Distributed Computing manuscript No.
(will be inserted by the editor)

Distributed Deterministic Edge Coloring using Bounded Neighborhood Independence

Leonid Barenboim * · **Michael Elkin** *

Received: date / Accepted: date

Abstract We study the *edge-coloring* problem in the message-passing model of distributed computing. This is one of the most fundamental problems in this area. Currently, the best-known deterministic algorithms for $(2\Delta-1)$-edge-coloring requires $O(\Delta)+\log^* n$ time [22], where Δ is the maximum degree of the input graph. Also, recent results of [5] for vertex-coloring imply that one can get an $O(\Delta)$-edge-coloring in $O(\Delta^{\epsilon}\cdot\log n)$ time, and an $O(\Delta^{1+\epsilon})$-edge-coloring in $O(\log\Delta\log n)$ time, for an arbitrarily small constant $\epsilon > 0$.

In this paper we devise a significantly faster deterministic edge-coloring algorithm. Specifically, our algorithm computes an $O(\Delta)$-edge-coloring in $O(\Delta^{\epsilon}) + \log^* n$ time, and an $O(\Delta^{1+\epsilon})$-edge-coloring in $O(\log\Delta)+\log^* n$ time. This result improves the state-of-the-art running time for deterministic edge-coloring with this number of colors in almost the entire range of maximum degree Δ. Moreover, it improves it exponentially in a wide range of Δ, specifically, for $2^{\Omega(\log^* n)} \le \Delta \le \text{polylog}(n)$. In addition, for small values of Δ (up to $\log^{1-\delta} n$, for some fixed $\delta > 0$) our deterministic algorithm outperforms all the existing *randomized* algorithms for this problem. Also, our algorithm is the first $O(\Delta)$-edge-coloring algorithm that has running time $o(\Delta) + \log^* n$, for the entire range of Δ. All previous (deterministic and randomized) $O(\Delta)$-edge-coloring algorithms require $\Omega(\min\{\Delta, \sqrt{\log n}\,\})$ time.

On our way to these results we study the *vertex-coloring* problem on graphs with bounded *neighborhood independence*. This is a large family of graphs, which strictly includes line graphs of r-hypergraphs (i.e., hypergraphs in which each hyperedge contains r or less vertices) for $r = O(1)$, and graphs of bounded growth. We devise a very fast deterministic algorithm for vertex-coloring graphs with bounded neighborhood independence. This algorithm directly gives rise to our edge-coloring algorithms, which apply to *general* graphs.

Our main technical contribution is a subroutine that computes an $O(\Delta/p)$-defective p-vertex coloring of graphs with bounded neighborhood independence in $O(p^2) + \log^* n$ time, for a parameter p, $1 \le p \le \Delta$. In all previous efficient distributed routines for m-defective p-coloring the product $m \cdot p$ is super-linear in Δ. In our routine this product is *linear* in Δ, and this enables us to speed up the algorithm drastically.

Keywords Defective-Coloring · Legal-Coloring · Line-Graphs

This research is supported by the Binational Science Foundation, grant No. 2008390. The first-named author is supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

* Department of Computer Science, Ben-Gurion University of the Negev, POB 653, Beer-Sheva 84105, Israel. E-mail: {leonidba,elkinm}@cs.bgu.ac.il

1 Introduction

1.1 Edge-Coloring

We study the *edge-coloring* problem in the *message passing model* of distributed computing. Specifically, we are given an n-vertex undirected unweighted graph $G = (V, E)$, with each vertex hosting an autonomous processor. The processors have distinct identity numbers (henceforth, Ids) from the range $\{1, 2, ..., n\}$. They communicate with each other over the edges of E. The communication occurs in discrete rounds. In each round each vertex can send a message to each of its neighbors, and these messages arrive to their destinations before the next round starts. The running time of an algorithm in this model is the number of rounds of communication that are required for the algorithm to terminate.

A legal *edge-coloring* φ of $G = (V, E)$ is a function $\varphi : E \rightarrow \mathbb{N}$ that satisfies that for any pair of edges $e, e' \in E$ that share an endpoint (henceforth, *incident*), it holds that $\varphi(e) \neq \varphi(e')$. Denote by $\Delta = \Delta(G)$ the maximum degree of the graph G. A classical theorem of Vizing [28] shows that for any graph G, its edges can be legally colored in $(\Delta + 1)$ colors. Obviously, at least Δ colors are required.

The edge coloring problem is one of the most fundamental problems in Graph Theory and Graph Algorithms. It also has numerous applications in Computer Science, including job-shop scheduling, packet routing, and resource allocation [15,11]. This problem was also extensively studied in the message-passing model [9–11,13,22,24]. Panconesi and Rizzi [22] showed that a $(2\Delta - 1)$-edge-coloring can be computed deterministically in $O(\Delta) + \log^* n$ time. Panconesi and Srinivasan [24] devised a randomized $(1.6\Delta + O(\log^{1+\epsilon} n))$-edge coloring algorithm that runs in polylogarithmic time, where $\epsilon > 0$ is an arbitrarily small constant. Dubhashi et. al. [10] used the Rődl nibble method to improve this to a randomized $(1+\epsilon)\Delta$-edge-coloring in time $O(\log n)$, as long as $\Delta = \omega(\log n)$. Grable and Panconesi [13] showed that if for every edge $e = (u, w)$, the degree of either u or w is sufficiently large (at least $2^{\Omega(\frac{\log n}{\log \log n})}$), then a $(1 + \epsilon)\Delta$-edge-coloring, for an arbitrarily small constant $\epsilon > 0$, can be computed in $O(\log \log n)$ time by a randomized algorithm. Czygrinow et. al. [9] devised a deterministic $O(\Delta \log n)$-edge-coloring that requires $O(\log^4 n)$ time.

A more general approach to the edge-coloring and many other related problems was taken in [1,21,23]. These papers presented algorithms that compute a *network decomposition*, i.e., a partition of the input graph into regions of small diameter. This partition admits also additional helpful properties. This partition can then be used to compute edge-coloring, vertex-coloring, maximal independent set, and other related structures. In particular, by this technique one can get a deterministic $(2\Delta - 1)$-edge-coloring algorithm that requires $2^{O(\sqrt{\log n})}$ time [23,1].

A (legal) vertex coloring ψ of $G = (V, E)$ is a function $\psi : V \rightarrow N$ that satisfies that for any edge $e = (u, w) \in E$, $\psi(u) \neq \psi(w)$. We refer to $\psi(u)$ as the *ψ-color of u*. By considering the line graph $L(G) = (E, \mathcal{E} = \{(e, e') \mid e \text{ and } e' \text{ share a vertex }\})$, it is easy to see that any vertex-coloring algorithm that employs $f(\Delta)$ colors, for a function $f()$, translates into an edge-coloring algorithm that employs $f(2(\Delta{-}1))$ colors, with essentially the same running time [1]. This observation enables one to harness many of the recent advances in vertex-coloring for obtaining significantly faster edge-coloring algorithms as well. Most relevant in this context are the results of [17,27,5]. Kothapalli et. al. [17] showed that an $O(\Delta)$-vertex-coloring (and, consequently, $O(\Delta)$-edge-coloring as well) can be computed in $O(\sqrt{\log n})$ rounds, by a randomized algorithm. Recently Schneider and Wattenhofer [27] devised a randomized algorithm that computes (1) a $(\Delta{+}1)$-vertex-(and $(2\Delta{-}1)$-edge-) coloring in $O(\log \Delta + \sqrt{\log n})$ time; (2) an $O(\Delta + \log n)$-coloring in $O(\log \log n)$ time; and (3) an $O(\Delta \log^{(k)} n + \log^{1+1/k} n)$-coloring in $f(k) = O(1)$ time, for some fixed function $f()$ and any positive integer k. In [5] the authors of the current paper devised a deterministic algorithm that, for an arbitrarily small constant $\epsilon > 0$, computes (1) an $O(\Delta^{1+\epsilon})$-coloring in $O(\log \Delta \log n)$ time; and (2) an $O(\Delta)$-coloring in $\Delta^\epsilon \log n$ time.

In the current paper we show that in the case of *edge*-coloring the factor $\log n$ can be eliminated. Specifically, we devise a deterministic algorithm that for an arbitrarily small constant $\epsilon > 0$, computes (1) an $O(\Delta^{1+\epsilon})$-edge-coloring in $O(\log \Delta) + \log^* n$ time; (2) an $O(\Delta)$-edge-coloring in $O(\Delta^\epsilon) + \log^* n$ time. We remark that our algorithm is the *first deterministic or randomized* $O(\Delta)$-edge-coloring algorithm that requires $o(\Delta) + \log^* n$ time, for the entire range of Δ. (The term $\log^* n$ is necessary, in view of Linial's lower bound [20].) All previous deterministic $O(\Delta)$-edge-coloring algorithms require $\Omega(\min\{\Delta, \log^{1+\epsilon} n\})$ time, while previous randomized $O(\Delta)$-edge-coloring algorithms require $\Omega(\min\{\Delta, \sqrt{\log n}\})$ time.

More generally, these results compare very favorably to the state-of-the-art. We start with comparing them to deterministic algorithms. For Δ in the range $\omega(\log^* n) \leq \Delta \leq O(\log n \log \log n)$ the fastest currently known algorithm for edge-coloring with $O(\Delta^{1+\epsilon})$ or less colors is due to Panconesi and Rizzi [22]. Its running time is $O(\Delta) + \log^* n$. Our algorithm runs *exponentially* faster, in time $O(\log \Delta) + \log^* n$, but it employs more colors ($O(\Delta^{1+\epsilon})$ instead of $(2\Delta - 1)$). In addition, another variant of our algorithm employs only $O(\Delta)$ colors, and has a significantly better running time than that of the algorithm of [22], specifically, $O(\Delta^\epsilon) + \log^* n$. We stress that this result holds in particular for the range $\Delta = o(\log n)$. In this range, all previously known deterministic algorithms for $O(\Delta)$-edge-coloring and $O(\Delta)$-vertex-coloring on general graphs have running time $\Omega(\Delta)$. Therefore, our algorithm is the first to compute an $O(\Delta)$-edge-coloring in sublinear in Δ time on general graphs.

For $\Delta = \Omega(\log n \log \log n)$ the fastest known algorithm for edge-coloring with $O(\Delta^{1+\epsilon})$ colors is due to [5]. Its running time is $O(\log \Delta \log n)$, instead of $O(\log \Delta) + \log^* n$ for our new algorithm. Note that as

[1] As long as one allows arbitrarily large messages.

long as Δ is at most polylogarithmic in n, the new running time is $O(\log\log n)$ instead of $O(\log n \log\log n)$ of [5], i.e., our improvement in this range is exponential as well. To summarize, our algorithm improves the state-of-the-art running time for deterministic algorithms in almost the entire range of the maximum degree Δ, i.e., for $\Delta = \omega(\log^* n)$, and it improves it exponentially for $2^{\Omega(\log^* n)} \leq \Delta \leq O(\log^k n)$, for an arbitrarily large constant k. We remark that even when Δ is greater than polylogarithmic in n, the running time $O(\log\Delta + \log^* n)$ of our $O(\Delta^{1+\epsilon})$-coloring algorithm is at least *quadratically* smaller than the previous state-of-the-art $O(\log\Delta \log n)$ due to [5]. To summarize, the factor of $\log n$ is a very significant factor in this context, and eliminating it results in major improvements. See Table 1 below for a concise comparison of previous and new deterministic results.

Next, we compare the running time and the number of colors of our *deterministic* algorithm with the state-of-the-art with respect to *randomized* algorithms. For $\Delta = \Omega(\log n)$ the recent randomized algorithm of Schneider and Wattenhofer [27] outperforms our algorithm. However, for $\Delta \leq \log^{1-\delta} n$, for an arbitrarily small constant $\delta > 0$, the algorithm of [27] either employs $\Omega(\log n)$ colors (i.e., more than $\Delta^{1+\epsilon}$ for an arbitrarily small $\epsilon > 0$), or its running time is $\Omega(\sqrt{\log n})$. (Note, however, that the randomized algorithm of [27] solves a generally harder vertex-coloring problem, rather than edge-coloring.) Hence in the range $\omega(\log^* n) \leq \Delta \leq \log^{1-\delta} n$, for some fixed constant $\delta > 0$, our deterministic algorithm outperforms all previous algorithms, deterministic and randomized. Moreover, in the range $2^{\Omega(\log^* n)} \leq \Delta \leq \log^{1-\delta} n$ our algorithm is *exponentially faster* than the previous ones. Indeed, for $\Delta \leq \sqrt{\log n}$ the best previous algorithm that achieves $O(\Delta^{1+\epsilon})$ or less colors is due to [22], whose running time is $O(\Delta) + \log^* n$. On the other hand, the running time of our algorithm is $O(\log\Delta) + \log^* n$. For $\sqrt{\log n} \leq \Delta \leq \log^{1-\delta} n$ the best previous algorithms that achieve that many colors are due to [27,17], and their running time is $O(\sqrt{\log n})$. Our algorithm requires in this range just $O(\log\Delta) + \log^* n = O(\log\log n)$ time. (On the other hand, the variant of the algorithm of [27] that runs in $O(\sqrt{\log n})$ time employs just $(2\Delta - 1)$ colors, as opposed to $\Delta^{1+\epsilon}$ colors that are employed by our algorithm. The algorithm of [22] also employs only $(2\Delta - 1)$ colors.) See Table 2 below for a concise comparison.

Observe also that the $\log^* n$ term in the running time of our algorithms is optimal up to a factor of 2, in view of the lower bounds of [20]. Specifically, Linial's lower bound [20] implies that $f(\Delta)$-edge-coloring, for any fixed function $f()$, requires at least $\frac{1}{2}\log^* n$ time. Moreover, there is a variant of our algorithm that achieves the precisely optimal additive term of $\frac{1}{2}\log^* n$, while achieving almost the same dependence on Δ. By "almost" the same dependence on Δ we mean that it achieves (for an arbitrarily small constant $\epsilon > 0$), (1) an $O(\Delta)$-edge-coloring in $O(\Delta^\epsilon) + \frac{1}{2}\log^* n$ time, and (2) a $\Delta^{1+o(1)}$-edge-coloring in $O(\log\Delta \cdot \frac{\log^* \Delta}{\log(\log^* \Delta)}) + \frac{1}{2}\log^* n$ time. In other words, item (1) is the same as that cited above, except that $\log^* n$ is replaced by $\frac{1}{2}\log^* n$, and in item (2) there is also a tiny slack factor of $\frac{\log^* \Delta}{\log(\log^* \Delta)}$.

1.2 Bounded Neighborhood Independence

Our results for edge-coloring follow from far more general results that we describe below. *Neighborhood independence* $I(G)$ of a graph $G = (V, E)$ is the maximum number of independent [1] neighbors of a single vertex $v \in V$. The family of graphs with constant neighborhood independence (henceforth, *bounded neighborhood independence*) is a very general family of graphs. Indeed, for any graph G, the neighborhood independence of its line graph $L(G)$ is at most 2. Moreover, for an r-hypergraph $\mathcal{H}$ (i.e., a hypergraph in which every hyperedge contains at most r vertices), $I(L(\mathcal{H})) \leq r$.

Another important family of graphs which is subsumed by the family of graphs with bounded neighborhood independence is the family of graphs of *bounded growth*. A graph $G = (V, E)$ is said to be of bounded growth if there exists a function $f()$ such that for any $r = 1, 2, \ldots$, the number of independent vertices at distance at most r from any given vertex is at most $f(r)$. Distributed algorithms for vertex-coloring and computing a maximal independent set on graphs from this family is a subject of intensive recent research [16,12,26]. The crowning result of this effort is the deterministic algorithm of [26] that computes a maximal independent set and a $(\Delta + 1)$-vertex-coloring for graphs from this family in optimal time $O(\log^* n)$. Note, however, that a graph G with a constant neighborhood independence may contain an arbitrarily large independent set U whose all vertices are at distance at most 2 from some given vertex v in G. Thus, graphs with bounded neighborhood independence may have unbounded growth. (Consider, for example, a graph H that is obtained by connecting each vertex of an $n/2$-vertex clique with a distinct isolated vertex. Each vertex in H has at most 2 independent neighbors. However, each vertex v in the clique has at least $n/2 = \Omega(\Delta)$ independent vertices at distance at most 2 from v, and so the graph H is not a graph of bounded growth.)

Yet another family of graphs which is subsumed by the family of graphs of bounded independence is the family of *claw-free* graphs. A graph is *claw-free* if it excludes $K_{1,3}$ as an induced subgraph. (In fact, for any

[1] Two vertices u, w are *independent* in G if $(u, w) \notin E$.

Range of Δ	$\omega(\log^* n) = \Delta = o(\log n \log\log n)$	$\Omega(\log n \log\log n) = \Delta$
Previous	$(2\Delta - 1)$ colors, $O(\Delta) + \log^* n$ time [22]	$O(\Delta)$ colors, $O(\Delta^\epsilon \log n)$-time [5]
		$O(\Delta^{1+\epsilon})$ colors, $O(\log \Delta \log n)$-time [5]
New	$O(\Delta)$ colors, $O(\Delta^\epsilon) + \log^* n$ time	$O(\Delta)$ colors, $O(\Delta^\epsilon) + \log^* n$ time
	$O(\Delta^{1+\epsilon})$ colors, $O(\log \Delta) + \log^* n$ time	$O(\Delta^{1+\epsilon})$ colors, $O(\log \Delta) + \log^* n$ time

Table 1 *A concise comparison of previous state-of-the-art edge-coloring* deterministic *algorithms with our new algorithms.*

Range of Δ	$\omega(\log^* n) = \Delta = O(\sqrt{\log n})$	$\Omega(\sqrt{\log n}) = \Delta \le \log^{1-\delta} n$
Previous	$(2\Delta - 1)$ colors, $O(\Delta) + \log^* n$ time [22]	$(2\Delta - 1)$ colors, $O(\sqrt{\log n})$ time [27]
New (Deter.)	$O(\Delta^{1+\epsilon})$ colors, $O(\log \Delta) + \log^* n$ time	$O(\Delta^{1+\epsilon})$ colors, $O(\log\log n)$ time

Table 2 *A concise comparison of previous state-of-the-art edge-coloring* randomized and deterministic *algorithms with our new* deterministic *algorithm. The algorithm of [22] is deterministic. The algorithm of [27] is randomized.*

$r = 2, 3, ...$, the family of graphs with independence at most r is precisely the family of graphs that exclude induced $K_{1,r+1}$.) The family of claw-free graphs attracted enormous attention in Structural Graph Theory. See, e.g., the series of papers by Chudnovsky and Seymour, starting with [6].

In this paper we devise a vertex-coloring algorithm for graphs of bounded neighborhood independence that computes (for an arbitrarily small constant $\epsilon > 0$) (1) an $O(\Delta)$-vertex-coloring in $O(\Delta^\epsilon) + \frac{1}{2}\log^* n$ time, and (2) a $\Delta^{1+o(1)}$-vertex-coloring in $O(\log \Delta \cdot \frac{\log^* \Delta}{\log(\log^* \Delta)}) + \frac{1}{2}\log^* n$ time. Modulo some subtleties, these results imply our main results about edge-coloring described in Section 1.1. In addition, they apply to line graphs of r-hypergraphs for any constant r, to claw-free graphs, to graphs of bounded growth, and to many other graphs.

1.3 Our Techniques

In the heart of our algorithms lie improved algorithms for computing *defective colorings*. For a non-negative integer m and positive integer χ, an *m-defective χ-vertex-coloring* φ of a graph $G = (V, E)$ is a function $\varphi : V \to \{1, 2, ..., \chi\}$ that satisfies that for every vertex $v \in V$, it has at most m neighbors colored by $\varphi(v)$. The parameter m is called the *defect* of the coloring. Defective coloring was introduced by Cowen et. al [7] and by Harary and Jones [14]. It was extensively studied from a graph-theoretic perspective [2,8]. Recently defective coloring was discovered to be very useful in the context of distributed graph coloring [4,5,18]. In particular, the state-of-the-art $(\Delta + 1)$-vertex-coloring algorithms for general graphs [4,18] are based on subroutines for computing defective coloring. For a parameter p, these subroutines compute an $O(\Delta/p)$-defective p^2-coloring. (In [4] the running time of such a subroutine is $O(p^2) + \frac{1}{2}\log^* n$, and in [18] it is $O(\log^* \Delta) + \frac{1}{2}\log^* n$.) It was observed in [4] that one could have devised significantly faster coloring algorithms if there were an efficient (distributed) routine for computing an m-defective χ-coloring with a *linear* in Δ product of m and χ. (The current state-of-the-art [4,18], has $m = O(\Delta/p)$, $\chi = p^2$, i.e., $m \cdot \chi = O(\Delta \cdot p)$ instead of the desired $O(\Delta)$.)

In this paper we show that if one restricts the attention to the family of bounded neighborhood independence graphs, then this goal can be achieved. Specifically, we devise an algorithm that computes an $O(\Delta/p)$-defective p-vertex-coloring of a given graph of bounded neighborhood independence in $O(p^2) + \frac{1}{2}\log^* n$ time. As a result we obtain a bunch of drastically faster algorithms for vertex-coloring these graphs, and, consequently, for edge-coloring general graphs.

Whether it is possible to devise an efficient $O(\Delta/p)$-defective p-vertex-coloring algorithm for general graphs remains a challenging open question. In [5] the authors of the current paper were able to circumvent this question by means of *arbdefective coloring*. Note that an $O(\Delta/p)$-defective p-vertex-coloring can be seen as a partition of the vertex set into p subsets, each inducing a subgraph of maximum degree at most $O(\Delta/p)$. In [5] the authors showed that the vertex set of a graph of arboricity [1] a can be efficiently partitioned into p subsets, each inducing a subgraph of arboricity $O(a/p)$. This partition is then employed in [5] to devise a suite of efficient algorithms for vertex-coloring general graphs. In particular, using this technique [5] devised an $O(\Delta^{1+\epsilon})$-vertex coloring algorithm in $O(\log \Delta \log n)$ time, for an arbitrarily small constant $\epsilon > 0$.

Note, however, that the factor of $\log n$ in the running time of the algorithms of [5] is inherent, because these algorithms rely heavily on the notion of arboricity, and more specifically, on the machinery of forest-decompositions developed in [3] for working with graphs of bounded arboricity. On the other hand, a lower bound

[1] The *arboricity* of a graph $G = (V, E)$ is $a(G) = \max\{\left\lceil \frac{|E(U)|}{|U|-1} \right\rceil : U \subseteq V, |U| \ge 2\}$.

shown in [3] stipulates that computing a forest decomposition requires $\Omega(\frac{\log n}{\log a})$ time, where a is the arboricity. Consequently, the factor of $\log n$ in the running time is unavoidable [1] using the approach of [5]. In the current paper we pursue a different line of attack. Specifically, we devise improved algorithms for *defective* coloring, rather than circumventing it and going through *arbdefective* coloring.

1.4 Structure of the Paper

In Section 2 we describe definitions and notation employed in our algorithms. In Section 3 we devise defective vertex-coloring algorithms for graphs with bounded neighborhood independence. In Section 4 we devise legal vertex-coloring algorithms for this family of graphs. In Section 5 we devise legal edge-coloring algorithms for general graphs. In Section 6 we present an extension to our algorithms.

2 Preliminaries

Unless the base value is specified, all logarithms in this paper are to base 2. For a non-negative integer i, the *iterative log-function* $\log^{(i)}(\cdot)$ is defined as follows. For an integer $n > 0$, $\log^{(0)} n = n$, and $\log^{(i+1)} n = \log(\log^{(i)} n)$, for every $i = 0, 1, 2, ...$ Also, $\log^* n$ is defined by: $\log^* n = \min\left\{i \mid \log^{(i)} n \leq 2\right\}$.

The *degree* of a vertex v in a graph $G = (V, E)$, denoted $\deg(v) = \deg_G(v)$, is the number of edges incident to v. A vertex u such that $(u, v) \in E$ is called a *neighbor* of v in G. The *neighborhood* $\Gamma(v) = \Gamma_G(v)$ of v is the set of neighbors of v. The maximum degree of a vertex in G, denoted $\Delta(G)$, is defined by $\Delta = \Delta(G) = \max_{v \in V} \deg(v)$. The graph $G' = (V', E')$ is a *subgraph* of $G = (V, E)$, denoted $G' \subseteq G$, if $V' \subseteq V$ and $E' \subseteq E$. The notation $V(G')$ and $E(G')$ is used to denote the vertex set V' of G', and the edge set E' of G', respectively.

The *line graph* $L(G) = (V'', E'')$ of a graph $G = (V, E)$ is a graph in which V'' contains a vertex v_e for each edge $e \in E$, and an edge $(v_e, v_{e'})$ if and only if the edges e and e' of E share a common endpoint. We say that a vertex $v_e \in V''$ and an edge $e \in E$ *correspond* to each other. By definition, for any graph G and positive integer k, a legal k-coloring of *vertices* of $L(G)$ is a legal k-coloring of *edges* of G, and vice versa. Note also that for an edge $e = (u, w)$ in G, the number of edges incident to it is $(\deg(u) - 1) + (\deg(w) - 1)$. Hence the maximum degree $\Delta(L(G))$ of the line graph $L(G)$ satisfies $\Delta(L(G)) \leq 2(\Delta - 1)$, where $\Delta = \Delta(G)$.

The *out-degree* of a vertex v in a directed graph $\hat{G}$ is the number of edges incident to v that are oriented outwards of v. An *orientation* σ of (the edge set of) a graph is an assignment of direction to each edge $(u, v) \in E$, either towards u or towards v. An edge (u, v) that is oriented towards v is denoted by $\langle u, v \rangle$. The *out-degree* of an orientation σ of a graph G is the maximum out-degree of a vertex in G with respect to σ. In a given orientation, each neighbor u of v that is connected to v by an edge oriented towards u is called a *parent* of v. In this case we say that v is a *child* of u.

For a graph $G = (V, E)$, a set of vertices $U \subseteq V$ is called an *independent set* if for every pair of vertices $v, w \in U$ it holds that $(v, w) \notin E$.

The minimum number of colors that can be used in a legal vertex-coloring of a graph G is called *the chromatic number* of G, denoted $\chi(G)$.

Next, we state a number of known results that will be used in our algorithms.

Lemma 21. *(1) [20, 25] A legal $O(\Delta^2)$-vertex-coloring can be computed in $\log^* n + O(1)$ time.*
(2) [4, 18] A legal $(\Delta + 1)$-vertex-coloring can be computed in $O(\Delta) + \log^* n$ time.
(3) [18] A $\lfloor \Delta/p \rfloor$-defective $O(p^2)$-vertex-coloring can be computed in $O(\log^* n)$ time.

3 Defective Coloring

In this section we present a defective vertex coloring algorithm for graphs with bounded neighborhood independence. We begin with a formal definition of this family of graphs.

Definition 31. Graphs with neighborhood independence bounded by c.
For a graph $G = (V, E)$ and a vertex $v \in V$, the *neighborhood independence* of v, denoted $I(v)$, is the size of a maximum-size independent subset $U \subseteq \Gamma(v)$.
The *neighborhood independence* of a graph G is defined as $I(G) = \max_{v \in V}\{I(v)\}$. For a positive parameter c, a graph $G = (V, E)$ is said to have *neighborhood independence bounded by* c if $I(G) \leq c$.

Let c be a fixed positive constant, and p be a parameter such that $1 \leq p \leq \Delta$. We devise a procedure, called *Procedure Defective-Color*, that computes an $O(\Delta/p)$-defective p-coloring on graphs with neighborhood independence bounded by c. This coloring is achieved by first computing a defective $O(p^2)$-coloring, and then reducing the number of colors to p, using special properties of graphs with bounded neighborhood independence. Procedure Defective-Color receives as input a

[1] In fact, the algorithm of [5] computes an $O(a^{1+\epsilon})$-coloring in time $O(\log a \log n)$ for graphs of arboricity a. An $O(\Delta^{1+\epsilon})$-coloring in $O(\log \Delta \log n)$ time is a direct corollary of this result. On the other hand, it is known [3] that $O(a^{1+\epsilon})$-coloring requires $\Omega(\frac{\log n}{\log a})$ time.

graph G with neighborhood independence bounded by c, a positive parameter b, and the parameter Λ which serves as an upper bound on the maximum degree of the input graph. The parameter b satisfies that $b \geq 1$, $b \cdot p \leq \Lambda$. This parameter controls the tradeoff between the defect of the resulting coloring and the running time of the procedure. Specifically, the defect behaves as $\frac{\Lambda}{p}(1 + O(1/b))$, and the running time is at most $O(b^2 \cdot p^2 + \log^* n)$. We assume that all vertices know the value of c before the computation starts.

The procedure starts with computing a $\lfloor \Lambda/(b \cdot p) \rfloor$-defective $O((b \cdot p)^2)$-coloring φ of G using Lemma 21(3). The coloring φ is employed for computing another defective coloring ψ of the vertices of G. The recoloring step spends one round for each φ-color class. Specifically, each vertex $v \in V$ computes $\psi(v)$ as follows. The vertex v waits for each neighbor u of v with $\varphi(u) < \varphi(v)$ to select a color $\psi(u)$. Once v receives a message from each such neighbor u with its color $\psi(u)$, it sets $\psi(v)$ to be a value from the range $\{1, 2, .., p\}$ that is used by the minimum number of neighbors u with $\varphi(u) < \varphi(v)$. Once v selects its color $\psi(v)$, it sends it to all its neighbors. This completes the description of the algorithm.

We need the following piece of notation. For a vertex v and an index $k \in \{1, 2, ..., p\}$, let $N_v(k) = |\{u \in \Gamma(v) \mid \psi(u) = k, \varphi(u) < \varphi(v)\}|$ denote the number of neighbors u of v that have smaller φ-color than v has, and whose ψ-color was set to k. Next, we provide the pseudocode of Procedure Defective-Color.

Algorithm 1 Procedure Defective-Color(G, b, p, Λ)

An algorithm for each vertex $v \in V$

1: $\varphi(v)$:= compute $\lfloor \Lambda/(b \cdot p) \rfloor$-defective $O((b \cdot p)^2)$-coloring using Lemma 21(3)
2: send $\varphi(v)$ to all neighbors
3: $\psi(v) := 0$
4: **while** $\psi(v) = 0$, in each round **do**
5:   **if** v received $\psi(u)$ for each neighbor u of v with $\varphi(u) < \varphi(v)$ **then**
6:     $m := \min\{N_v(k) \mid k \in \{1, 2, ..., p\}\}$
7:     $\psi(v)$:= a color $k \in \{1, 2, ..., p\}$ such that $N_v(k) = m$
8:     send $\psi(v)$ to all neighbors
9:   **end if**
10: **end while**

Observe that a vertex v waits only for neighbors with smaller φ-color before selecting $\psi(v)$. Consequently, it selects the color $\psi(v)$ after at most $\varphi(v)$ rounds from the time when step 2 of Algorithm 1 was executed. This fact is stated in the following lemma.

Lemma 31. *Let φ be the coloring computed in the first step of Algorithm 1. Let R be the round in which step 2 is executed. A vertex v selects a color $\psi(v) \neq 0$ in round $R + \varphi(v)$ or earlier.*

Proof Let $\ell = O((b \cdot p)^2)$ be the number of colors employed by φ. The lemma is proved by induction on the number of rounds. We prove that once a round $i = R + 1, R + 2, \ldots, R + \ell$ is completed, all vertices v with $\varphi(v) \leq i$ have already selected the color $\psi(v)$. For the base case, observe that the vertices v with $\varphi(v) = 1$ have no neighbors with smaller φ-color. Therefore they select a ψ-color in round $R + 1$, immediately after receiving the φ-colors of their neighbors in round R. For the induction step, suppose that in round $i - 1$ all vertices v with $\varphi(v) \leq i - 1$ have already selected the color $\psi(v)$. Hence, each vertex u with $\varphi(u) \leq i$ receives the color ψ for each neighbor with smaller φ-color before round i. Consequently, the vertices u select a ψ-color in round i or earlier. $\square$

Since the coloring φ employs $\ell = O((b \cdot p)^2)$ colors, it follows that all vertices select a ψ-color at most ℓ rounds after the computation of the defective coloring in step 1 of Algorithm 1. By Lemma 21(3), step 1 requires $O(\log^* n)$ rounds. The overall running time of Algorithm 1 is given in the following corollary.

Corollary 32. *The running time of Procedure Defective-Color is $O((b \cdot p)^2 + \log^* n)$.*

In what follows we prove the correctness of Procedure Defective-Color. By the Pigeonhole principle, the number of neighbors u of a vertex v such that $\varphi(u) < \varphi(v)$ and $\psi(u) = \psi(v)$ is at most Λ/p. (Otherwise, there are more than (Λ/p) neighbors of v that are colored by a ψ-color i, for each $i = 1, 2, ..., p$. Hence, v has more than Λ neighbors, a contradiction.) In addition, since φ is a $\lfloor \Lambda/(b \cdot p) \rfloor$-defective coloring, there are at most $\lfloor \Lambda/(b \cdot p) \rfloor$ neighbors u of v that have the same φ-color as v has, i.e., satisfy $\varphi(u) = \varphi(v)$. These neighbors may also end up selecting the same ψ-color that v selects. Hence the number of neighbors u of v that satisfy $\varphi(u) \leq \varphi(v)$ and $\psi(u) = \psi(v)$ is at most $\Lambda/p + \Lambda/(b \cdot p)$. In order to prove that ψ is an $O(\Lambda/p)$-defective coloring we also prove a somewhat surprising claim regarding the other neighbors of v. Specifically, we prove that the number of neighbors u of a vertex v such that $\varphi(u) > \varphi(v)$ and $\psi(u) = \psi(v)$ is $O(\Lambda/p)$ as well. Consequently, ψ is an $O(\Lambda/p)$-defective p-coloring.

For $i = 1, 2, ..., p$, let G_i be the subgraph induced by the ψ-color class i, i.e., by the vertex set $\{v \in V \mid \psi(v) = i\}$. As a first step we show that the chromatic number $\chi(G_i)$ of G_i is at most $(\Lambda/(b \cdot p) + \Lambda/p) + 1$. (See Lemma 34.) We prove this claim by presenting an acyclic orientation of G_i with out-degree at most $(\Lambda/(b \cdot p) + \Lambda/p)$. Since a graph with acyclic orientation with out-degree d is legally $(d+1)$-colorable (see Lemma 33), the claim follows. Then we show that bounded chromatic number in conjunction with bounded neighborhood independence imply a bounded degree.

Lemma 33. *A graph G with an acyclic orientation μ with out-degree d satisfies that $\chi(G) \leq d+1$.*

Lemma 33 is well-known. We provide its proof for completeness.

Proof We color the vertex set of G as follows. A vertex v waits for all its neighbors u connected to v by outgoing edges $\langle v, u\rangle$ to select a color. Then it selects a color from the range $\{1, 2, ..., d+1\}$ that is not used by any such neighbor. (The number of neighbors u as above is at most the out-degree of the orientation, that is, at most d.) Since the orientation is acyclic, this process terminates and produces a $(d+1)$-coloring. The coloring is legal since for each edge $(u, v) \in E$, if it is oriented by μ towards v, the vertex u selects a color that is different from the color of v. Otherwise, v selects a color that is different from the color of u. □

Lemma 34. *For $i = 1, 2, ..., p$, it holds that $\chi(G_i) \leq (\Lambda/(b \cdot p) + \Lambda/p) + 1$.*

Proof Let μ_i be the following orientation of G_i. For each edge $e = (u, v) \in E(G_i)$, orient the edge towards the endpoint that is colored by a smaller φ-color. If $\varphi(u) = \varphi(v)$, then orient e towards the endpoint with smaller Id among u, v. Each vertex v in G_i has at most Λ/p neighbors u in G_i with smaller φ-colors. (This is because $\psi(u) = \psi(v)$ and $\varphi(u) < \varphi(v)$.) In addition, each vertex v in G_i has at most $\Lambda/(b\cdot p)$ neighbors u in G_i with $\varphi(v) = \varphi(u)$. Consequently, μ_i has out-degree at most $(\Lambda/(b \cdot p) + \Lambda/p)$.

Next, we prove that the orientation μ_i is acyclic. Let C be a cycle of G_i. Let v be a vertex on the cycle C with the largest φ-color. If there are several vertices that satisfy this condition, let v be the vertex with the greatest Id. Let u and w be the neighbors of v in C. Since $\varphi(u) < \varphi(v)$ or ($\varphi(u) = \varphi(v)$ and $\text{Id}(u) < \text{Id}(v)$), the edge (v, u) is oriented by μ_i towards u. Similarly, the edge (v, w) is oriented towards w. Therefore, C is not a consistently oriented cycle. Consequently, μ_i is acyclic. Since the out-degree of μ_i is at most $(\Lambda/(b \cdot p) + \Lambda/p)$, Lemma 33 implies that $\chi(G_i) \leq (\Lambda/(b \cdot p) + \Lambda/p) + 1$. □

The next lemma shows that the family of graphs with bounded neighborhood independence is closed under taking vertex-induced subgraphs.

Lemma 35. *For a positive integer c, let $G = (V, E)$ be a graph with neighborhood independence at most c. The subgraph induced by a subset $U \subseteq V$ also has neighborhood independence at most by c.*

Proof Let $\mathcal{G} = G(U)$ be the subgraph induced by U. For a vertex $u \in U$, $\Gamma_{\mathcal{G}}(u)$ is the neighborhood of u in $\mathcal{G}$, and $\Gamma_G(u)$ is the neighborhood of u in G. Suppose for contradiction that there exists a vertex $u \in U$ such that there is an independent set $W \subseteq \Gamma_{\mathcal{G}}(u)$ with cardinality $|W| > c$. For a pair of vertices $v, w \in W$, it holds that $(v, w) \notin E$. In addition, $\Gamma_{\mathcal{G}}(u) \subseteq \Gamma_G(u)$. Therefore, $W \subseteq \Gamma_G(u)$ is an independent set with more than c vertices. This is a contradiction. □

We employ Lemmas 33 - 35 to prove the correctness of Procedure Defective-Color.

Theorem 36. *Suppose that Procedure Defective-Color is invoked on a graph G with maximum degree Δ and with neighborhood independence bounded by a positive constant c. Suppose also that it receives as input three integer parameters $b \geq 1, p \geq 1$, $\Lambda \geq 1$, such that $b \cdot p \leq \Lambda$, and $\Lambda \geq \Delta$. Then Procedure Defective-Color computes a $((\Lambda/(b\cdot p)+\Lambda/p)\cdot c+c)$-defective p-coloring.*

Proof Recall that G_i is the graph induced by vertices with ψ-color i returned by Procedure Defective-Color, for $i = 1, 2, ..., p$. By Lemma 35, since G_i is an induced subgraph of G, the neighborhood independence of G_i is bounded by c. We prove that the maximum degree of G_i, for $i = 1, 2, ..., p$, is at most $(\Lambda/(b \cdot p) + \Lambda/p) \cdot c + c$. Suppose for contradiction that there is a vertex $v \in G_i$ such that $\deg_{G_i}(v) > (\Lambda/(b \cdot p) + \Lambda/p) \cdot c + c$. Let φ' be a legal coloring of G_i that employs the minimum number of colors. Each color class of φ' is an independent set. Therefore, for a positive integer q, the number of neighbors u of v such that $\varphi'(u) = q$ is at most c. Consequently, the number of different colors employed for coloring the set $\Gamma_{G_i}(v)$ of neighbors of v in G_i is at least $\left\lceil \deg_{G_i}(v)/c \right\rceil > (\Lambda/(b \cdot p) + \Lambda/p) + 1$. However, by Lemma 34, $\chi(G_i) \leq (\Lambda/(b \cdot p) + \Lambda/p) + 1$, a contradiction. □

We summarize this section with the following corollary.

Corollary 37. *For any constant ϵ, $0 < \epsilon < 1$, and an integer parameter p, $1 \leq p \leq \Lambda \cdot \epsilon/2c$, $\Lambda \geq \Delta$, a $((c+\epsilon) \cdot \frac{\Lambda}{p} + c)$-defective p-coloring of a graph G with $I(G) \leq c$ can be computed in $O(p^2 + \log^* n)$ time.*

Proof Set $b = \lceil \frac{c}{\epsilon} \rceil$. By Theorem 36, Procedure Defective-Color computes a defective p-coloring with a defect parameter $((\Lambda/(b \cdot p) + \Lambda/p) \cdot c + c)$
$= ((\Lambda/(\lceil \frac{c}{\epsilon} \rceil \cdot p) + \Lambda/p) \cdot c + c) \leq \epsilon \cdot \frac{\Lambda}{p} + c \cdot \frac{\Lambda}{p} + c$
$= (c+\epsilon) \cdot \frac{\Lambda}{p} + c$. By Corollary 32, the running time is $O(p^2 + \log^* n)$, since c and ϵ are constants. □

Observe that for graphs with bounded neighborhood independence, the product of the defect $O(\Delta/p)$

and the number of colors p in the coloring produced by Corollary 37 is $O(\Delta)$. This is in sharp contrast to the current state-of-the-art for distributed defective coloring in *general* graphs [4,18], which is $O(\Delta/p)$-defective p^2-coloring. On the other hand, the latter coloring can be computed faster, specifically, within $O(\log^* n)$ time [18].

4 Legal Coloring Graphs with Bounded Neighborhood Independence

In this section we employ the defective coloring algorithm from the previous section to produce a legal vertex coloring of graphs with neighborhood independence at most c. We start with presenting a simpler algorithm that does not achieve our strongest bounds (Section 4.1). Then we proceed to improving these bounds further (Section 4.2).

4.1 The main algorithm

In this section we employ the defective coloring algorithm from the previous section for legal vertex coloring of graphs with neighborhood independence at most $c = O(1)$. Fix an arbitrarily small constant $\epsilon > 0$. Once a $(c+\epsilon) \cdot \frac{\Delta}{p} + c = O(\Delta/p)$-defective p-coloring ψ of a graph G is computed, it constitutes a vertex partition $V_1, V_2, ..., V_p$, in which V_i is the set of vertices with ψ-color i, for $i = 1, 2, ..., p$. In other words, $V = \bigcup_{i=1}^{n} V_i$, and for a pair of distinct indices $i, j \in \{1, 2, ..., p\}$, $i \neq j$, $V_i \cap V_j = \emptyset$. The subgraph G_i induced by V_i has maximum degree $O(\Delta/p)$, since each vertex in G has at most $O(\Delta/p)$ neighbors with the same ψ-color. Therefore, one can legally color the subgraphs $G_1, G_2, ..., G_p$ employing $O(\Delta/p)$ colors for each subgraph, using Lemma 21(2). These colorings, denoted $\varphi_1, \varphi_2, ..., \varphi_p$, are computed in parallel on the subgraphs $G_1, G_2, ..., G_p$. Let $m = O(\Delta/p)$ denote the maximum number of colors employed by φ_i, for $i = 1, 2, .., p$. Next, the colorings are combined into a unified legal coloring φ of G as follows. Observe that each vertex $v \in V$ belongs exactly to one subgraph G_j among $G_1, G_2, ..., G_p$. We set $\varphi(v) = \varphi_j(v) + (j-1) \cdot m$. (The color $\varphi(v)$ can also be thought of as a pair $(j, \varphi_j(v))$, where $v \in V_j$.)

The coloring φ is a legal coloring of G, since for any pair of vertices $u, v \in G$, if they belong to the same subgraph G_i, $i \in \{1, 2, ..., p\}$, then $\varphi_i(v) \neq \varphi_i(u)$, and, therefore, $\varphi(v) \neq \varphi(u)$. Otherwise, u belongs to G_i, and v belong to G_j, for some $1 \leq i \neq j \leq p$. Since $|(j-1) \cdot m - (i-1) \cdot m| \geq m$, and $|\varphi_j(u) - \varphi_i(v)| \leq m-1$, in this case also it holds that $\varphi(v) \neq \varphi(u)$.

The running time required for computing φ is the running time of computing a legal coloring of a graph with degree $\left\lfloor (c+\epsilon) \cdot \frac{\Delta}{p} + c \right\rfloor = O(\Delta/p)$. Hence, by Lemma 21(2), the running time of computing φ from a given $O(\Delta/p)$-defective p-coloring ψ is $O(\Delta/p + \log^* n)$. By Corollary 37, the running time of computing ψ is $O(p^2 + \log^* n)$. Therefore, the overall running time of computing φ from scratch is $O(\Delta/p + p^2 + \log^* n)$. This running time is optimized by setting $p = \lfloor \Delta^{1/3} \rfloor$, resulting in overall $O(\Delta^{2/3} + \log^* n)$ time. The number of colors employed by the resulting legal coloring is at most $((c+\epsilon) \cdot \frac{\Delta}{p} + c + 1) \cdot p \leq (c+\epsilon') \cdot \Delta$, for any constant ϵ', $\epsilon' > \epsilon$. (Note that $c = O(1)$ and $\Delta/p = \omega(1)$). We summarize this result in the following lemma.

Lemma 41. *For any constant $\epsilon' > 0$, a legal $((c+\epsilon') \cdot \Delta)$-coloring of graphs with neighborhood independence at most c can be computed in $O(\Delta^{2/3} + \log^* n)$ time.*

Next, we present a significantly faster $O(\Delta)$-coloring procedure for the family of graphs with neighborhood independence bounded by c, for a positive constant c. The procedure is called *Procedure Legal-Color*. During its execution defective colorings are computed several times. In the first phase of the procedure a defective coloring of the input graph is computed. This coloring forms a partition of the original graph into vertex-disjoint subgraphs, each with maximum degree smaller than Δ. Then the procedure is invoked recursively on these subgraphs in parallel. This invocation partitions each subgraph into more subgraphs with yet smaller maximum degrees. This process repeats itself until the maximum degrees of all subgraphs are sufficiently small. Then, legal colorings of these subgraphs are computed in parallel, and merged into a unified legal coloring of the input graph. Even though Procedure Defective-Color is invoked many times by Procedure Legal-Color, the running time of Procedure Legal-Color is much smaller than the time given in Lemma 41. The improvement in time is achieved by selecting different parameters in the defective colorings computations, making the invocations significantly faster than a single invocation with the parameter $p = \lfloor \Delta^{1/3} \rfloor$.

In particular, in the algorithm that was described above we partitioned the vertex set of the graph into p disjoint subsets with maximum degree $\Delta' \leq (c+\epsilon')\frac{\Delta}{p}$ each. In each of the p subgraphs one can invoke recursively the algorithm from Lemma 41. It will produce a $((c+\epsilon')^2 \cdot \frac{\Delta}{p})$-coloring of each of the subgraphs, i.e., a $((c+\epsilon')^2 \cdot \Delta)$-coloring of the original graph. The running time becomes $O((\frac{\Delta}{p})^{2/3} + p^2 + \log^* n)$. By setting $p = \Delta^{1/4}$, we achieve the running time $O(\Delta^{1/2} + \log^* n)$. Generally, suppose for a constant $i \in \{1, 2, ...\}$ that we have a $((c+\epsilon')^i \cdot \Delta)$-coloring algorithm with running time $O(\Delta^{\frac{2}{2+i}} + \log^* n)$. Then the above argument converts it into a $((c+\epsilon')^{i+1} \cdot \Delta)$-coloring algorithm with running time $O(\Delta^{\frac{2}{2+(i+1)}} + \log^* n)$. To summarize:

Theorem 42. *For any constant $i \in \{1, 2, ...\}$, and any arbitrarily small constant $\eta > 0$, a $((c^i + \eta) \cdot \Delta)$-coloring of a graph G with maximum degree Δ and neighborhood independence at most c can be computed in $O(\Delta^{\frac{2}{2+i}} + \log^* n)$ time.*

In what follows we extend this argument to the case of superconstant i. For this end we need to take a closer look on the constants that are involved in the analysis of the above algorithm. Denote $\alpha = \log^* n$. Let $\vartheta > 0$ be a constant such that (1) a legal $(\Delta+1)$-vertex-coloring can be computed in at most $\vartheta \cdot \Delta + \alpha$ time (See Lemma 21(2); we denote this algorithm $\mathcal{A}_0(\Delta)$), and (2) a $(c+\epsilon)(1+\frac{p}{\Delta})\frac{\Delta}{p}$-defective p-coloring of a graph G with $I(G) \le c$ can be computed in at most $\vartheta(p^2 + \alpha)$ time. (See Procedure Defective-Color and Corollary 37.)

Let p be a sufficiently large constant that satisfies however that $\frac{p}{\Delta} \le \epsilon$. (For this condition to hold we need $\Delta \ge \frac{p}{\epsilon}$. Indeed, otherwise a $(\Delta+1)$-vertex-coloring can be computed in $\alpha + O(1)$ time by $\mathcal{A}_0(\Delta)$.) Under this condition Procedure Defective-Color(G, b, p, Δ), $b < \frac{1}{\epsilon}$, produces a $(c+\epsilon)(1+\epsilon)\frac{\Delta}{p}$-defective p-coloring in $\vartheta(p^2 + \alpha)$ time. Denote $C = (c+\epsilon)(1+\epsilon)$.

Let $\eta > 0$ be arbitrarily small positive constant. Our objective at this point is to derive an algorithm that employs at most $\Delta^{1+\eta}$ colors and runs in $O(\log \Delta \cdot \alpha)$ time.

Consider again one-level variant $\mathcal{A}_1(\Delta, p)$ of the algorithm that was described above. Specifically, $\mathcal{A}_1(\Delta, p)$ computes (via Procedure Defective-Color) a $(c+\epsilon)(1+\epsilon)\frac{\Delta}{p}$-defective p-coloring ψ of the input graph G. This coloring defines a partition $G_1, G_2, ..., G_p$, with maximum degree $\Delta(G_i) \le \Delta' = (c+\epsilon)(1+\epsilon)\frac{\Delta}{p}$. On each of these subgraphs, $\mathcal{A}_1(\Delta, p)$ invokes the $(\Delta'+1)$-coloring algorithm $\mathcal{A}_0(\Delta')$, and merges the colorings.

Denote by $N_1(\Delta, p)$ the overall number of colors that the algorithm $\mathcal{A}_1(\Delta, p)$ employs. Then $N_1(\Delta, p) \le p \cdot (\Delta'+1) = p \cdot (\frac{\Delta}{p}(c+\epsilon)(1+\epsilon)+1) \le \Delta((c+\epsilon)(1+\epsilon)+\epsilon) = \Delta(C+\epsilon)$.
Denote by $T_1(\Delta, p)$ the running time of $\mathcal{A}_1(\Delta, p)$. Then

$$T_1(\Delta, p) \le \vartheta \cdot (p^2 + \alpha) + \vartheta \cdot \Delta' + \alpha$$
$$\le \alpha + \vartheta(p^2 + \alpha + C \cdot \frac{\Delta}{p}). \quad (1)$$

For $i = 1, 2, ...$, consider the following recursive algorithm $\mathcal{A}_{i+1}(\Delta, p)$. This algorithm starts (exactly as $\mathcal{A}_1(\Delta, p)$) with computing a $C \cdot \frac{\Delta}{p}$-defective p-coloring ψ of G. This coloring defines the subgraphs $G_1, G_2, ..., G_p$ as above, with maximum degrees at most $\Delta' = C \cdot \frac{\Delta}{p}$. On each of these subgraphs we invoke the algorithm $\mathcal{A}_i(\Delta', p)$ (instead of $\mathcal{A}_0(\Delta')$), and merge the colorings. The merging is done in the same way as was described above.

Lemma 43. *For $i = 1, 2, ...,$ the algorithm $\mathcal{A}_i(\Delta, p)$ produces a legal $N_i(\Delta, p)$-coloring within $T_i(\Delta, p)$ time, where*
$N_i(\Delta, p) \le \Delta \cdot (C+\epsilon) \cdot C^{i-1}$,
$T_i(\Delta, p) \le \alpha + \vartheta(i \cdot (p^2 + \alpha) + (C^i \cdot \Delta/p^i))$.

Proof The proof is by induction on i. The induction base $(i = 1)$ was proven above. Next we prove the induction step.

The number of colors $N_{i+1}(\Delta, p)$ that the algorithm $\mathcal{A}_{i+1}(\Delta, p)$ employs is given by
$N_{i+1}(\Delta, p) \le p \cdot N_i(\Delta', p) \le p \cdot \Delta' \cdot (C+\epsilon) \cdot C^{i-1} = \Delta \cdot (C+\epsilon) \cdot C^i$.

The running time of $\mathcal{A}_{i+1}(\Delta, p)$ is given by
$T_{i+1}(\Delta, p) \le \vartheta(p^2+\alpha) + T_i(\Delta', p) \le \vartheta(p^2+\alpha) + \alpha + \vartheta(i(p^2+\alpha) + (C^i \cdot \Delta'/p^i) \le \alpha + \vartheta((i+1)(p^2+\alpha) + C^{i+1} \cdot \Delta/p^{i+1})$. This completes the proof. □

Set $i = \log_{p/C} \Delta$. It follows that $T_i(\Delta, p) \le \alpha + \vartheta(\log_{p/C} \Delta \cdot (p^2+\alpha) + 1)$, and $N_i(\Delta, p) \le \Delta \cdot \frac{C+\epsilon}{C} \cdot C^{\frac{\log \Delta}{\log p/C}} = \Delta(1+\frac{\epsilon}{C}) \cdot \Delta^{\frac{\log C}{\log p - \log C}}$.

Let δ, $0 < \delta < \eta$, be a small constant. Set p so that $\frac{\log C}{\log p - \log C} < \delta$, i.e., $p > C^{1+1/\delta} = ((c+\epsilon)(1+\epsilon))^{1+1/\delta}$. It follows that $N_i(\Delta, p) < \Delta^{1+\delta}(1+\frac{\epsilon}{C}) \le \Delta^{1+\eta}$, and $T_i(\Delta, p) = O(\log^* n \cdot \log \Delta)$. (Recall that $C^{1+1/\delta} < p = O(1)$.) To summarize:

Theorem 44. *Given a graph G with maximum degree Δ and $I(G) \le c = O(1)$ and an arbitrary small constant $\eta > 0$, the algorithm $\mathcal{A}_i(\Delta, p)$ with i and p set as was described above provides a $\Delta^{1+\eta}$-coloring in time $O(\log^* n \cdot \log \Delta \cdot c^{O(1/\eta)}) = O(\log^* n \cdot \log \Delta)$.*

Another notable point on the tradeoff curve is that one can get a $\Delta^{1+o(1)}$-coloring in $O((\log \Delta)^{1+\eta} \cdot \log^* n)$ time, for an arbitrarily small constant $\eta > 0$. To achieve this, set $p = \log^{\eta/2} \Delta$ (and $i = \log_{p/C} \Delta$). This result now follows directly from Lemma 43.

Observe also that the result of Theorem 42 (with somewhat inferior constants) can also be achieved from Lemma 43 by substituting $p = \Delta^{\eta/2}$, for an arbitrarily small constant $\eta > 0$.

4.2 An improved version

In this section we show that our algorithms can be modified to guarantee running time of $O(\log \Delta \cdot \frac{\log^* \Delta}{\log(\log^* \Delta)}) + \frac{1}{2} \log^* n$, while maintaining the same bound of $\Delta^{1+\eta}$ on the number of colors. We do so in two steps. First, we prove a bound of $O(\log \Delta \log^* \Delta + \log^* n)$, and then we improve it further.

The change that we introduce to our algorithms is the following one. Before invoking Procedure Legal-Color we invoke Linial's algorithm (see Lemma 21(1)) for computing a legal $O(\Delta^2)$-coloring ρ of the original graph G. This invocation requires $O(\log^* n)$ time. This auxiliary coloring will help us to execute the algorithm $\mathcal{A}_i(\Delta, p)$ faster. Specifically, we will use the following result of [18].

Theorem 45. *[Theorem 4.9 in [18]]: Given a d'-defective M-coloring of a graph G of maximum degree Δ and a defect parameter d, $d' \leq d < \Delta$, a d-defective $O((\frac{\Delta - d'}{d+1-d'})^2)$-coloring of G can be computed in $O(\log^* M)$ rounds.*

We will use this theorem with $d' = 0$. In this case the resulting coloring is a d-defective $O((\frac{\Delta}{d})^2)$-coloring. Given a parameter p, set $d = \lfloor \Delta/p \rfloor$. We get a $\lfloor \Delta/p \rfloor$-defective $O(p^2)$-coloring, in $O(\log^*(\Delta^2)) = O(\log^* \Delta)$ time. Our algorithm invokes Procedure Defective-Color in each recursive invocation of the algorithm $\mathcal{A}_i(\Delta, p)$. Procedure Defective-Color, in turn, computes a $\left\lfloor \frac{\Delta}{b \cdot p} \right\rfloor$-defective $O((b \cdot p)^2)$-coloring φ on line 1 of Algorithm 1, using Lemma 21(3). This step requires $O(\log^* n)$ time. In the modified version of our algorithm we will use the algorithm of Theorem 45 with the auxiliary $O(\Delta^2)$-coloring ρ as its input, and with the parameter $d = \left\lfloor \frac{\Delta}{b \cdot p} \right\rfloor$. In this way we obtain the desired $\left\lfloor \frac{\Delta}{b \cdot p} \right\rfloor$-defective $O((b \cdot p)^2)$-coloring φ in $O(\log^* \Delta)$ additional time, instead of $O(\log^* n)$ time.

The analysis of running time $T'_i(\Delta, p)$ of the modified algorithm $\mathcal{A}'_i(\Delta, p)$ proceeds alone the same lines as the analysis of the original algorithm. Denote $\alpha' = \log^* \Delta$. Computing the auxiliary coloring ρ requires $\alpha + O(1) = \log^* n + O(1)$ time [20]. Denote by $T''_i(\Delta, p)$ the running time of $\mathcal{A}'_i(\Delta, p)$ given the auxiliary coloring ρ, and let $T'_i(\Delta, p) = T''_i(\Delta, p) + \alpha + O(1)$ be the total running time of $\mathcal{A}'_i(\Delta, p)$.

Each invocation of Procedure Defective-Color requires now at most $\vartheta(p^2 + \alpha')$ time, instead of $\vartheta(p^2 + \alpha)$. Hence

$$T''_1(\Delta, p) \leq \vartheta(p^2 + \alpha') + \vartheta\Delta' + \alpha \leq \alpha + \vartheta(p^2 + \alpha' + C \cdot \frac{\Delta}{p}). \quad (2)$$

More generally, we argue that

$$T''_i(\Delta, p) \leq \alpha + \vartheta(i \cdot (p^2 + \alpha') + C^i \cdot \Delta/p^i). \quad (3)$$

Similarly to the case of $T_i(\Delta, p)$ the proof is by induction on i. The base of the induction is (2). The induction step follows from the inequality
$T''_{i+1}(\Delta, p) \leq \vartheta(p^2 + \alpha') + T''_i(\Delta', p)$.
Set $i = \log_{p/C} \Delta$ (where $C = (c + \epsilon)(1 + \epsilon)$) and $p > C^{1+1/\delta}$ (exactly as for the original algorithm $\mathcal{A}_i(\Delta, p)$.) We conclude that $T''_i(\Delta, p) \leq \alpha + O(\log^* \Delta \cdot \log \Delta)$. Hence
$T'_i(\Delta, p) \leq O(\alpha) + T''_i(\Delta, p) = O(\log^* n + \log^* \Delta \cdot \log \Delta)$.

This can be further improved by setting $p = \sqrt{\log^* \Delta}$ (and $i = \log_{p/C} \Delta$). Now (3) implies

$$T''_i \leq \alpha + \vartheta(\frac{\log \Delta}{\log(\log^* \Delta)} \cdot \log^* \Delta + 1) = O(\log^* n + \log \Delta \cdot \frac{\log^* \Delta}{\log(\log^* \Delta)}).$$

Hence
$T'_i(\Delta, p) \leq O(\alpha) + T''_i(\Delta, p) = O(\log^* n + \log \Delta \cdot \frac{\log^* \Delta}{\log(\log^* \Delta)})$ as well.

Moreover, the number of colors also improves: $N'_i(\Delta, p) \leq \Delta(1 + \frac{\epsilon}{C})C^i = \Delta(1 + \frac{\epsilon}{C})C^{\frac{\log \Delta}{\log p - \log C}} = \Delta(1 + \frac{\epsilon}{C})\Delta^{\frac{\log C}{\log p - \log C}} = \Delta^{1 + O(1/\log(\log^* \Delta))} = \Delta^{1+o(1)}$.

Observe also that by substituting $p = \log^{\eta/2} \Delta$, for an arbitrarily small constant $\eta > 0$, we obtain $T'_i(\Delta, p) = O(\alpha) + O(\log^{1+\eta} \Delta) = O(\log^* n + \log^{1+\eta} \Delta)$, and the number of colors becomes $\Delta \cdot 2^{O(\frac{\log \Delta}{\log \log \Delta})}$.

Note also that the analysis above implies that the leading constant factor in front of $\log^* n$ is just 2. Specifically, one $\log^* n$ term is required for computing the auxiliary coloring ρ, and another $\log^* n$ term is used for computing $(\Delta' + 1)$-colorings of subgraphs with maximum degree at most Δ' at the bottom level of the recursion. However, a close inspection of the algorithm of [4] reveals that it spends $\log^* n$ time for computing an $O(\Delta'^2)$-coloring ρ', and then spends another $O(\Delta')$ time to convert it into a $(\Delta' + 1)$-coloring. On the other hand, given an auxiliary $O(\Delta^2)$-coloring ρ, one can convert it into an $O(\Delta'^2)$-coloring ρ' of a subgraph with maximum degree at most Δ' in $O(\log^* \Delta)$ time. (This can be done using Linial's algorithm [20].) Hence the running time of our algorithm for computing $\Delta^{1+o(1)}$-coloring reduces to $O(\log \Delta \frac{\log^* \Delta}{\log(\log^* \Delta)}) + \log^* n$. Moreover, one can compute the auxiliary coloring ρ using an algorithm of Szegedy and Vishwanathan [25] (instead of Linial's algorithm [20]) in $\frac{1}{2} \log^* n$ time. This leads to the bound of $O(\log \Delta \frac{\log^* \Delta}{\log(\log^* \Delta)}) + \frac{1}{2} \log^* n$ on the running time of our algorithm. These considerations enable one to replace $O(\log^* n)$ by $\frac{1}{2} \log^* n$ also in Theorem 42. To summarize:

Theorem 46. *For any constant $\epsilon > 0$, and a graph G with bounded neighborhood independence,*
(1) an $O(\Delta)$-coloring of G can be computed in $O(\Delta^\epsilon) + \frac{1}{2} \log^* n$ time.
(2) a $\Delta^{1+o(1)}$-coloring of G can be computed in $O(\log \Delta \frac{\log^* \Delta}{\log(\log^* \Delta)}) + \frac{1}{2} \log^* n$ time.
(3) a $\Delta \cdot 2^{O(\frac{\log \Delta}{\log \log \Delta})}$-coloring of G can be computed in $O((\log \Delta)^{1+\epsilon}) + \frac{1}{2} \log^* n$ time.

It is easy to see that the chromatic number $\chi(G)$ of a graph G with $I(G) \le c$ satisfies $\chi(G) \ge \frac{\Delta}{c}$. Therefore, the results of Theorem 46 can be interpreted as approximation algorithms for the vertex-coloring problem on graphs of bounded neighborhood independence. Specifically, the 1st (respectively, 2nd; 3rd) statement of Theorem 46 shows that an $O(1)$-approximation (resp., $\Delta^{o(1)}$-approximation, resp. $2^{O(\frac{\log \Delta}{\log\log \Delta})}$-approximation) for this problem can be computed in $O(\Delta^{\epsilon}) + \frac{1}{2}\log^* n$ (resp., $O(\log \Delta \frac{\log^* \Delta}{\log(\log^* \Delta)}) + \frac{1}{2}\log^* n$; $O((\log \Delta)^{1+\epsilon}) + \frac{1}{2}\log^* n$) time.

5 Legal Edge Coloring in General Graphs

In this section we show that the techniques described in Sections 3 and 4 can be used to devise very efficient edge coloring algorithms for *general* graphs. First, observe that every line graph is claw-free, hence its neighborhood independence is at most 2. We provide the proof of this well-known fact for completeness.

Lemma 51. *For a graph $G = (V, E)$, the line graph $L(G)$ has neighborhood independence bounded by 2.*

Proof Let u be a vertex in $L(G)$ with at least three neighbors. Let $e_u \in E$ be the corresponding edge of u. Let v, w, x be any three neighbors of u in $L(G)$. Let e_v, e_w, e_x be the corresponding edges in E, respectively. Each edge e_v, e_w, e_x share a common endpoint with e_u. Therefore, at least two edges among e_v, e_w, e_x share a common endpoint. Suppose, without loss of generality, that these are e_v and e_w. Then, the vertices v and w are neighbors in $L(G)$. Hence the neighborhood $\Gamma(u)$ of u does not contain three independent vertices. Consequently, the neighborhood independence of $L(G)$ is at most 2. □

Observe also that Lemma 51 extends directly to line graphs of general r-hypergraphs. Specifically, for any hypergraph $\mathcal{H}$, the neighborhood independence of the line graph $L(\mathcal{H})$ is at most r. It follows that our results for graphs of bounded neighborhood independence (Theorem 46) apply to line graphs of r-hypergraphs, for any constant positive integer r.

Recall that for any graph G and positive integer k, a legal k-coloring of *vertices* of $L(G)$ is a legal k-coloring of *edges* of G, and vice versa. Recall also that the maximum degree $\Delta(L(G))$ of the line graph $L(G)$ satisfies $\Delta(L(G)) \le 2(\Delta - 1)$, where $\Delta = \Delta(G)$. Consequently, if we are given a line graph $L(G)$ of a graph G with $\Delta(G) = \Delta$, our algorithm can compute an $O(\Delta(L(G))) = O(\Delta)$-vertex-coloring of $L(G)$ in $O(\Delta^{\epsilon}) + \frac{1}{2}\log^* n$ time, for any constant $\epsilon > 0$. Similarly, one can also compute $\Delta^{1+o(1)}$-vertex-coloring (respectively, $\Delta \cdot 2^{O(\frac{\log \Delta}{\log\log \Delta})}$-vertex-coloring) of $L(G)$ in $O(\log \Delta \frac{\log^* \Delta}{\log(\log^* \Delta)}) + \frac{1}{2}\log^* n$ (resp., $(\log \Delta)^{1+\zeta} + \frac{1}{2}\log^* n$) time, for $\zeta > 0$ being an arbitrarily small positive constant. These vertex colorings give rise directly to edge coloring of G with the same number of colors.

On the other hand, in the distributed edge-coloring problem we are given as input the graph G, rather than its line graph $L(G)$. Nevertheless, one can simulate the distributed computation of an algorithm on $L(G)$ using the network $G = (V, E)$. To this end, each vertex of $L(G)$ is simulated by one endpoint of an appropriate edge in G. Consequently, a message sent over an edge of $L(G)$ will be sent over at most two edges in the simulation on G.

Lemma 52. *Any algorithm with running time T for the line graph $L(G)$ of the input graph G, can be simulated by G, and requires at most $2T + O(1)$ time.*

Proof For each edge $e \in E$, one of the endpoints of e simulates a vertex in $L(G)$ correspond to e. (Hence, each vertex in G may simulate many vertices of $L(G)$.) Specifically, for each edge $e = (u, v) \in E$, such that $\mathrm{Id}(u) < \mathrm{Id}(v)$, the vertex that corresponds to e in $L(G)$ is simulated by u. We denote the vertex in $L(G)$ that corresponds to e by v_e. The Id of v_e is set as the ordered pair $\langle \mathrm{Id}(u), \mathrm{Id}(v) \rangle$. This guarantees unique Ids for vertices in $L(G)$. Sending a message from a vertex w in $L(G)$ to its neighbor w' is simulated as follows. If the vertices that simulate w and w' are neighbors in G, the message is sent directly. Otherwise, the distance between the simulating vertices is 2. The vertices w and w' correspond to edges e and e' in E that share a common endpoint v'. In this case, the message is sent from the vertex that simulates w to v', and from v' to the vertex that simulates w. Hence, any algorithm for the line graph can be simulated on the original graph, increasing the running time by a factor of at most 2. The additive term of $O(1)$ in the running time above reflects the time spent for computing unique Ids. □

We apply Lemma 52 in conjunction with our results for vertex-coloring of $L(G)$, and obtain the following theorem.

Theorem 53. *For a graph $G = (V, E)$ with maximum degree Δ, and positive arbitrarily small constants $\epsilon, \zeta > 0$, our algorithm computes:* ***(1)*** *$O(\Delta)$-edge-coloring of G in $O(\Delta^{\epsilon}) + \frac{1}{2}\log^* n$ time,*
(2) *$\Delta^{1+o(1)}$-edge-coloring of G in $O(\log \Delta \frac{\log^* \Delta}{\log(\log^* \Delta)}) + \frac{1}{2}\log^* n$ time,*
(3) *$\Delta \cdot 2^{O(\frac{\log \Delta}{\log\log \Delta})}$-edge-coloring of G in $O((\log \Delta)^{1+\zeta}) + \frac{1}{2}\log^* n$ time.*

We remark that similarly to the situation with Theorem 46, Theorem 53 can also be interperted as an approximation for the edge-coloring problem. (Note that the chromatic index $\chi'(G)$, i.e., the minimum number of colors required to color legally the edges of G, is at least Δ.)

By Theorem 42 (using $c = 2$) we also get the following corollary.

Corollary 54. *For any constant $i \in \{1, 2, ...\}$, and any arbitrarily small $\epsilon > 0$, a $((2^i + \epsilon) \cdot (2\Delta - 1))$-edge-coloring of a graph with maximum degree Δ can be computed in $O(\Delta^{\frac{2}{2+i}} + \log^* n)$ time.*

Despite its simplicity and generality, the simulation technique that we described above has some downsides. Most notably, it causes the algorithm to send up to Δ messages through a single edge in a single round. Consequently, the resulting algorithm requires message size of $O(\Delta \log n)$. In what follows we present edge-coloring variant of our algorithm from Sections 3 and 4. This algorithm provides nearly the same bounds as in Theorem 53, but does so using much shorter messages of size $O(\log n)$. Specifically, the algorithm of Theorem 53 requires $O(\log \Delta \cdot \frac{\log^* \Delta}{\log(\log^* \Delta)}) + \frac{1}{2} \log^* n$ time to compute a $\Delta^{1+\eta}$ coloring for an arbitrarily small constant $\eta > 0$. A new version that we will now present does not have the slack factor of $\frac{\log^* \Delta}{\log(\log^* \Delta)}$, but instead spends $O(\log \Delta) + \log^* n$ time for this task. On the other hand, the $\frac{1}{2} \log^* n$ term in all the three cases will become $\log^* n$.

One ingredient of our edge-coloring algorithm is the following simple routine, due to Kuhn [18], for computing defective edge coloring in $O(1)$ time. We describe it for the sake of completeness. Let p', $1 \leq p' \leq \Delta$, be a parameter. Each vertex v labels the edges $e_1, e_2, ..., e_{deg(v)}$ that are incident to v with labels in $\{1, 2, ..., p'\}$, so that there is no label that is assigned to more than $\lceil \Delta/p' \rceil$ edges. (For example, it can label the edges $e_1, e_2, ...,$ $e_{\lceil \Delta/p' \rceil}$ with 1, $e_{\lceil \Delta/p' \rceil + 1}, e_{\lceil \Delta/p' \rceil + 2}, ..., e_{2 \cdot \lceil \Delta/p' \rceil}$ with 2, etc'.) For each edge $e = (u, w)$, both endpoints u and w send to each other the labels $\ell_u(e)$ and $\ell_w(e)$ that they assigned to the edge e. Finally, they set the colors $\varphi(e)$ of e to be equal to the pair $(\ell_u(e), \ell_w(e))$, where the order is determined by the identities of u and w. (For example, $\ell_u(e)$ appears before $\ell_w(e)$ if $\mathrm{Id}(u) < \mathrm{Id}(w)$, and after it otherwise.)

Obviously, the number of colors employed by φ is at most p'^2. To analyze its defect consider an edge $e = (u, w)$. Suppose without loss of generality that $\mathrm{Id}(u) < \mathrm{Id}(w)$. For an edge $e' = (u, z)$ with $\mathrm{Id}(u) < \mathrm{Id}(z)$ to get the same color as e, u should have label both e and e' with the same label. Hence there are at most $\lceil \Delta/p' \rceil$ incident edges $e' = (u, z)$ to $e = (u, w)$ with $\mathrm{Id}(u) < \mathrm{Id}(z)$ and $\varphi(e') = \varphi(e)$. For an edge $e' = (u, z)$ with $\mathrm{Id}(u) > \mathrm{Id}(z)$ to get the same color as e, the labels $\ell_u(e')$ and $\ell_w(e)$ must agree. However, u labels at most $\lceil \Delta/p' \rceil$ edges with the label $\ell_w(e)$, and so there are at most $\lceil \Delta/p' \rceil$ incident edges $e' = (u, z)$ to $e = (u, w)$ with $\mathrm{Id}(u) > \mathrm{Id}(z)$ and $\varphi(e') = \varphi(e)$. Hence overall there are at most $2 \cdot \lceil \Delta/p' \rceil$ edges e' that share with e the vertex u that satisfy $\varphi(e') = \varphi(e)$. By symmetrical considerations there are at most $2 \cdot \lceil \Delta/p' \rceil$ edges e' that share with e the vertex w and satisfy $\varphi(e') = \varphi(e)$. Hence the overall defect of φ is at most $4 \cdot \lceil \Delta/p' \rceil$.

Corollary 55. *[18] For any parameter p', $1 \leq p' \leq \Delta$, a $4 \cdot \lceil \Delta/p' \rceil$-defective p'^2-edge-coloring can be computed in $O(1)$ time.*

Hence the $\lfloor \Lambda/(b \cdot p) \rfloor$-defective $O((b \cdot p)^2)$-edge- coloring that is required on line 1 of Algorithm 1 (Procedure Defective-Color) can also be computed in $O(1)$ time. (We remark that in our algorithm, $b \cdot p$ is never greater than $\Lambda/4$, and thus the factor 4 in the defect of the coloring provided by Corollary 55 can be essentially ignored.) Next, we analyze the edge-coloring variant of Procedure Defective-Color (Algorithm 1). In our implemenation, for each edge $e = (u, w)$, at all times both the endpoints u and w of e will maintain the current color of e. In line 1 of the procedure the $\lfloor \Lambda/(b \cdot p) \rfloor$-defective $O((b \cdot p)^2)$-coloring φ is computed using the routine that was described above. As a result of this routine (that requires $O(1)$ time), for each edge $e = (u, w)$, both u and w know $\varphi(e)$.

Consider the while-loop on lines 4 - 10 of Algorithm 1. For an edge $e = (u, w)$ with $\varphi(e) = i$ for some $i \in \{1, 2, ..., O((b \cdot p)^2)\}$, at some point its endpoints u and w recolor e, i.e., compute the ψ-color $\psi(e)$. To this end each of them needs to know the numbers $N_e(k) = |\{e' \in \Gamma(e) \mid \psi(e') = k, \varphi(e') < \varphi(e)\}|$, for every $k \in \{1, 2, ..., p\}$. ($\Gamma(e)$ is the set of the edges incident to e.) Define also $N_{e,u}(k) = |\{e' \ni u \mid e' \neq e, \psi(e') = k, \varphi(e') < \varphi(e)\}|$. Observe that $N_e(k) = N_{e,u}(k) + N_{e,w}(k)$. Since u (respectively, w) can compute $N_{e,u}(1), N_{e,u}(2), ..., N_{e,u}(p)$ (resp., $N_{e,w}(1), N_{e,w}(2)$, ..., $N_{e,w}(p)$) locally, it follows that for u (resp., w) to be able to compute $N_e(1), N_e(2), ..., N_e(p)$ it needs to receive from w (resp., u) the numbers $N_{e,w}(1), N_{e,w}, (2)$, ..., $N_{e,w}(p)$ (resp., $N_{e,v}(1), N_{e,v}, (2), ..., N_{e,v}(p)$). This requires sending p messages over each edge in each direction.

Hence Procedure Defective-Color invoked with parameters G, b, p, Λ can be implemented for computing $((\Lambda/(b \cdot p) + \Lambda/p) \cdot 2 + 2)$-defective p-coloring ψ in $O((b \cdot p)^2)$-time. (Observe that the neighborhood independence is $c = I(L(G)) = 2$, and that the $\log^* n$ term of the runnning time from Corollary 32 disappears because we use Kuhn's defective edge-coloring routine, which requires $O(1)$ time.) This implementation, however, requires to send $O(p)$ messages of size $O(\log n)$ each through edges of G. If one allows only short messages (i.e., of size $O(\log n)$), then the running time of the procedure grows to $O((b \cdot p)^2 \cdot p) = O(b^2 \cdot p^3)$. Observe that in both cases, as a result of this invocation, for every edge $e = (u, w)$, both endpoints u and w know the resulting φ-color $\varphi(e)$.

The edge-coloring analogue $\mathcal{B}_i(\Delta, p)$ of the algorithm $\mathcal{A}_i(\Delta, p)$ is very similar to the algorithm $\mathcal{A}_i(\Delta, p)$. There are only two differences. The first difference is that $\mathcal{B}_i(\Delta, p)$ invokes the edge-coloring variant of Procedure Defective-Color, which was described above, instead of the original Procedure Defective-Color. As we have seen, assuming that $p \cdot \log n$ bits can be sent through an edge within one round, the edge-coloring variant of Procedure Defective-Color requires $O((b \cdot p)^2) \le \vartheta \cdot p^2$ time, for some constant $\vartheta > 0$, and not $\vartheta(p^2 + \alpha)$ time. (The latter was our estimate on the running time of the original Procedure Defective-Color.) Otherwise, if just $O(\log n)$ bits can be sent through an edge within one round, the running time of the edge-coloring variant of Procedure Defective-Color is at most $\vartheta \cdot p^3$.

The second difference is in the bottom level of the recursion. Specifically, while the algorithm $\mathcal{A}_0(\Delta)$ invokes an algorithm from [4] or from [18] for computing a $(\Delta + 1)$-vertex coloring in at most $\vartheta \cdot \Delta + \frac{1}{2}\alpha$ time, the algorithm $\mathcal{B}_0(\Delta)$ invokes a $(2\Delta - 1)$-edge-coloring algorithm due to Panconesi and Rizzi [22]. The latter algorithm requires at most $\vartheta \cdot \Delta + \alpha$ time. (Observe that one could also use an algorithm from [4] of from [18] for edge-coloring. This, however, would incur an increase of factor of Δ in the message size.)

Next, we analyze the resulting edge-coloring variant of Procedure Legal-Color. The number of colors it employs can be bounded almost exactly in the same way as for the vertex-coloring variant of the procedure. The only difference is that in the base case we now use $2\Delta - 1$ colors instead of $\Delta + 1$, and as a result the constant factor of 2 propogates all the way through the analysis. Hence the resulting estimates grow by a factor of 2. The analysis of the running time changes slightly, and it depends on the size of messages that we are allowed to use.

First, consider the case that we are allowed to use p short messages over each edge on every round. The running time of $\mathcal{B}_1(\Delta, p)$ satisfies (see (1)) $T_1(\Delta, p) \le \vartheta \cdot p^2 + \vartheta \cdot \Delta' + \alpha \le \alpha + \vartheta(p^2 + C \cdot \frac{\Delta}{p})$. More generally, the inductive argument of Lemma 43 implies that $T_i(\Delta, p) \le \alpha + \vartheta(p^2 + C^i \cdot \Delta/p^i)$.

Let $\eta > 0$ be an arbitrarily small constant, and δ be a constant such that $0 < \delta < \eta$. Let $p > C^{1+1/\delta}$ be a constant, and $i = \log_{p/C} \Delta$. We obtain a $\Delta^{1+\eta}$-edge-coloring in $\alpha + O(\log \Delta) = O(\log \Delta) + \log^* n$ time. By setting $p = \log^{\eta/2} \Delta$ we get a $\Delta \cdot 2^{O(\frac{\Delta}{\log\log \Delta})}$-edge-coloring in $O(\log^{1+\eta} \Delta) + \log^* n$ time. Finally, by setting $p = \Delta^{\epsilon/2}$, for an arbitrarily small constant $\epsilon > 0$, we get an $O(\Delta)$-edge-coloring in $O(\Delta^\epsilon) + \log^* n$ time.

We remark, however, that the constants hidden by the O-notation in the $O(\Delta)$-edge-coloring variant of this algorithm can be improved by using the algorithm from the beginning of Section 4.1. Specifically, for any constant $i \in \{1, 2, ..., \}$ that algorithm produces a $(2^{i+1} + \eta) \cdot \Delta$-edge-coloring in $O(\Delta^{\frac{2}{2+i}}) + \log^* n$ time for an arbitrarily small constant $\eta > 0$. The latter algorithm employs, however, large messages. To improve the message size to $O(\log n)$ in that algorithm one needs to spend $O(p^3)$ time instead of $O(p^2)$ time for computing defective edge-coloring. As a result we obtain a $(2^{i+1} + \eta) \cdot \Delta$-edge-coloring in $O(\Delta^{\frac{3}{3+i}}) + \log^* n$ time, for any $i \in \{1, 2, ...\}$, for an arbitrarily small constant $\eta > 0$, using short messages.

The size of the messages that the algorithm employs allows sending p numbers $N_{e,u}(1), N_{e,u}(2), ..., N_{e,u}(p)$ in one message, where e is an edge and u is one of its endpoints. Each of these numbers is an integer between 1 and Δ, and so overall one needs $O(p \cdot \log \Delta)$ bits to encode them. Also, for implementing the algorithm of [22] on the bottom level of the recursion one needs messages of size $O(\log n)$. Hence the total message size is $O(\max\{p \cdot \log \Delta, \log n\})$, where the value of p depends on the number of colors we want to get. For $\Delta^{1+\eta}$-edge-coloring $p = O(1)$; for $\Delta \cdot 2^{O(\frac{\log \Delta}{\log\log \Delta})}$-edge-colorig $p = \log^{\eta/2} \Delta$, and for $O(\Delta)$-edge-coloring $p = \Delta^{\epsilon/2}$. Hence for $\Delta^{1+\eta}$-edge-coloring the message size is $O(\log n)$, as desired.

Next, consider the case that we are allowed only messages of size $O(\log n)$. Then, as we have seen, inside Procedure Defective-Color we need to send p messages over each edge $e = (u, w)$ for both endpoints u and w to be able to compute $N_e(1), N_e(2), ..., N_e(p)$, and requires $O(p)$ time. The overall running time of Procedure Defective-Color in this case is, as was already shown, at most $\vartheta \cdot p^3$. Hence the estimates of the running time become $T_1(\Delta, p) \le \alpha + \vartheta(p^3 + C \cdot \frac{\Delta}{p})$, and more generally, $T_i(\Delta, p) \le \alpha + \vartheta(p^3 + C^i \cdot \Delta/p^i)$. This results in the same (up to constant factors hidden by the O-notation) estimates on the running time for $\Delta^{1+\eta}$-edge-coloring, $\Delta \cdot 2^{O(\frac{\log \Delta}{\log\log \Delta})}$-edge-coloring, and $O(\Delta)$-edge-coloring. In the latter case, however, to get running time of $O(\Delta^\epsilon) + \log^* n$ one needs to set $p = \Delta^{\epsilon/3}$ rather than $p = \Delta^{\epsilon/2}$. To summarize:

Theorem 56. *For any graph G and positive arbitrary small constants $\epsilon, \eta, \zeta > 0$,*
(1) an $O(\Delta)$-edge-coloring of G can be computed in

$O(\Delta^{\epsilon}) + \log^* n$ time.
(2) an $O(\Delta^{1+\eta})$-edge-coloring of G can be computed in $O(\log \Delta) + \log^* n$ time.
(3) a $\Delta \cdot 2^{O(\frac{\log \Delta}{\log\log \Delta})}$-edge-coloring of G can be computed in $O((\log \Delta)^{1+\zeta}) + \log^* n$ time.
Moreover, the algorithm that computes these colorings employs messages of size $O(\log n)$.

The next theorem summarizes the tradeoffs we can achieve for $O(\Delta)$-edge-coloring, using large and short messages.

Theorem 57. *For any constant $i \in \{1, 2, ...\}$, for an arbitrarily small constant $\eta > 0$, using large (respectively, short) messages our algorithm produces a $(2^{i+1} + \eta)\Delta$-edge-coloring within $O(\Delta^{\frac{2}{2+i}}) + \log^* n$ (resp., $O(\Delta^{\frac{3}{3+i}}) + \log^* n$) time. By large (resp., short) messages we mean messages of size $O(\Delta^{\frac{1}{2+i}} \cdot \log n)$ (resp., $O(\log n)$).*

6 A Tradeoff

In this section we extend the results stated in Theorem 56, and obtain a much faster algorithm that uses much more colors. Specifically, for any monotonic non-decreasing function $g(\cdot)$, one can get an $O(\frac{\Delta^2}{g(\Delta)})$-vertex-coloring of graphs with bounded neighborhood independence in $O(\log g(\Delta) \cdot \frac{\log^* g(\Delta)}{\log(\log^* g(\Delta))}) + \frac{1}{2}\log^* n$ time. This result also translates into an $O(\frac{\Delta^2}{g(\Delta)})$-edge-coloring algorithm with running time $O(\log g(\Delta)) + \log^* n$. This extension is closely related to the vertex-coloring tradeoff presented in [5]. Specifically, there it is shown that one can produce an $O(\frac{\Delta^2}{g(\Delta)})$-vertex-coloring of a general graph in $O(\log n \cdot \log g(\Delta))$ time.

For a small positive constant $0 < \eta < 1$, set $q(\Delta) = g(\Delta)^{\frac{1}{1-\eta}}$, and $p = p(\Delta) = \frac{\Delta}{q(\Delta)}$. Compute a $\frac{\Delta}{p}$-defective $O(p^2)$-coloring φ of G in time $O(\log^* n)$ using Lemma 21(3) (an algorithm from [18]). This coloring partitions G into $O(p^2)$ vertex disjoint subgraphs $G_1, G_2, ..., G_{O(p^2)}$, each of maximum degree at most $\frac{\Delta}{p}$. Since the original graph G has neighborhood independence at most c, by Lemma 35 it follows that each of the subgraphs $G_1, G_2, ..., G_{O(p^2)}$ has neighborhood independence at most c as well.

On each of these subgraphs in parallel, invoke our algorithm whose properties are summarized in Theorem 46(2). Within $O(\log \frac{\Delta}{p} \cdot \frac{\log^*(\Delta/p)}{\log(\log^*(\Delta/p))}) + \frac{1}{2}\log^* n$ time we obtain an $O((\frac{\Delta}{p})^{1+\eta})$-vertex-coloring of each of these subgraphs. Overall we obtain an $O(p^2 \cdot (\frac{\Delta}{p})^{1+\eta}) = O(p^{1-\eta} \cdot \Delta^{1+\eta})$-vertex-coloring of G within $O(\log \frac{\Delta}{p} \cdot \frac{\log^*(\Delta/p)}{\log(\log^*(\Delta/p))} + \log^* n)$ time. Since $p = p(\Delta) = \frac{\Delta}{q(\Delta)}$, we have an $O(\frac{\Delta^2}{q(\Delta)^{1-\eta}})$-vertex-coloring within $O(\log q(\Delta) \cdot \frac{\log^* q(\Delta)}{\log(\log^* q(\Delta))} + \log^* n)$ time. Finally, since $g(\Delta) = q(\Delta)^{1-\eta}$, we derive an $O(\frac{\Delta^2}{g(\Delta)})$-vertex-coloring within $O(\log g(\Delta) \cdot \frac{\log^* g(\Delta)}{\log(\log^* g(\Delta))} + \log^* n)$ time.

One can also compute an auxiliary $O(\Delta^2)$-coloring ρ in the beginning of the computation in $\frac{1}{2}\log^* n$ time using the algorithm of Szegedy and Vishwanathan [25]. This coloring can be used to compute a $\frac{\Delta}{p}$-defective $O(p^2)$-coloring φ in just $O(\log^* \Delta)$ time. As we have seen in Section 4.2, it can also be used to eliminate the $\frac{1}{2}\log^* n$ term from the running time of the algorithm from Theorem 46. To summarize:

Corollary 61. *For any monotonic non-decreasing function $g(\cdot)$, one can compute an $O(\frac{\Delta^2}{g(\Delta)})$-vertex-coloring of a graph with bounded neighborhood independence in $O(\log g(\Delta) \cdot \frac{\log^* g(\Delta)}{\log(\log^* g(\Delta))}) + \frac{1}{2}\log^* n$ time.*

By the techniques that are described in Section 5 we can also $O(\frac{\Delta^2}{g(\Delta)})$-edge-color general graphs in $O(\log g(\Delta) \cdot \frac{\log^* g(\Delta)}{\log(\log^* g(\Delta))}) + \frac{1}{2}\log^* n$ time, and also in $O(\log g(\Delta)) + \log^* n$ time. The latter running time is achieved using only messages of size $O(\log n)$.

7 Conclusion and Open Questions

We showed that an $O(\Delta)$-edge-coloring can be computed in $O(\Delta^{\epsilon} + \log^* n)$ time, for an arbitrarily small $\epsilon > 0$. Specifically, a $((4+\epsilon)\Delta)$-edge-coloring can be computed in $O(\Delta^{2/3} + \log^* n)$ time, a $((8+\epsilon)\Delta)$-edge-coloring can be computed in $O(\Delta^{1/2} + \log^* n)$ time, etc'. With short messages our algorithm computes a $((4+\epsilon)\Delta)$-edge-coloring in $O(\Delta^{3/4} + \log^* n)$ time, a $((8+\epsilon)\Delta)$-edge-coloring in $O(\Delta^{3/5} + \log^* n)$ time, etc'. Improving this tradeoff is an interesting open problem. Another challenging problem is to obtain an $O(\Delta)$-edge-coloring algorithm that requires polylogarithmic time. Our algorithm constructs a $\Delta \cdot 2^{O(\frac{\log \Delta}{\log\log \Delta})}$-edge-coloring in polylogarithmic time.

Acknowledgements The authors thank Alessandro Panconesi for sending them several papers and addressing their questions.

References

1. B. Awerbuch, A. V. Goldberg, M. Luby, and S. Plotkin. Network decomposition and locality in distributed computation. In *Proc. of the 30th Symposium on Foundations of Computer Science*, pages 364–369, 1989.

2. J. Andrews, and M. Jacobson. On a generalization of a chromatic number. *Congressus Numer*, 47:33-48, 1985.
3. L. Barenboim, and M. Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. In *Proc. of the 27th ACM Symp. on Principles of Distributed Computing*, pages 25–34, 2008.
4. L. Barenboim, and M. Elkin. Distributed $(\Delta+1)$-coloring in linear (in Δ) time. In *Proc. of the 41th ACM Symp. on Theory of Computing*, pages 111-120, 2009.
5. L. Barenboim, and M. Elkin. Deterministic distributed vertex coloring in polylogarithmic time. In *Proc. of the 29th ACM Symp. on Principles of Distributed Computing*, pages 410-419, 2010.
6. M. Chudnovsky, and P. Seymour. The structure of claw-free graphs. *Surveys in Combinatorics 2005, London Math Soc. Lecture Note Series*, 327:153-171, 2005.
7. L. Cowen, R. Cowen, and D. Woodall. Defective colorings of graphs in surfaces: partitions into subgraphs of bounded valence. *Journal of Grah Theory*, 10:187–195, 1986.
8. L. Cowen, W. Goddard, and C. Jesurum. Coloring with defect. In *Proc. of the 8th ACM-SIAM Symp. on Discrete Algorithms, New Orleans, Louisiana, USA*, pages 548–557, January 1997.
9. A. Czygrinow, M. Hanckowiak, and M. Karonski. Distributed $O(\Delta \log n)$-edge-coloring algorithm. In *Proc. of the 9th Annual European Symposium on Algorithms*, pages 345-355, 2001.
10. D. Dubhashi, D. Grable, and A. Panconesi. Nearly-optimal, distributed edge-colouring via the nibble method. *Theoretical Computer Science*, 203:225–251, 1998.
11. D. Durand, R. Jain, and D. Tseytlin. Applying randomized edge coloring algorithms to distributed communication: an experimental study. In *Proc. of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 264–274, 1995.
12. B. Gfeller, and E. Vicari. A randomized distributed algorithm for the maximal independent set problem in growth-bounded graphs. In *Proc. of the 26th ACM Symp. on Principles of Distributed Computing*, pages 53–60, 2007.
13. D. Grable, and A. Panconesi. Nearly optimal distributed edge coloring in $O(\log \log n)$ rounds *Random Structures and Algorithms*, 10(3):385–405, 1998.
14. F. Harary, and K. Jones. Conditional colorability II: Bipartite variations. *Congressus Numer*, 50:205-218, 1985.
15. R. Jain, K. Somalwar, J. Werth, and J. C. Browne. Scheduling parallel I/O operations in multiple bus systems. *ELSEVIER Journal of Parallel and Distributed Computing*, 16(4):352-362, 1992.
16. F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the Locality of Bounded Growth. In*Proc. of the 24rd ACM Symp. on Principles of Distributed Computing*, pages 60–68, 2005.
17. K. Kothapalli, C. Scheideler, M. Onus, and C. Schindelhauer. Distributed coloring in $\tilde{O}(\sqrt{\log n})$ bit rounds. In *Proc. of the 20th International Parallel and Distributed Processing Symposium*, 2006.
18. F. Kuhn. Weak graph colorings: distributed algorithms and applications. In *proc. of the 21st ACM Symposium on Parallel Algorithms and Architectures*, pages 138–144, 2009.
19. F. Kuhn, and R. Wattenhofer. On the complexity of distributed graph coloring. In *Proc. of the 25th ACM Symp. on Principles of Distributed Computing*, pages 7–15, 2006.
20. N. Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.
21. N. Linial and M. Saks. Low diameter graph decomposition. *Combinatorica* 13: 441 - 454, 1993.
22. A. Panconesi, and R. Rizzi. Some simple distributed algorithms for sparse networks. *Distributed computing*, 14(2):97–100, 2001.
23. A. Panconesi, and A. Srinivasan. On the complexity of distributed network decomposition. *Journal of Algorithms*, 20(2):581-592, 1995.
24. A. Panconesi, and A. Srinivasan. Distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM J. on Computing*, 26(2):350–368, 1997.
25. M. Szegedy, and S. Vishwanathan. Locality based graph coloring. In *Proc. 25th ACM Symposium on Theory of Computing*, pages 201-207, 1993.
26. J. Schneider, and R. Wattenhofer. A log-star distributed Maximal Independent Set algorithm for Growth Bounded Graphs. In *Proc. of the 27th ACM Symp. on Principles of Distributed Computing*, pages 35–44, 2008.
27. J. Schneider, and R. Wattenhofer. A new technique for distributed symmetry breaking. To appear in *Proc. of the 29th ACM Symp. on Principles of Distributed Computing*, 2010.
28. V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret Analiz*, 3:25–30, 1964.