On Greedy Graph Coloring in the Distributed Model

Adrian Kosowski and Łukasz Kuszner*

Department of Algorithms and System Modeling,
Gdańsk University of Technology, Poland

Abstract. In the paper we consider distributed algorithms for greedy graph coloring. For the largest-first (LF) approach, we propose a new distributed algorithm which is shown to color a graph in an expected time of $O(\Delta \log n \log \Delta)$ rounds, and we prove that any distributed LF-coloring algorithm requires at least $\Omega(\Delta)$ rounds. We discuss the quality of obtained colorings in the general case and for particular graph classes. Finally, we show that other greedy graph coloring approaches, such as smallest-last (SL) or dynamic-saturation (SLF), are not suitable for application in distributed computing, requiring $\Omega(n)$ rounds.

1 Introduction

Problem definition. We discuss the vertex coloring problem in a *distributed network*. Such a network consists of a set V of processors and a set E of bidirectional communication links between pairs of processors. It can be modeled by an undirected graph $G = (V, E)$. We denote $n = |V|$, $m = |E|$ and for each vertex v define its *open neighborhood* $N(v) = \{u : \{u, v\} \in E\}$ and *vertex degree* $\deg_G v = |N(v)|$. In order to distinguish neighbours of higher degree, we will use the symbols $N_>(v) = \{u \in N(v) : \deg(u) > \deg(v)\}$ and similarly $N_\geq(v) = \{u \in N(v) : \deg(u) \geq \deg(v)\}$.

To *color* the vertices of G means to give each vertex a positive integer color value 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*. In many practical considerations, such as code assignment in wireless networks [1], it is desirable to minimise number of used colors. The smallest possible positive integer k for which there exists a k-coloring of G is called the *chromatic number* $\chi(G)$. This value is bounded from above by $\Delta + 1$, where Δ denotes the maximum vertex degree of the graph.

Model of computation. We assume the common model used widely in previous research on the subject [3, 11, 16]. Moreover, we assume neither any global parameters known *a priori* for any vertex in a graph, nor unique identifiers. We allow each vertex of the graph to know only its own local state and local states of neighboring vertices. To measure time complexity we use the number of *synchronized rounds* as such a measure is used in most cited material, even though the algorithms discussed here may be adapted for the asynchronous model.

* Supported by State Committee for Scientific Research (Poland) grant no. 4 T11C 047 25.

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

A good distributed algorithm is one where $C_{\mathsf{A}}(G)$ is close to $\chi(G)$, the chromatic number of G, and where $T_{\mathsf{A}}(G)$ is small relative to the number of vertices in G. The difference $C_{\mathsf{A}}(G) - \chi(G)$ can be viewed as a measure of the effectiveness of the algorithm. It has to be remembered though, that in the general case approximating $\chi(G)$ within a factor of $n^{1/7-\varepsilon}$ is an NP-hard problem, for any $\varepsilon > 0$ [2].

1.1 Preliminaries: greedy graph coloring in a distributed context

For a given graph G and the sequence of vertices $K = (v_1, v_2, \ldots, v_n)$, we will use the term *greedy coloring* to describe the following procedure of color assignment:

algorithm Greedy-Color(G, K):
for $v := v_1$ **to** v_n **do**
  give vertex v the smallest possible color;

Definition 1. *A* sequential coloring algorithm *is an algorithm which determines a sequence K of vertices of G, and then colors G using the procedure* Greedy-Color(G, K)*.*

Below we briefly recall the basic principles of the most common sequential algorithms; a more detailed analysis of sequential coloring can be found in [8, 9].

- S algorithm: no assumptions are made concerning sequence K.
- LF algorithm: sequence K is formed by arranging the vertices of graph G in non-ascending order of degrees,
- SL algorithm: sequence K is formed by iteratively removing a vertex of minimal degree from the graph and placing it at the end of K.
- SLF (DSATUR) algorithm: sequence K is formed by dynamically arranging the vertices of graph G in non-ascending order of saturation degrees, where the saturation degree is the number of neighboring vertices which have already been colored (ties are broken by choosing the vertex of greater degree).

For some sequential algorithm A, we will call a coloring of a graph an A-*coloring* if it may be obtained by coloring the graph greedily using a sequence of vertices K legal for algorithm A. In particular, it is easy to observe that an S-coloring of graph G is equivalent to a *Grundy coloring* [5] of G, i.e. such a coloring, that no single vertex may have its color value decreased without affecting the color of some other vertex. All other sequential algorithms also produce Grundy colorings and may also have other, stronger properties (see [9, 14, 17] for an extensive characterisation).

The sequential algorithms discussed in this paper are of practical significance due to the number of graph classes for which they always produce optimal or near-optimal results. It is natural to pose the problem of constructing distributed algorithms with analogous properties.

Definition 2. *A distributed graph coloring algorithm* DA *is said to be a* distributed implementation *of sequential algorithm* A (*or simply: a* distributed A-coloring algorithm) *if all possible results of algorithm* DA *are correct* A-*colorings.*

1.2 Summary of main results

State-of-the-art results. For the general graph coloring problem some extremely fast algorithms have been described. Linial in [11] gave an algorithm working in $O(\log^* n)$ time but using $O(\Delta^2)$ colors. This result was improved later on by De Marco and Pelc [13]. The algorithm given in the paper uses $O(\Delta)$ colors and $O(\log^*(n/\Delta))$ rounds, but local computations are not even polynomially bounded. On the other hand, a very simple algorithm for coloring arbitrary graphs with $(\Delta + 1)$ colors was given by Johansson [7]. It was proved to run in $O(\log n)$ time, but the number of colors used by the algorithm is close to Δ even if the graph is bipartite. This is not surprising, since Johansson's algorithm has no mechanism for economizing on the number of colors. Further improvements were proposed in [4]. In that paper a very similar technique was used to compute a coloring of triangle-free graphs using $O(\Delta/\log\Delta)$ colors, but the algorithm can fail for some instances of the problem.

To the best of our knowledge, the first greedy distributed approach to graph coloring was studied by Panconesi and Rizzi [15]. The authors used a forest decomposition technique to achieve $O(\Delta^2 \log^* n)$ time. Recently an algorithm motivated by sequential LF-coloring was described in [6]. Analysis shows that it runs in $O(\Delta^2 \log n)$ time. However, it sometimes leads to colorings which are not LF-colorings, so the described algorithm is not a distributed implementation of the LF algorithm according to the Definition 2.

Our contribution. In Section 2 we describe a new approach to distributed LF-coloring, showing an algorithm based on iterated maximal independent set construction with $O(\Delta \log n \log \Delta)$ expected runtime, which is shown to be nearly optimal and improves earlier results from [6, 15]. In Subsection 2.3 we briefly discuss the quality of colorings obtained using the proposed algorithm and compare it to its sequential counterpart. In Section 3 we show that every distributed implementation of the LF algorithm requires $\Omega(\Delta)$ time, whereas the SL and SLF approaches may in fact for some graphs require $\Omega(n)$ rounds to color.

2 A distributed implementation of the LF algorithm

Before discussing the details of the distributed implementation of the LF algorithm, we present an equivalent characterization of a correct LF-coloring. For a given coloring of G, let $IS_{(d,c)} \in V$ denote the independent set of vertices of G of degree d and colored with color c.

Lemma 1. *An assignment of colors to G is a correct LF-coloring iff for all $1 \le d, c \le \Delta + 1$ the set $IS_{(d,c)}$ is a maximal independent set in the subgraph $H_{(d,c)}$ of G induced by the set of vertices $\left(\bigcup_{c_i \ge c} IS_{(d,c_i)}\right) \setminus N\left(\bigcup_{d_i > d} IS_{(d_i,c)}\right)$.*

Proof. ($\Rightarrow$) Consider a coloring obtained by an arbitrary sequence of the LF algorithm. Clearly, $IS_{(d,c)}$ is an independent set. By contradiction, suppose that the set $IS_{(d,c)}$ is not maximal and can be extended by some vertex v of $H_{(d,c)}$. This implies that v is of degree $\deg_G v = d$ and has some color $c(v) > c$, which means that it is adjacent in G to a vertex u previously colored by the LF algorithm with color c,

i.e. such that $v \in N(u)$ and $\deg_G u \geq d$. Hence we either have $v \in N(IS_{(d,c)})$ or $v \in N\left(\bigcup_{d_i>d} IS_{(d_i,c)}\right)$, a contradiction.
($\Leftarrow$) It suffices to observe that if a coloring of G fulfills the right hand side of the lemma, then it may be obtained using the LF algorithm with the sequence of vertices: $K = (IS_{(\Delta,1)}, IS_{(\Delta,2)}, \ldots, IS_{(\Delta,\Delta+1)}, IS_{(\Delta-1,1)}, IS_{(\Delta-1,2)}, \ldots, IS_{(\Delta-1,\Delta)}, \ldots, IS_{(1,1)}, IS_{(1,2)})$, where the elements of each independent set may be enumerated in arbitrary order. □

We now propose a distributed algorithm in which each vertex v is characterised by three principal local state variables: $c(v)$ which will store the color of vertex v at the end of the coloring, $d(v)$ which constantly stores the degree of v in G, and a binary flag $f(v)$ which specifies whether v has already reached its final color. Using the terminology from Lemma 1, at any point of the execution of the algorithm we classify the vertices of G into three categories, depending on the information currently available to the vertex from its own local state and the local states of its neighbours:

– v is *correctly colored*, if v can determine that it will belong to $IS_{(d(v),c(v))}$ at the end of the coloring,
– v is *actively uncolored*, if v can determine that it will not belong to any of the sets $IS_{(d(v),1)}, \ldots, IS_{(d(v),c(v)-1)}, N\left(\bigcup_{d_i>d} IS_{(d_i,c)}\right)$, but cannot infer whether it will belong to set $IS_{(d(v),c(v))}$ at the end of the coloring.
– v is *passively uncolored*, if v can determine that it will not belong to any of the sets $IS_{(d(v),1)}, \ldots, IS_{(d(v),c(v)-1)}$, but cannot infer whether it will belong to the set $N\left(\bigcup_{d_i>d} IS_{(d_i,c)}\right)$ at the end of the coloring.

Theorem 1. *There exists a distributed graph coloring algorithm using local state variables $c(v), d(v), f(v)$, such that at any stage of execution each vertex belongs to exactly one of three categories: correctly colored, actively uncolored and passively uncolored. Moreover, a correctly colored vertex will remain correctly colored throughout the rest of the execution.*

Proof (sketch). Let us assume the interpretation of the state variables as in the earlier description. Initially, let $c(v) := 1$ and $f(v) :=$ *false*. The value $f(v)$ will be set to *true* when vertex v becomes correctly colored. Let us assume that throughout the algorithm the value $c(v)$ will never decrease and may only increase when $f(v) =$ *false* and it is certain that v will not belong to set $IS_{(d(v),c(v))}$.

We will show that under these assumptions it is possible to construct an algorithm such that a vertex v with $f(v) =$ *false* is actively uncolored if there does not exist a vertex $u \in N_>(v)$ such that $c(u) \leq c(v)$ and $f(u) =$ *false*, or passively uncolored in the opposite case. Indeed, it suffices that the algorithm repeatedly performs the following two actions in successive rounds:

1. For each vertex v, if there exists a correctly colored vertex $u \in N_\geq(v)$ such that $c(u) = c(v)$, increase $c(v)$ by 1 and repeat the step if necessary.
2. For each actively uncolored vertex v, attempt to include v in the independent set $IS_{(d(v),c(v))}$. If successful, mark v as correctly colored ($f(v) :=$ *true*).

The inclusion of v in the independent set $IS_{(d(v),c(v))}$ performed in Action 2 may only fail if two neighbouring vertices attempt to join the same set simultaneously, thus implying the need for a separate tie-breaking mechanism. It is easy to see that Action 1 is performed only for actively uncolored vertices directly after losing a tie in action 2 and for passively uncolored vertices. Simple inductive reasoning shows that the earlier assumed condition for identifying passively and actively uncolored vertices is indeed correct, which completes the proof. □

Assuming that the actions of the algorithm presented in the proof of Theorem 1 are understood as rounds in the distributed model, we observe that during action 2 all actively uncolored vertices with the same value of variables d and c attempt to join the same independent set $IS_{(d,c)}$. For instance, directly after the initialisation of the algorithm the set of actively uncolored vertices is equal to the set of vertices of degree Δ, all of which attempt to join independent set $IS_{(\Delta,1)}$. The number of such vertices may be arbitrarily large (even equal to n in the case of Δ-regular graphs), thus necessitating an efficient approach to the distributed independent set problem, described in detail in Subsection 2.1.

2.1 Tie-breaking in the distributed independent set problem

The problem of constructing a maximal independent set IS by adding subsets of candidate vertices in successive rounds, encountered in the proof of Theorem 1, has no deterministic solution in the distributed model. The first efficient probabilistic approach was proposed by Luby [12] for a parallel processing system, but due to its nature the algorithm may also be applied in a distributed setting. For a given graph G, let $S \subseteq V$ denote an independent set of vertices and let $P_i \subseteq V \setminus (S \cup N(S))$ be the set of candidates for inclusion into S in the i-th stage of the algorithm. The algorithm divides set P_i into three disjoint subsets, $P_i = W_i \cup N(W_i) \cup P_{i+1}$, where W_i denotes the independent set of vertices merged with S at the end of the stage (known as *winners*, $S := S \cup W_i$), $N(W_i)$ is the neighborhood of W_i which will never enter S (known as *losers*), and P_{i+1} is the set of candidates remaining for later consideration. The process continues until for some k we have $P_k = \emptyset$, then $IS := S$ is a maximal independent set, with respect to the set of all candidate vertices.

The details of the i-th stage of the algorithm may be written as follows. Let H_i denote the subgraph of G induced by set P_i and let E_i be its edge set. First, each vertex $v \in P_i$ is either assigned local state value $r(v) = 0$ and transferred to P_{i+1} with probability $1 - 1/(2 \deg_{H_i} v)$, or *contends* for a place in W_i with probability $1/(2 \deg_{H_i} v)$. Next, each of the contending vertices v draws a random number with uniform distribution. A contending vertex v becomes a winner if $r(v) > \max_{u \in N(v)} r(u)$, and becomes a loser in the opposite case. Luby showed that the described algorithm fulfills the following property.

Theorem 2 ([12]). *Let D_i denote the random variable given as the ratio $D_i = E_{i+1}/E_i$. Then the mean value of D_i is bounded by $\mathrm{E}[D_i] \leq 7/8$.*

Observe that for obvious reasons the value of variable D_i lies within the range $D_i \in [0, 1]$. Consequently, from the above theorem we obtain the following conclusion.

Corollary 1. *In each stage of the algorithm, the number of edges in the candidate set decreases by not less than $1/16$-th part with probability at least $1/15$, $\Pr[D_i \geq 1/16] \geq 1/15$.*

Let us now consider the random variable T describing the number of rounds performed by the maximum independent set algorithm before its completion. Let random variable L_i be defined as $L_i = \log_{16/15} E_i$. Initially, $L_0 = \log_{16/15} E_0 \in O(\log m) = O(\log n)$. In each stage of the algorithm (which may easily be implemented in the form of three rounds), the value $L_i - L_{i+1}$ is always non-negative and, by Corollary 1, not less than 1 with probability at least 1/15. Hence we obtain the following statement.

Corollary 2. *The number of rounds T performed before the candidate set is empty has a probability distribution with mean value* $\mathrm{E}[T] \in O(\log n)$ *and probability mass function $f_T(x)$, bounded from above for $x \in \Omega(\log n)$ by that of the negative binomial distribution with a probability parameter of* $1/15$.

In further analysis it is important to remember that the probability distribution of variable T is understood in terms of randomly drawn local variables, and is independent of the structure of sets S and P_i.

2.2 An algorithm for LF-coloring in $O(\Delta \log n \log \Delta)$ rounds

A formal description of the proposed LF-coloring algorithm is obtained by combining the approach from the proof of Theorem 1 with the results of considerations from Subsection 2.1. For the values $1 \leq d, c \leq \Delta + 1$ we will in parallel be constructing the maximal independent sets $IS_{(d,c)}$. At a given stage of construction of set $IS_{(d,c)}$, the set $S_{(d,c)}$ of known elements will consist of vertices v having $c(v) = c, d(v) = d, f(v) = \mathit{true}$, while the set $P_{(d,c)}$ of candidates will consist of actively uncolored vertices having $c(v) = c, d(v) = d, f(v) = \mathit{false}$. The complete pseudocode of distributed LF-coloring algorithm DLF is given below. Implementation details of independent set tie-breaking follow Subsection 2.1, and we assume that function rnd$[a, b]$ returns an integer with uniform distribution from the range $[a, b]$. Note that when implementing the process of contention for independent set $IS_{(d,c)}$, at most d neighbours contend for one place in the independent set, thus it is sufficient to assume $[1, d^4]$ as the range from which random values $r(v)$ are drawn, without escalating the number of ties.

algorithm DLF(G):
`Round` 0:
  $f(v) := \mathit{false}$; $d(v) = \deg_G(v)$;
`Round` $3k + 1$:
  if $f(v) = \mathit{false}$
    then while $\exists_{u \in N_{\geq}(v)} (c(u) = c(v) \wedge f(u) = \mathit{true})$
      do $c(v) := c(v) + 1$;
`Round` $3k + 2$:
  if $f(v) = \mathit{false} \wedge c(v) < \min_{u \in N_{>}(v)} \{c(u) : f(u) = \mathit{false}\}$
    then if $1 = \mathrm{rnd}[1, 2 \cdot |\{u \in N(v) : d(u) = d(v) \wedge c(u) = c(v)\}|]$
      then $r(v) := \mathrm{rnd}[1, d(v)^4]$;
      else $r(v) := 0$;
`Round` $3k + 3$:
  if $r(v) > \max_{u \in N(v)} r(u)$
    then $f(v) := \mathit{true}$;

$$
\begin{tikzcd}
IS_{(\Delta,1)} \arrow[r] \arrow[d] & IS_{(\Delta,2)} \arrow[r] \arrow[d] & IS_{(\Delta,3)} \arrow[r, dotted] \arrow[d] & IS_{(\Delta,\Delta)} \arrow[r] \arrow[d] & IS_{(\Delta,\Delta+1)} \\
IS_{(\Delta-1,1)} \arrow[r] \arrow[d, dotted] & IS_{(\Delta-1,2)} \arrow[r] \arrow[d, dotted] & IS_{(\Delta-1,3)} \arrow[r, dotted] \arrow[d, dotted] & IS_{(\Delta-1,\Delta)} & \\
IS_{(2,1)} \arrow[r] \arrow[d] & IS_{(2,2)} \arrow[r] \arrow[d] & IS_{(2,3)} & & \\
IS_{(1,1)} \arrow[r] & IS_{(1,2)} & & &
\end{tikzcd}
$$

Fig. 1. Illustration of worst-case time ordering of independent set construction

Theorem 3. *Algorithm* DLF *determines an* LF*-coloring of G in $O(\Delta \log n \log \Delta)$ rounds.*

Proof. The proof of correctness is complete when we observe that by Lemma 1 finding a correct LF-coloring of G is equivalent to determining maximal independent sets $IS_{(\Delta,1)}, IS_{(\Delta,2)}, \ldots, IS_{(\Delta,\Delta+1)}, IS_{(\Delta-1,1)}, IS_{(\Delta-1,2)}, \ldots, IS_{(\Delta-1,\Delta)}, \ldots, IS_{(1,1)}, IS_{(1,2)}$. It is a direct conclusion from Theorem 1 that the DLF algorithm does indeed determine these independent sets through the local variables $(d(v), c(v))$ of the vertices.

Careful analysis of algorithm DLF shows that for any fixed d and c, the process of construction of set $IS_{(d,c)}$ is dependant only on the construction of sets $IS_{(d_i,c_i)}$, for $d_i \geq d, c_i \leq c$. Moreover, vertices once added to an independent set are never removed from it. Without any time gain we may therefore assume that the construction of independent set $IS_{(d,c)}$ starts directly after the construction of sets $IS_{(d+1,c)}$ and $IS_{(d,c-1)}$ is complete. This would result in a time dependency diagram as shown in Figure 1, where a pointer denotes flow of control after completion of the preceding action. Let $T_{(d,c)}$ denote the random variable describing the number of rounds used for the construction of set $IS_{(d,c)}$. From Figure 1 it is easy to observe that the anticipated completion time T_{DLF} of the DLF algorithm is bounded by the expression:

$$T_{\mathsf{DLF}} \leq (\Delta + 1) \cdot \mathrm{E}\left[\max_{1 \leq d,c \leq \Delta+1} T_{(d,c)}\right]$$

However, a characterisation of the mass function of the distribution of $T_{(d,c)}$ is given by Corollary 2. Moreover, this distribution remains the same regardless of the nature of the constructed independent set, thus making the family of variables $T_{(d,c)}$ pairwise independent. By bounding the negative binomial distribution from above by the exponential distribution and performing a number of technical transformations (which we leave out), we obtain the following result:

$$\mathrm{E}\left[\max_{1 \leq d,c \leq \Delta+1} T_{(d,c)}\right] \in O(\log n \log \Delta)$$

Thus, we may finally write $T_{\mathsf{DLF}} \in O(\Delta \log n \log \Delta)$, which completes the proof. □

2.3 Quality characteristics of distributed LF-colorings

As a natural consequence of Lemma 1, algorithm DLF produces worst-case colorings which never use more colors than the worst-case colorings given by a sequential

n	φ	Δ	C_{LF}	C_{DLF}	T_{DLF}
100	0.001	1.21	2.00	2.00	9.67
	0.005	2.81	2.09	2.11	13.06
	0.025	7.00	3.42	3.48	20.71
500	0.001	3.63	2.44	2.52	16.46
	0.005	8.43	3.95	3.97	24.83
	0.025	24.24	7.67	7.71	56.19
2500	0.001	9.61	4.00	4.00	28.53
	0.005	26.51	7.99	8.00	60.58
	0.025	91.73	21.07	21.02	211.36

n	ϱ	Δ	C_{LF}	C_{DLF}	T_{DLF}
1000	2	11.79	4.63	4.68	31.71
	10	35.83	10.06	10.08	80.64
	50	132.38	29.49	29.61	313.87
5000	2	13.23	4.99	5.00	35.32
	10	38.53	10.25	10.30	84.79
	50	138.22	29.28	29.30	324.25
25000	2	14.44	5.00	5.00	38.83
	10	41.51	10.73	10.88	88.32
	50	142.71	29.82	29.82	329.19

Table 1. An experimental average-case comparison of the sequential and distributed LF-coloring algorithms. The tests were conducted for a sample of 100 uniform edge probability random graphs of fixed order n and edge density $\varphi = m/\binom{n}{2}$ (left table) or mean vertex degree $\varrho = 2m/n$ (right table), each of which was colored 10 times by both algorithms.

implementation of LF. It is for instance known that any LF-coloring is optimal or near optimal for numerous graph classes e.g. complete k-partite graphs, caterpillars, crowns, bipartite wheels [9]; as a result, algorithm DLF also performs well for all these graph classes. As a matter of fact, it is easy to observe that the worst case performance of DLF is exactly the same as that of LF, by the following fact (which we leave without proof).

Corollary 3. *A coloring of graph G is an* LF*-coloring iff it is a* DLF*-coloring.*

However, the sequential LF and distributed DLF algorithms may have a different probability of achieving a given coloring, thus affecting their average-case performance. Here we confine ourselves to an experimental comparison of the average number of colors used by LF and DLF for random graphs of different order, edge density and average vertex degree, the results of which are presented in Table 1. It can be clearly seen that the number of colors used by both algorithms is nearly identical, though as a rule marginally smaller for the sequential algorithm. Both LF and DLF clearly outperform all non-greedy algorithms based on the assignment of random colors from the range $[1, \Delta + 1]$. In the conducted simulation, the expected execution time of the DLF algorithm shows a nearly linear relation to the maximum degree Δ of the graph.

3 Complexity bounds on distributed greedy coloring

We will now show that the most popular sequential coloring algorithms impose strong lower bounds on the expected computational time in a distributed setting.

First, consider the SL and SLF algorithms. Both of them exactly color paths and rings, hence all distributed implementations of SL and SLF have the same property. Linial [11] proved that the exact coloring of a ring requires $\Omega(n)$ rounds, so we obtain the following conclusion.

Corollary 4. *Any distributed implementation of* SL *or* SLF *requires $\Omega(n)$ time.*

Now, let us consider the LF algorithm. We will show that any distributed implementation of LF algorithm requires $\Omega(\Delta)$ rounds. To achieve this, we construct a family

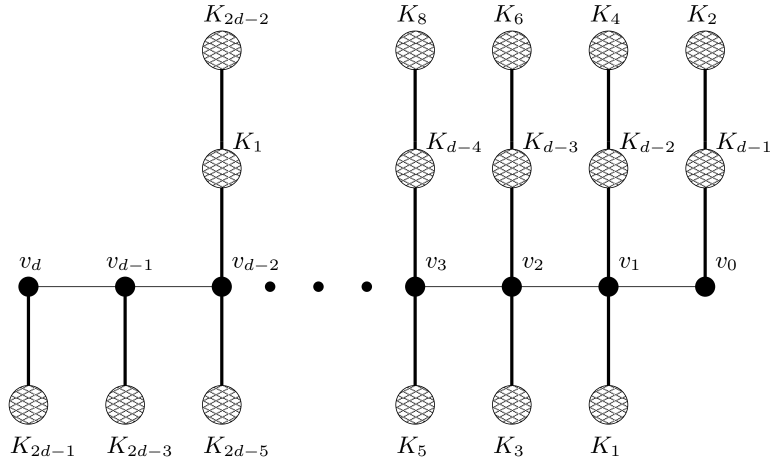

Fig. 2. An example of a graph which requires $\Omega(\Delta)$ time to LF-color in the distributed model

of graphs G_d with $\mathrm{diam}(G_d) = d$ and $\Delta(G) = 2d$. A representative of such a family is depicted in Figure 2. Vertices $v_0, v_1, \ldots, v_d$ induce a path of length d. Some additional components are connected to particular vertices to ensure that vertex v_i obtains a color $d - i + 1$ in each LF-coloring of G_d, that is component K_r depicts a complete graph with r vertices and a bold line between such a component and v_i illustrate that each of the vertices of K_r is connected to v_i. Similarly, when two components K_{r_1} and K_{r_2} are connected, each vertex from K_{r_1} is connected to each vertex from K_{r_2}, thus forming a clique $K_{r_1+r_2}$. We have $\deg v_i = d+i$ and $|N_>(v_i)| = d-i$, hence it is easy to observe that the only possible color for v_i in any LF-coloring is $d-i+1$. However, if vertex v_d were to be removed from the graph, the colors of all other vertices of the path would decrease by 1. Thus we have shown that color of v_0 depends on the value of the length of the path, $d \in \Omega(\Delta)$. As information in our model can propagate only at the speed of one vertex per round we have the following.

Corollary 5. *Any distributed implementation of* LF *requires $\Omega(\Delta)$ time.*

Final conclusions. Taking into account the above corollaries, the $O(\Delta \log n \log \Delta)$ distributed implementation of LF presented in Section 2 may be considered nearly optimal among distributed LF-coloring algorithms, and to have lower complexity than any possible implementation of an SL-coloring or SLF-coloring algorithm. As a matter of fact, it is also difficult to say whether there exists any S-coloring (Grundy coloring) algorithm whose runtime does not involve a similar polynomial Δ-factor [6, 15]. For graphs of bounded degree, the proposed distributed LF-coloring algorithm is, to the best of our knowledge, the first of the well known graph coloring heuristics running in $O(\log n)$ rounds.

References

1. Battiti, R., Bertossi, A. A., and Bonuccelli, M. A.: Assigning codes in wireless networks. Wireless Networks **5** (1999) 195–209.
2. Bellare, M., Goldreich, O., and Sudan, M., Free bits, PCPs and non-approximability — towards tight results, SIAM J. Comp. **27** (1998), 804–915.
3. Chaudhuri, P.: Algorithms for some graph problems on a distributed computational model. Information Sciences **43** (1987), 205–228.
4. Grable, D. A., and Panconesi, A.: Fast distributed algorithms for Brooks-Vizing colorings. J. Algorithms **37** (2000) 85–120.
5. Grundy, P. M.: Mathematics and games. Eureka **2** (1939), 6–8.
6. Hansen, J., Kubale, M., Kuszner, Ł. and Nadolski, A.: Distributed largest-first algorithm for graph coloring. Proc. Euro-Par, LNCS **3149** (2004), 804–811.
7. Johansson, Ö.: Simple distributed $(\Delta + 1)$-coloring of graphs. Inf. Process. Lett. **70** (1999) 229–232.
8. Kosowski, A. and Manuszewski, M.: Classical Coloring of Graphs. In: *Graph Colorings*, AMS Contemporary Math. 352 (2004), Providence, USA, 1–20.
9. Kubale, M.: Introduction to Computational Complexity and Algorithmic Graph Coloring. GTN (1998), Gdańsk, Poland.
10. Kubale, M. and Kuszner, Ł.: A better practical algorithm for distributed graph coloring. Proc. PARELEC (2002) 72–75.
11. Linial, N.: Locality in distributed graph algorithms. SIAM J. Computing **21** (1992) 193–201.
12. Luby, M.: A simple parallel algorithm for the maximal independent set problem. SIAM J. Comput., **15** (1986) 1036–1053.
13. De Marco, G. and Pelc, A.: Fast distributed graph coloring with $O(\Delta)$ colors. Proc. SODA (2001) 630–635.
14. Olariu, S. and J. Randall: Welsh-Powell opposition graphs. Inf. Process. Lett. **31** (1989) 43–46.
15. Panconesi, A. and Rizzi, R.: Some simple distributed algorithms for sparse networks. Distributed Computing **14** (2001), 97–100.
16. Panconesi, A. and Srinivasan, A.: Improved distributed algorithms for coloring and network decomposition problems. Proc. STOC (1992) 581–592.
17. Turner, J. S.: Almost all k-colorable graphs are easy to color. J. Algorithms **9** (1988) 63–82.