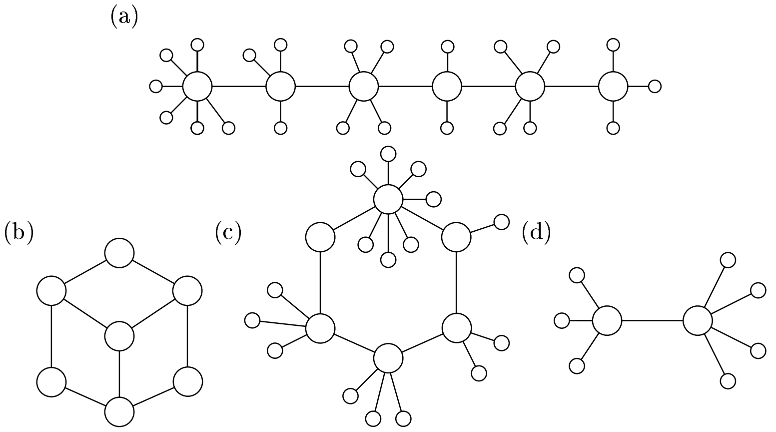


**Fig. 2.** An example of: (a) caterpillar, (b) $BW_3$ - bipartite wheel, and (c) crown, (d) double-star

Complete graph $K_n$ is colored with $n = \Delta + 1$ colors by the DLF algorithm. In addition, when we remove one or two edges from $K_n$ the DLF algorithm still produces an exact coloring.

**Proposition 3.** *For $n \geq 2$, $C_{\mathrm{DLF}}(K_n - e) = \chi(K_n - e) = n - 1$.*

Now consider some special classes of sparse graphs: caterpillars, double-stars, bipartite wheels and crowns.

**Proposition 4.** *Caterpillars are colored with at most three colors by the DLF algorithm.*

*Proof.* A caterpillar is a tree whose leaves are attached to a path (see Fig. 2(a)). Vertices from the path will be colored with at most three colors and each leaf gets a color 1 or 2. □

**Proposition 5.** *For $k \geq 4$ the bipartite wheel $BW_k$ is colored optimally within two rounds and $BW_3$ is colored optimally within three rounds.*

*Proof.* A bipartite wheel is a graph obtained from an even cycle of length $2k$ by adding a central vertex connected to every second vertex of the cycle (see Fig. 2(b)). It is clear that for $k \geq 4$ in the first round the central vertex is colored with color 1. In the second round all vertices obtain colors without conflict. Those connected to the central vertex get color 2 and the others receive color 1. The result for $k = 3$ follows by a similar argument. □

**Proposition 6.** *Crowns are colored with at most three colors.*

*Proof.* A crown is a graph obtained from a cycle by attaching some pendant edges (see Fig. 2(c)). This case is similar to Proposition 4. A cycle will be colored with at most three colors and each leaf will be colored by either color 1 or 2. □

**Proposition 7.** *Double-stars are colored optimally by the DLF algorithm.*

*Proof.* A double-star is a tree with $n-2$ leaves (see Fig. 2(d)). In the first two rounds the vertices of degree greater than 1 are colored with colors 1 and 2. Next, all leaves are colored in the following round. Each of them has exactly one colored neighbor, so it is colored with 1 or 2. □

**Table 1.** Comparison of the number of colors used by the trivial and DLF algorithms in the worst case.

| $G$ | $\chi(G)$ | $C_T(G)$ | $C_{\mathrm{DLF}}(G)$ |
|---|---|---|---|
| complete $k$-partite | $k$ | $\Delta+1$ | $k$ |
| $K_{p,q}$ | 2 | $\Delta+1$ | 2 |
| $K_n-e$ | $n-1$ | $\Delta+1$ | $n-1$ |
| $BW_k$ | 2 | 5 | 2 |
| caterpillar | 2 | $\Delta+1$ | 3 |
| double-star | 2 | 4 | 2 |
| crown | 2 or 3 | $\Delta+1$ | 3 |

In Table 1 we summarize the results given above and compare these results to the worst-case performance of the trivial algorithm on the same classes of graphs. In the first column we list the graph classes and in the second we give the corresponding chromatic number of any graph in the class. For each class the third column gives the worst-case value of $C_T(G)$, the number of colors used by the trivial algorithm, and the fourth column gives the corresponding worst-case value of $C_{\mathrm{DLF}}(G)$. Table 1 shows that for these classes of graphs the DLF algorithm is more effective than the trivial algorithm and, in fact, it always produces an optimal or nearly optimal coloring.

## References


1. Battiti, R., Bertossi, A. A., and Bonuccelli, M. A.: Assigning codes in wireless networks. Wireless Networks **5** (1999) 195–209.
2. Grable, D. A., and Panconesi, A.: Fast distributed algorithms for Brooks-Vizing colorings. J. Algorithms **37** (2000) 85–120.
3. Johansson, Ö.: Simple distributed $\Delta+1$ – coloring of graphs. Inf. Process. Lett. **70** (1999) 229–232.
4. Karp, R. M.: Probabilistic recurrence relations. J. ACM **41** (1994) 1136–1150.
5. Kubale, M.: Introduction to Computational Complexity and Algorithmic Graph Coloring. GTN, Gdańsk, 1998.
6. Kubale, M. and Kuszner, Ł.: A better practical algorithm for distributed graph coloring. Proc. of IEEE International Conference on Parallel Computing in Electrical Engineering (2002) 72–75.