Deterministic Distributed Vertex Coloring: Simpler, Faster, and without Network Decomposition*

Mohsen Ghaffari
Dept. of Computer Science
ETH Zurich
8092 Zurich, Switzerland
Email: ghaffari@inf.ethz.ch

Fabian Kuhn
Dept. of Computer Science
University of Freiburg
79110 Freiburg, Germany
Email: kuhn@cs.uni-freiburg.de

Abstract—We present a simple deterministic distributed algorithm that computes a \((\Delta+1)\)-vertex coloring in \(O(\log^2(\Delta+1) \cdot \log n)\) rounds. The algorithm can be implemented with \(O(\log n)\)-bit messages. The algorithm can also be extended to the more general \((\text{degree}+1)\)-list coloring problem.

Obtaining a polylogarithmic-time deterministic algorithm for \((\Delta+1)\)-vertex coloring had remained a central open question in the area of distributed graph algorithms since the 1980s, until a recent network decomposition algorithm of Rozhoň and Ghaffari [STOC’20]. The current state of the art is based on an improved variant of their decomposition, which leads to an \(O(\log^2 n)\)-round algorithm for \((\Delta+1)\)-vertex coloring.

Our coloring algorithm is completely different and considerably simpler and faster. It solves the coloring problem in a direct way, without using network decomposition, by gradually rounding a certain fractional color assignment until reaching an integral color assignment. Moreover, via the approach of Chang, Li, and Pettie [STOC’18], this improved deterministic algorithm also leads to an improvement in the complexity of randomized algorithms for \((\Delta+1)\)-coloring, now reaching the bound of \(O(\log^2 \log n)\) rounds.

As a further application, we provide faster deterministic distributed algorithms for the following vertex coloring variants. In graphs of arboricity \(a\), we show that a \((2+\varepsilon)a\)-vertex coloring can be computed in \(O(\log^2 a \cdot \log n)\) rounds. We also show that for \(\Delta \geq 3\), \(\Delta\)-coloring of a \(\Delta\)-colorable graph \(G\) can be computed in \(O(\log^2 \Delta \cdot \log^2 n)\) rounds.

Keywords-distributed graph algorithms; distributed coloring; deterministic rounding; list coloring

I. INTRODUCTION

Graph coloring has been one of the central problems in the area of distributed graph algorithms for over three decades. See, e.g., the Distributed Graph Coloring book of Barenboim and Elkin [7]. In this paper, we present a surprisingly simple deterministic distributed algorithm that improves on the state of the art considerably, and also leads to a faster randomized distributed algorithm. We first review the model and the state of the art, and then state our contribution.

A. Background on the Coloring Problem

Model: We work with the standard synchronous message passing model of distributed algorithms. The network is abstracted as an \(n\)-node undirected graph \(G = (V,E)\), where each node represents one processor and has a unique \(O(\log n)\)-bit identifier. Initially, nodes do not know the topology of the network graph \(G\), except for each knowing its own neighbors. Besides this, the nodes might know some global parameters (or parameter estimates), such as the number of nodes \(n\) and the maximum degree \(\Delta\) (or suitable upper bounds on them). The communication between the processors/nodes happens in synchronous rounds, where per round each node can send one message to each of its neighbors in \(G\). The variant where the messages are allowed to be of unbounded size is known as the LOCAL model [28], [34] while the variant with bounded-size messages, usually \(O(\log n)\) bits, is referred to as the CONGEST model [34]. At the end, each processor should know its own part of the output, e.g., in the coloring problem, the color assigned to its node. The main complexity measure for the algorithms is the number of rounds until all nodes know their output.

State of the Art: Here, we give a brief overview. We defer a more exhaustive review of the related work to Section I-D.

The celebrated work of Luby [30] and Alon, Babai, and Itai [1] from the 1980s provide randomized distributed algorithms for \((\Delta+1)\)-coloring with round complexity \(O(\log n)\)—in fact, even for the harder problem of maximal independent set. However, obtaining a deterministic algorithm with a similar round complexity has remained elusive. In his seminal work [28], [29], and after discussing [1], [30], Linial wrote

“It is therefore particularly interesting to find out the best time complexity in terms of \(n\) for finding a \((\Delta+1)\)-coloring, and in particular whether polylogarithmic time suffices.”

The sentence, and especially the latter part, refers to deterministic algorithms. Shortly after, deterministic algorithms with round complexity \(2^{O(\sqrt{\log n \cdot \log \log n})}\) and then

*A full version is available at https://arxiv.org/abs/2011.04511 [18].
were presented by Awerbuch et al. [2] and Panconesi and Srinivasan [33], respectively. These algorithms are based on a generic tool known as network decomposition, where the graph is decomposed into clusters of small diameter, which are colored with few colors and thus can then be processed efficiently in the LOCAL model. However, the question of obtaining a poly(log n) time algorithm remained open for a long time. Indeed, Open Problem 11.3 in the influential 2013 book of Barenboim and Elkin on distributed graph coloring asked for an even weaker objective:

"Open Problem 11.3 Devise a $\Delta \cdot \text{poly}(\log(\Delta))$-coloring in deterministic polylogarithmic time.”

The first resolution of this question was presented recently: Rozhoň and Ghaffari [35] presented an $O(\log^2 n)$ round deterministic algorithm in the LOCAL model that solves the more general network decomposition problem and this led to an $O(\log^2 n)$ round deterministic algorithm for $(\Delta + 1)$-coloring. An improved variant of this network decomposition algorithm was presented more recently by Grunau, Ghaffari, and Rozhoň [17], which for $(\Delta + 1)$-coloring implies a $O(\log^5 n)$ round algorithm. Both of these coloring algorithms use large messages, as they end up gathering the topology around some node. Bamberger, Kuhn, and Maus [4] resolved this issue by presenting a CONGEST-model coloring algorithm for low-diameter graphs. This is inspired by the derandomization approach of Luby [31] and Censor-Hillel et al. [11] of reducing the amount of randomness to $O(\log^2 n)$ bits and then fixing the bits one by one, via pessimistic estimators and global communication. Put together with the faster network decomposition algorithm [17], this gives an $O(\log^5 n)$ round deterministic algorithm for $(\Delta + 1)$-coloring. This is the state of the art deterministic algorithm for $(\Delta + 1)$-coloring in the CONGEST model.

On the side of randomized algorithms, there has also been much progress (as we shall review later in Section I-D). Interestingly, the state of the art in the LOCAL model is an $O(\log^5 \log n)$-round algorithm that follows from plugging the deterministic algorithm mentioned above into the randomized framework of Chang et al. [12]. It is also known that any improvement on the randomized complexity requires (and would imply) an improvement on the deterministic complexity [13].

B. Our Contribution

We present a surprisingly simple algorithm that solves the $(\Delta + 1)$-vertex coloring directly, without using the recent breakthrough results on network decomposition (or their ideas) [17], [35]. Besides being considerably simpler than the state of the art, the algorithm is also quadratically faster and runs in $O(\log^2 \Delta \log n)$ rounds, with small $O(\log n)$-bit messages. It also extends to the list coloring generalization of the problem.

**Theorem 1.1 (Informal).** There is a deterministic distributed algorithm that computes a $(\Delta + 1)$-coloring in any graph with at most $n$ nodes and maximum degree at most $\Delta$ in $O(\log^2 \Delta \log n)$ rounds, and using $O(\log n)$ bit messages.

The algorithm can be generalized to the $(\text{degree} + 1)$-list coloring problem where each node $v$ should choose its color from a prescribed list of colors $L_v \subseteq \{1, ..., \ell\}$ with size $|L_v| \geq d(v) + 1$, where $d(v)$ denotes the degree of node $v$. Then, the complexity is $O(\log^2 \Delta \log n)$ rounds if we can use $O(\Delta \cdot \text{log } \ell)$-bit messages, or $O(\log^{17} \Delta \cdot \log n)$ rounds using $O(\log \ell)$-bit messages.

Besides its simplicity and faster round complexity, our result is qualitatively different than prior work. For instance, using the ingredients of Theorem 1.1, we can color a $1 - \epsilon$ fraction of nodes for an arbitrarily small constant $\epsilon > 0$ in $O(\log^2 \Delta + \log n)$ rounds of the LOCAL model. Notably, this is nearly-independent of the network size $n$, and has only logarithmic dependencies on the maximum degree $\Delta$. In contrast, with the previous algorithm that follows from [17], [35], coloring even just an $\epsilon$-fraction of the nodes requires $\Theta(\log^4 n)$ rounds. A similar difference extends to the results in the CONGEST model.

Moreover, our approach provides a new useful structural understanding of the coloring problem, which we hope might find applications in other computational settings. The previous method was based on network decomposition [17], [35] and essentially relied on breaking the graph into low-diameter parts and then solving the coloring in each part by gathering the entire topology. In that sense, those methods, despite providing an efficient solution of coloring in the distributed model, did not provide any new structural understanding of the coloring problem. As we will outline in Section I-C, our approach is completely different and it allows us to cast the $(\Delta + 1)$-coloring problem as a few instances of a simple and clean rounding problem, which gradually turn fractional color assignments into integral color assignments, while approximately maintaining a simple quality measure. We believe that this structure is of independent interest and we are hopeful that it will find applications in other computational settings.

**Other implications, randomized coloring:** By plugging our deterministic list-coloring algorithm into the randomized coloring algorithm of Chang et al. [12], we can also improve the randomized complexity from the $O(\log^5 \log n)$ bound of [17] to $O(\log^4 \log n)$:

**Corollary 1.2.** There is a randomized algorithm in the LOCAL model that computes a $(\Delta + 1)$-coloring in any graph with at most $n$ nodes and maximum degree at most
\[ \Delta \text{ in } O(\log^3 \log n) \text{ rounds, with high probability}^{1}. \]

We note that in a very recent paper, Halldórsson, Nolin, and Tonoyan [21] give an improved randomized CONGEST algorithm for \((\Delta + 1)\)-coloring. Their algorithm can use the deterministic CONGEST algorithm that we give in Theorem I.1 and by applying our result, they show that \((\Delta + 1)\)-coloring can also be solved in \(O(\log^3 \log n)\) rounds in the randomized CONGEST model.

Other implications, coloring low-arboricity graphs and \(\Delta\)-coloring: We also provide a variant of Theorem I.1 with node weights — e.g., coloring a subset of nodes with a constant fraction of the weights in \(O(\log^2 \Delta)\) rounds of the LOCAL model and \(O(\log^2 C)\) rounds of the CONGEST model — and show that this leads to improvements for coloring graphs of low arboricity and for \(\Delta\)-coloring. In particular, in graphs of arboricity \(\alpha\), we show that a \((2 + \varepsilon)\alpha\)-vertex coloring can be computed deterministically in \(O(\log^3 \Delta \cdot \log n)\) rounds. We also show that for \(\Delta \geq 3\), a \(\Delta\)-coloring of a \(\Delta\)-colorable graph \(G\) can be computed deterministically in \(O(\log^2 \Delta \cdot \log^2 n)\) rounds. The details are omitted from this extended abstract. They appear in the full version of the paper [18].

C. Our Method in a Nutshell

We start by giving a high-level description of our algorithm for the LOCAL model. The algorithm is based on the most basic randomized coloring method where each node repeatedly tries to get colored by choosing a uniformly random available color. More precisely, consider an edge \(\{u, v\}\) between two nodes \(u\) and \(v\) of degree \(d(u)\) and \(d(v)\) and assume that \(u\) and \(v\) have color lists \(L_u\) of size \(|L_u| > d(u)\) and \(L_v\) of size \(|L_v| > d(v)\), respectively. If \(u\) chooses a uniformly random color from \(L_u\) and \(v\) picks a uniformly random color from \(L_v\), the probability that both nodes pick the same color is less than \(1/\max\{d(u) + 1, d(v) + 1\}\).

If every node picks a uniformly random color from its list, the expected number of monochromatic edges is less than \(n\). If we are given a coloring with \(O(n)\) monochromatic edges, it is simple to find a subset \(S\) of nodes of size \(\Theta(n)\) such that the induced subgraph \(G[S]\) is properly colored.\(^2\) The main goal of our paper is to develop an efficient deterministic distributed algorithm to assign a color \(x_u \in L_u\) to each node \(u\) such that the number of monochromatic edges is \(O(n)\) and we can thus color a constant fraction of the nodes. Repeating \(O(\log n)\) times then solves the problem of coloring all nodes of the graph.

Our general approach is based on the idea of rounding fractional assignments to integral assignments. We note that rounding ideas have been used successfully in the past for the maximal matching problem [14]. In the case of the coloring problem, we start with a fractional assignment of colors to the nodes and we gradually round this fractional solution to obtain an assignment of a single color to each node. For a node \(u\), a fractional color assignment can just be thought of as a probability distribution over the colors in \(u\)’s list \(L_u\). We define the cost of a fractional assignment as the expected number of monochromatic edges in the setting where each node \(u\) independently picks a color according to this distribution. We say that a fractional color assignment to a node \(u\) is \(1/Q\)-fractional if each color \(c \in L_u\) is assigned to \(u\) with value \(a/Q\) for some integer \(a \in \{0, \ldots, Q\}\). Initially, each node \(u\) computes an initial \(1/\Theta(d(u))\)-integral color assignment in which each color of \(L_u\) obtains approximately equal values. As observed above, the total cost of such an assignment is \(O(n)\). Now, assume that we are given a \(1/(2Q)\)-integral fractional color assignment. We want to turn this color assignment into an \(1/Q\)-integral fractional color assignment such that the expected number of monochromatic edges (essentially) does not increase. If the fractional assignments of nodes are rounded one node at a time, each node can simply decide which half of the \(1/(2Q)\)-fractional values to round up by \(1/(2Q)\) and which half of those values to round down by \(1/(2Q)\) so that the expected cost of its edges does not increase. This can be parallelized in the following way. The rounding decisions of nodes that are non-adjacent do not depend on each other. If we are given a coloring of \(G\) with \(C\) colors, nodes of the same color class can thus be rounded in parallel and obtain an \(1/Q\)-integral assignment in \(O(C)\) rounds. In order for this to be useful, we however need to speed up this process significantly.

This is where we use the fact that we only do gradual rounding steps. The rounding from \(1/(2Q)\)-integral values to \(1/Q\)-integral values guarantees that each fractional value increases at most by a factor 2. Therefore, the probability of an edge becoming monochromatic can increase at most by a factor 4, even if the rounding is done in a worst-case way. We define edge weights \(w(e)\), where \(w(e)\) is the probability of the edge becoming monochromatic. By using a defective coloring algorithm of [24, 25], for any parameter \(\varepsilon > 0\), we can efficiently compute an \(O(1/\varepsilon^2)\)-coloring of \(G\) such that the total weight of monochromatic edges is at most an \(\varepsilon\)-fraction of the total weight of all the edges. In Section V, we further show that because we only need to bound the overall weight of monochromatic edges, we can even reduce the number of colors to \(O(1/\varepsilon)\).

Let \(E'\) be the set of monochromatic edges of this defective coloring. Because the edges in \(E'\) only have an \(\varepsilon\)-fraction of the total weight of all edges, even if nodes round their fractional values in a worst-case way, the total cost of the edges in \(E'\) is still at most a \(4\varepsilon\)-fraction of the overall cost of
the original fractional assignment. On the remaining graph \( G' = (V, E \setminus E') \), we have a proper \( O(1/\varepsilon) \)-coloring and we can therefore round the fractional values in time \( O(1/\varepsilon) \) without loss in the quality of the fractional solution. Overall, we can get from a \( 1/(2Q) \)-integral fractional assignment to a \( 1/Q \)-fractional integral assignment in time \( O(1/\varepsilon) \) and at the cost of losing a \( 1 + O(\varepsilon) \)-factor in the overall cost. By choosing \( \varepsilon = 1/\log \Delta \), we can do \( O(\log \Delta) \) rounding steps to get from a \( 1/O(\Delta) \)- integral fractional assignment to an integral assignment such that the total number of monochromatic edges only increases by a factor of \( (1 + O(\varepsilon))^{O(\log \Delta)} = O(1) \). The total running time is \( O(\log^2 \Delta \cdot \log n) \)-time bound for \((\text{degree } + 1)\)-list coloring in the LOCAL model.

The above algorithm requires the LOCAL model because for the rounding, nodes have to learn the complete fractional color assignment of their neighbors. For the CONGEST model, we build on ideas that were recently developed in [4], [26]. Assume that all nodes have a list consisting of colors from a range \( C \). Consider an arbitrary (fixed) partition of the color space \( C \) into a small number of parts \( C_1, \ldots, C_k \) of size approximately \( |C_i| \approx |C|/k \). Instead of directly coloring the nodes, the goal for each node \( v \) is to pick one of the parts \( i \in \{1, \ldots, k\} \) and update its list to \( L_u \cap C_i \). Node \( v \) then only remains in conflict with the neighbors that also pick the same color subspace \( C_i \) and the goal is to find an assignment of color subspaces such that on average over all nodes, the ratio between remaining degree and list size does not grow by more than a \( 1 + \varepsilon \) factor for an appropriate choice of \( \varepsilon \). In [4], it is shown that a simple randomized choice of color subspace \( C_i \) solves this problem in expectation. By using a generalization of the rounding process described above for coloring in the LOCAL model, we can efficiently compute a good assignment of color spaces. With the right choice of parameters, this leads to a \((\text{degree } + 1)\)-coloring algorithm with a time complexity of \( O(\log^2 |C| \cdot \log n) \), using messages of only \( O(\log |C|) \) bits.

D. Other Related Work

Graph coloring and variants of it have been at the center of studies in distributed algorithms for over three decades and there is a vast amount of related work on this topic. Here, we provide a brief (and certainly not exhaustive) review of the most relevant work. We refer to the 2013 Distributed Graph Coloring book of Barenboim and Elkin [7] for more information.

Deterministic algorithms, focusing on the \( n \)-dependency: We start with a review of the literature focusing on the \( n \)-dependency in the complexity. Linial’s \( O(\log^* n) \) round \( O(\Delta^2) \)-coloring algorithm directly translates to an \( O(\Delta^2 + \log^* n) \) round algorithm for \((\Delta + 1)\)-coloring. This complexity was improved to \( O(\Delta \log \Delta + \log^* n) \) by Kuhn and Wattenhofer [27], and subsequently to \( O(\Delta + \log^* n) \) by Kuhn [25] and independently by Barenboim and Elkin [6]. Recently, Barenboim, Elkin, and Goldenberg [8] presented an alternative (locally-iterative) algorithm with the same \( \Delta + \log^* n \) complexity. The complexity was improved to \( O(\Delta^{3/2}) + O(\log^* n) \) by Barenboim [5], and then to \( O(\Delta^{1/2}) + O(\log^* n) \) by Frieze, Heinrich, and Kosowski [16]. The bound has remained around this complexity, with the exception of the following improvements: the aforementioned

improved shortly after to \( 2^{O(\sqrt{\log n})} \) rounds by Panconesi and Srinivasan [33]. In terms of polylogarithmic-time algorithms, the first significant progress was made by Barenboim and Elkin who gave an \( O(\Delta^{1+o(1)}) \) coloring in \( \log n \) rounds.

Since then, progress on the \((\Delta + 1)\)-vertex coloring problem remained elusive, except for a recent \( 2^{O(\sqrt{\log \Delta}) \log n} \)-round algorithm of Kuhn [26]. But much more progress, and especially in the polylogarithmic complexity regime, was made on the simpler problem of \((2\Delta - 1)\) edge coloring (which is the special case of vertex coloring if we take the line graph) and its tighter variants: Ghaffari and Su gave a \( \text{poly}(log n) \) round algorithm for \((2\Delta - 1)(1 + o(1))\) edge coloring. Fischer, Ghaffari, and Kuhn [15] gave a \( \text{poly}(\log n) \) round algorithm for \( 2\Delta - 1 \) coloring. Ghaffari et al. [19] gave a \( \text{poly}(\log n) \) algorithm for \((1 + o(1))\Delta \) edge coloring assuming \( \Delta = \Omega(\log n) \) and a \( 3\Delta/2 \) coloring for general graphs. Improving on the polylogarithmic complexity, Harris [22] gave an algorithm with round complexity \( \text{poly}(\log^2 \Delta) \cdot \text{poly}(\log n) \) for \((2\Delta - 1)\)-edge coloring. All these developments remained confined to the edge-coloring problem and could not be extended to the harder vertex coloring problem.

The question of whether a \( \text{poly}(\log n) \) round deterministic algorithm for \((\Delta + 1)\)-coloring exists was finally resolved in 2020: Rozhoň and Ghaffari [35] gave an \( \text{poly}(\log^* n) \) round algorithm for network decomposition, which led to an algorithm with the same complexity for \((\Delta + 1)\)-coloring, in the LOCAL model. Bamberger et al. [4] showed how to solve coloring using small messages in low-diameter graphs and, combining this with the network decomposition of [35], obtained a \( \text{poly}(\log n) \) round algorithm in the CONGEST model for \((\Delta + 1)\)-coloring. Most recently, the network decomposition algorithm was improved in [17] to complexity \( \text{poly}(\log^5 n) \). For the \((\Delta + 1)\) coloring problem, this implied an \( \text{poly}(\log^5 n) \) round algorithm in the LOCAL model and a fairly involved \( \text{poly}(\log^6 n) \) algorithm in the CONGEST model.

Deterministic algorithms, focusing on the \( \Delta \)-dependency: We now review the literature with a focus on the \( \Delta \)-dependency. Linial’s \( O(\log^* n) \) round \( O(\Delta^2) \)-coloring algorithm directly translates to an \( O(\Delta^2 + \log^* n) \) round algorithm for \((\Delta + 1)\)-coloring. This complexity was improved to \( O(\Delta \log \Delta + \log^* n) \) by Kuhn and Wattenhofer [27], and subsequently to \( O(\Delta + \log^* n) \) by Kuhn [25] and independently by Barenboim and Elkin [6]. Recently, Barenboim, Elkin, and Goldenberg [8] presented an alternative (locally-iterative) algorithm with the same \( \Delta + \log^* n \) complexity. The complexity was improved to \( O(\Delta^{3/2}) + O(\log^* n) \) by Barenboim [5], and then to \( O(\Delta^{1/2}) + O(\log^* n) \) by Frieze, Heinrich, and Kosowski [16]. The bound has remained around this complexity, with the exception of the following improvements: the aforementioned
work of Barenboim et al. [8] improved the bound to \(O(\sqrt{\Delta \log \Delta} \log^* \Delta + \log^* n)\) and Maus and Tonoyan [32] sharpened this to \(O(\sqrt{\Delta \log \Delta} + \log^* n)\) rounds. These algorithms with \(O(\Delta^{1/2}) + O(\log^* n)\) complexity use large messages (roughly, \(O(\Delta)\) bits). For the \((\Delta - 1)\) edge coloring problem, which is a special case of \((\Delta + 1)\)-vertex coloring, we now know much better bounds: Kuhn [26] gave an \(2^{\sqrt{\log \Delta}} + O(\log^* n)\) round algorithm, and the bound was recently improved to \(\log^{O(\log \log \Delta)} \Delta + O(\log^* n)\) rounds by Balliu, Kuhn, and Olivetti [3].

Randomized algorithms, focusing on the \(n\)-dependency: As mentioned at the beginning, the celebrated algorithms of Luby [30] and Alon, Babai, and Itai [1] for maximal independent set gave an \(O(\log n)\) round algorithm for \((\Delta + 1)\)-coloring, and using \(O(\log n)\) bit messages. Barenboim, Elkin, Pettie, and Schneider [10] improved the complexity to \(O(\log \Delta) + 2^{O(\sqrt{\log \log \Delta})}\). The complexity was further improved by Harris, Su, and Schneider [23] to \(O(\sqrt{\log \Delta} + 2^{O(\sqrt{\log \log \log \Delta})})\), and then by Chang, Li, and Pettie [12] to \(O(\log^* \Delta) + 2^{O(\sqrt{\log \log \log \Delta})}\). In all these results, the latter term comes from the complexity of solving coloring (in fact, the harder variant of degree +1 list coloring) in graphs with \(O(\log n)\) nodes. Indeed, it is known that any faster randomized algorithm would imply a faster deterministic algorithm [13]. That term was improved with the network decomposition result of Rozhoň and Ghaffari [35] to \(\text{poly}(\log \log n)\). The best known upper bound on the complexity for randomized \((\Delta + 1)\)-coloring in the LOCAL model, prior to our work, is \(O(\log^3 \log n)\) rounds and follows from the improved network decomposition of [17] combined with the randomized coloring algorithm of Chang et al. [12]. The same round complexity has recently also been achieved in [20] for the CONGEST model. An even more recent paper by Hallårdörrsson, Nolin, and Tonoyan [21] gives an even faster randomized CONGEST algorithm for \((\Delta+1)\)-coloring. Their algorithm uses the deterministic CONGEST algorithm that we give in this paper to improve the round complexity of the problem to \(O(\log^3 \log n)\), matching the best in the LOCAL model that we give in Corollary I.2.

E. Mathematical Preliminaries and Outline

For a graph \(G = (V,E)\) and a node \(v \in V\), we use \(N(v)\) to denote the set of neighbors of \(v\) and we use \(d(v) = |N(v)|\) to denote the degree of \(v\). For an integer \(k \geq 1\), we define \([k] := \{1, \ldots, k\}\). Further, unless specified otherwise, when writing \(\log x\), we always mean \(\log_2 x\).

The remainder of the paper is organized as follows. In Section II, we give formal definitions of the type of weighted defective colorings we use in our algorithms and we state the formal defective coloring results we use. In Section III, we define and analyze a problem, where each node needs to choose a label from a given domain and there is a cost associated with each edge, depending on the edge and the labels assigned to both nodes of the edge. We in particular show that a fractional assignment of labels can be efficiently rounded to an integral labeling at only a small increase in cost. Finally in Section IV, we show how this rounding procedure can be used to obtain our coloring algorithms and prove Theorem I.1.

II. WEIGHTED DEFECTIVE COLORING

In this section, we define the notion of weighted defective coloring, review the known algorithms for it, and then state our faster algorithm for a relaxation of this problem, which we call weighted average defective coloring. This is a basic tool that we will use throughout the algorithm presented in the next section. We note that the main novelty of our approach lies in the next sections. The following weighted defective coloring variant has in particular been studied in [24].

Definition II.1 (Weighted Defective Coloring). Given a weighted graph \(G = (V,E,w)\) with non-negative edge weights \(w(e) \geq 0\) for all \(e \in E\), a parameter \(\epsilon > 0\), and an integer \(C \geq 1\), a weighted \(\epsilon\)-relative defective \(C\)-coloring of \(G\) is an assignment \(\varphi : V \rightarrow [C]\) of colors in \([C]\) to the vertices of \(G\) such that

\[
\forall u \in V : \sum_{v \in N(u)} I_{\{\varphi(u) = \varphi(v)\}} \cdot w(u,v) \leq \epsilon \cdot \sum_{v \in N(u)} w(u,v).
\]

Known Algorithms for Weighted Defective Coloring.: Kuhn [25] showed that in unweighted graphs, one can compute an \(\epsilon\)-relative coloring with \(O(1/\epsilon^2)\) colors in \(O(\log^* q)\) rounds of the CONGEST model, if one is given an initial proper \(q\)-coloring of the graph. Kawarabayashi and Schwartzman [24] observed that essentially the same algorithm and analysis can be also be generalized to the weighted case (for a formal statement, see Lemma V.1 in Section V).

Relaxing to Weighted Average Defective Coloring and Faster Algorithms.: We use weighted defective coloring as a basic subroutine in our algorithms. In fact, we can use even a weaker version of weighted defective coloring, where the defect is only bounded on average. For this weaker notion, we present a faster algorithm, i.e., one with a better trade-off between the weighted defect and the number of colors.

Definition II.2 (Weighted Average Defective Coloring). Given a weighted graph \(G = (V,E,w)\) with non-negative edge weights \(w(e) \geq 0\) for all \(e \in E\), a parameter \(\epsilon > 0\), and an integer \(C \geq 1\), a weighted average \(\epsilon\)-relative defective \(C\)-coloring of \(G\) is an assignment \(\varphi : V \rightarrow [C]\) of colors in \([C]\) to the nodes of \(G\) such that

\[
\sum_{u \in V} \sum_{v \in N(u)} I_{\{\varphi(u) = \varphi(v)\}} \cdot w(u,v) \leq \epsilon \cdot \sum_{u \in V} \sum_{v \in N(u)} w(u,v).
\]

The above definition is equivalent to requiring that at most an \(\epsilon\)-fraction of the total weight of all the edges of \(G\) is on monochromatic edges.
The following lemma shows that for every integer $C \geq 1$, there is an efficient distributed algorithm, with round complexity essentially linear in $C$, that computes a $C$-coloring such that the average weighted defect per node $v$ is almost equal to $1/C$ times the total weight of the edges of $v$. The proof of Lemma II.3 can be obtained by adapting relatively standard techniques and it therefore appears at the end of the paper in Section V.

**Lemma II.3** (Weighted Average Defective Coloring). Let $G = (V,E,w)$ be a weighted graph with non-negative edge weights and assume that we are given a proper vertex coloring of $G$ with colors in $[q]$. Then, for every integer $C \geq 1$ and every $\delta > 0$, there is a deterministic $O(C/\delta + \log^* q)$-round algorithm to compute a weighted average $(1 + \delta)/C$-relative defective $C$-coloring of $G$. The algorithm requires messages of size $O(\log(n))$ bits.

We note that the time bound in Lemma II.3 has an additive $\log^* q$ term. We could just use $\log^* q = \log^* n$ by using the unique IDs as colors. However, since we have to apply the weighted average defective coloring lemma several times, this would give a slightly worse overall time complexity for our coloring algorithm. In the application, we will therefore always first compute an $O(\Delta^2)$-coloring in time $O(\log n)$ at the beginning of the algorithm and we can afterwards have $\log^* q = \log^* \Delta$. In all our applications of the lemma, $C/\delta$ will be $\omega(\log^* \Delta)$, so that the term disappears asymptotically.

**Remark.** We note that for an unweighted graph $G = (V,E)$, the above lemma already gives a simple way to obtain an $O(\Delta \log n)$ coloring in $\text{poly}(\log n)$ time. In particular, for $C = 2$, the theorem implies that the nodes $V$ can be partitioned into two parts $V_a$ and $V_b$ such that the two induced subgraphs $G[V_a]$ and $G[V_b]$ together have at most $(1 + \delta) \cdot |E|/2$ edges. When iterating the same procedure on the individual parts $k = \log \Delta$ times and by setting $\delta = 1/\log \Delta$, we can divide the nodes $V$ into $2^k = \Delta$ parts $V_1, \ldots, V_{\Delta}$ such that all the induced subgraphs $G[V_1], \ldots, G[V_{\Delta}]$ together have at most

$$\frac{(1 + \delta)^k}{2k} \cdot |E| = \frac{(1 - 1/\log \Delta)^{\log \Delta}}{\Delta} \cdot |E| < \frac{e \cdot |E|}{\Delta} = O(n)$$

edges. Therefore a constant fraction of the nodes have at most a constant number of neighbors in their own part and by computing a maximal independent set of the bounded-degree parts of each graph $G[V_i]$, in $O(\log^* q)$ additional time, we can therefore color a constant fraction of all nodes of $G$ with $\Delta$ colors. By repeating $O(\log n)$ times, this gives an extremely simple $O(\log \Delta/\delta \cdot \log n) = O(\log^2 \Delta \cdot \log n)$-round algorithm to color $G$ with $O(\Delta \log n)$ colors. Even for such a coloring, the only previously known $\text{poly}(\log(n))$-time deterministic algorithm was based on first computing a network decomposition by using the recent algorithms of [35] or [17].

### III. Vertex Labelings with Edge Costs

As mentioned in the overview of our technique, a key part of our coloring algorithms is the rounding step. In this section, we present our rounding approach, formalized in the context of a vertex labeling problem that we define next.

#### A. Problem Statement

**Definition III.1** (Edge Cost Vertex Labeling). Given a graph $G = (V,E)$ and an edge cost function $c : (V \times \mathcal{L})^2 \rightarrow \mathbb{R}_{\geq 0}$. A solution is an assignment of a label $\ell_v \in \mathcal{L}$ to each node $v \in V$, where the cost of the labeling is given as

$$C := \sum_{\{u,v\} \in E} c((u, \ell_u), (v, \ell_v)).$$

As the main result of this section, we give an efficient deterministic algorithm that achieves the following. Given an arbitrary random assignment of labels, we show how to deterministically compute a labeling with a total cost that is almost as good as the expected cost of the random labeling. It is convenient to think of a random label assignment as a fractional labeling of the nodes, as defined below.

**Definition III.2** (Fractional Vertex Labeling). Given a graph $G = (V,E)$ and a label set $\mathcal{L}$, a fractional vertex labeling of $G$ is an assignment of fractional values $x_{v,\ell} \in [0,1]$ for each label $\ell$ to each node $v \in V$ such that for all $v \in V$, $\sum_{\ell \in \mathcal{L}} x_{v,\ell} = 1$. A fractional vertex labeling is called $(1/Q)$-integral for some positive integer $Q$ if all fractional values $x_{v,\ell}$ are integer multiples of $1/Q$.

The edge cost of a fractional vertex labeling is given by the expected cost if each node independently picks its label according to the probability distribution given by its fractional values.

**Definition III.3** (Edge Cost of Fractional Labeling). Given a graph $G = (V,E)$ and an instance of the vertex labeling with edge costs problem with label set $\mathcal{L}$ and edge cost function $c : (V \times \mathcal{L})^2 \rightarrow \mathbb{R}_{\geq 0}$, the cost $C$ of a fractional labeling $x_{v,\ell}$ for $v \in V$ and $\ell \in \mathcal{L}$ is defined as

$$C := \sum_{\{u,v\} \in E} \sum_{(\ell_u,\ell_v) \in \mathcal{L}^2} x_{u,\ell_u} \cdot x_{v,\ell_v} \cdot c((u, \ell_u), (v, \ell_v)).$$

#### B. The Algorithm for Rounding Vertex Labelings

We next show how a given fractional vertex labeling can be efficiently turned into an integer vertex labeling, while only losing a small factor in the overall edge cost. We first show how the method of conditional expectations can be used together with a given vertex coloring of a graph to round a given fractional solution without loss. We then see how to obtain a fast rounding method with only a small loss on the overall edge cost.
Lemma III.4 (Basic Rounding Lemma). Let $G = (V, E)$ be a graph and assume that for some integer $Q \geq 1$ and a label set $L$, we are given a $1/(2Q)$-integral fractional vertex labeling $x_{v,\ell}$ and an edge cost function $c : (V \times L)^2 \to \mathbb{R}_{\geq 0}$. If we are also given a proper $\gamma$-vertex coloring of $G$, there is an $\gamma$-round distributed algorithm to compute an $1/Q$-integral fractional vertex labeling $x'_{v,\ell}$ such that for all $v$ and $\ell$, $x'_{v,\ell} \leq 2 \cdot x_{v,\ell}$ and such that the total cost of the fractional vertex labeling $x'_{v,\ell}$ is not larger than the cost of the fractional vertex labeling $x_{v,\ell}$. The distributed algorithm requires messages of size $O\left(\min\{Q \cdot \log |L|, |L| \cdot \log Q\}\right)$ bits.

Proof: For a node $v \in V$, we define $L_v \subseteq L$ to be the set of labels $\ell$ for which $x_{v,\ell}$ is an integer multiple of $1/Q$ and we define $\overline{L}_v := L \setminus L_v$ to be the remaining set of labels. To get a $1/Q$-integral solution, for each node $v \in V$, we do the following. If $\ell \in L_v$, we can just define $x'_{v,\ell} := x_{v,\ell}$ because $x_{v,\ell}$ is already $1/Q$-integral. For the labels in $\overline{L}_v$, first observe that for each label $\ell \in \overline{L}_v$, we have $x_{v,\ell} = i/(2Q)$ for some odd integer $i \geq 1$ (otherwise, $x_{v,\ell}$ would be $1/Q$-integral). Hence, we know that the set $\overline{L}_v$ is of even size. If $\ell \in \overline{L}_v$, we set $x'_{v,\ell}$ to either $x_{v,\ell} - 1/(2Q)$ or to $x_{v,\ell} + 1/(2Q)$. In order to make sure that $\sum_{\ell \in L_v} x'_{v,\ell} = 1$ as required by Definition III.2, for exactly half the labels $\ell \in \overline{L}_v$, we need to set $x'_{v,\ell} = x_{v,\ell} - 1/(2Q)$ and for exactly half the labels $\ell \in \overline{L}_v$, we need to set $x'_{v,\ell} = x_{v,\ell} + 1/(2Q)$. In this way, we clearly guarantee that $x'_{v,\ell} \leq 2 \cdot x_{v,\ell}$. In the following, we describe how we choose which labels in $\overline{L}_v$ to round down and which labels in $\overline{L}_v$ to round up.

Recall that we are given a proper coloring of the nodes of $G$ with colors from $[\gamma]$. For every $i \in [\gamma]$, let $V_i$ be the set of nodes that are colored with color $i$. The algorithm consists of $\gamma$ phases $1, \ldots, \gamma$, where in phase $i$, all nodes in $V_i$ choose their new fractional label assignment. In each phase $i$, we show that we can round the fractional assignments of all $v \in V_i$ such that the total edge cost of the fractional labeling does not increase. For convenience, for all nodes $v \in V$, we define a variable $y_{v,\ell}$, which we initialize to $y_{v,\ell} = x_{v,\ell}$ at the beginning of the algorithm. When processing node $v$, we decide about the new fractional value $x'_{v,\ell}$ and we set $y_{v,\ell} = x'_{v,\ell}$. Like this, the values $y_{v,\ell}$ always define the current partially rounded fractional assignment.

Note that because we are given a proper $\gamma$-coloring, for every $i \in [\gamma]$, the nodes in $V_i$ form an independent set of $G$. Let us now focus on one phase $i$ and consider some node $v \in V_i$. Because $V_i$ is an independent set, for all the edges that are incident to node $v$, the change of the cost of those edges in phase $i$ only depends on how node $v$ chooses its new rounded fractional label assignment. We therefore have to show that node $v$ can pick the values $x'_{v,\ell}$ for all $\ell \in \overline{L}_v$ such that the total cost of its edges does not increase. For every node $v \in V_i$, we now define some weight $W_{v,\ell}$ for every $\ell \in \overline{L}_v$ as follows:

$$W_{v,\ell} := \sum_{u \in N(v)} \sum_{\ell' \in \overline{L}_v} y_{u,\ell'} \cdot c((v, \ell), (u, \ell')).$$

Note that $W_{v,\ell}$ is the cost of $v$‘s edges if $v$ would fix its label to $\ell$ (and the fractional assignment of $v$’s neighbors remains fixed). If the fractional value $x_{v,\ell}$ is increased or decreased by $1/(2Q)$, the total cost of $v$’s edges therefore increases or decreases by $W_{v,\ell}/(2Q)$. In order to not increase the total weight, $v$ therefore wants to decrease the fractional value for labels with small $W_{v,\ell}$ and it wants to increase the fractional value for labels with large $W_{v,\ell}$. Let $L := \overline{L}_v$ be the number of labels of node $v$ for which $x_{v,\ell}$ has to be rounded up or down. As mentioned, we know that $L$ is an even number. We define a set $\overline{L}_{v,-} \subseteq \overline{L}_v$ of size $\overline{L}_{v,-} = L/2$ by adding the $L/2$ labels $\ell \in \overline{L}_v$ with the largest weights $W_{v,\ell}$ to the set $\overline{L}_{v,-}$ (ties broken arbitrarily). This in particular implies that

$$\sum_{\ell \in \overline{L}_{v,-}} W_{v,\ell} \geq \sum_{\ell \in \overline{L}_{v,+}} W_{v,\ell}. $$

Node $v \in V_i$ computes its new fractional assignment $x'_{v,\ell}$ as follows. For all $\ell \in \overline{L}_{v,-}$, we set $x'_{v,\ell} := x_{v,\ell} - 1/(2Q)$ and for all $\ell \in \overline{L}_{v,+}$, we set $x'_{v,\ell} := x_{v,\ell} + 1/(2Q)$. Let $C_v$ be the total cost of all edges of $v$ at the beginning of phase $i$ and let $C'_v$ be the total cost of all edges of $v$ at the end of phase $i$. We have

$$C'_v - C_v = \frac{1}{2Q} \left[ \sum_{\ell \in \overline{L}_{v,+}} W_{v,\ell} - \sum_{\ell \in \overline{L}_{v,-}} W_{v,\ell} \right] \leq 0. $$

The rounding therefore does not increase the total edge cost. Clearly, every phase can be implemented in a single round, each node just needs to learn the current fractional assignment of all neighbors. A fractional assignment with $1/(2Q)$-integral fractional values can be encoded with $O\left(\min\{Q \cdot \log |L|, |L| \cdot \log Q\}\right)$ bits. This proves the bound on the message size, which concludes the proof.

The above lemma has a round complexity that is linear in the number of colors. Of course, if aiming for a proper coloring, this would require up to $\Delta + 1$ colors. In the next lemma, we see how to get a faster algorithm. Instead of using a proper coloring, we use a (weighted average) defective coloring. We will see that by just ignoring all monochromatic edges, we can relax the strict rounding objective to an approximate rounding (details appear in the full version [18]).

Lemma III.5 (Approximate Rounding Lemma). Let $\varepsilon > 0$ be a parameter and let $G = (V, E)$ be a graph and assume that for some integer $Q \geq 1$ and a label set $L$, we are given a $1/(2Q)$-integral fractional vertex labeling $x_{v,\ell}$ and an edge cost function $c : (V \times L)^2 \to \mathbb{R}_{\geq 0}$. If we are
also given a proper \( \gamma \)-vertex coloring of \( G \), there is an \( O(1/\varepsilon + \log^* \gamma) \)-round distributed algorithm to compute an 
1/Q-integral fractional vertex labeling \( x_{v,1} \) such that the total cost of the fractional vertex labeling \( x_{v,1} \) is by at most 
a \( 1 + \varepsilon \) factor larger than the cost of the fractional vertex labeling \( x_{v,\ell} \). The distributed algorithm requires messages of size 
\( O(\min \{ Q \cdot \log |\mathcal{L}|, |\mathcal{L}| \cdot \log Q \} + \log \gamma) \) bits.

**Corollary III.6.** Let \( \varepsilon > 0 \) be a parameter and let \( G = (V, E) \) be a graph and assume that for some integer \( k \geq 1 \), we are given a \( 1/2^k \)-integral fractional vertex labeling \( x_{v,\ell} \) and an edge cost function \( c : (V \times \mathcal{L})^2 \rightarrow \mathbb{R}_{\geq 0} \). If a proper \( \gamma \)-coloring of \( G \) is given, there is an \( O(k^2/\varepsilon + \log^* \gamma) \)-round algorithm to compute an integral vertex labeling with a total edge cost at most \( (1 + \varepsilon) \) times the total edge cost of the given fractional labeling. The algorithm uses messages of size 
\( O(\min \{ 2^k \cdot \log |\mathcal{L}|, |\mathcal{L}| \cdot k \} + \log \gamma) \) bits.

**IV. COLORING ALGORITHMS**

In this section, we explain how we use the vertex-labeling procedure explained in the previous section to obtain efficient vertex color algorithms.

**A. LOCAL Model Algorithm**

Consider a \((\text{degree} + 1)\)-list coloring problem: each node \( v \) is given a list of colors \( L_v \subseteq \{1, \ldots, \mathcal{C}\} \), with the guarantee that \( |L_v| \geq d(v) + 1 \) where \( d(v) \) denotes the degree of node \( v \). The objective is to assign each node \( v \) a color \( \ell \in L_v \) such that any two neighboring nodes have different colors.

**Theorem IV.1.** There is a deterministic distributed algorithm that solves any \((\text{degree} + 1)\)-list coloring problem in \( O(\log^2 \Delta \cdot \log n) \) rounds, using messages of size \( O(\Delta \cdot \log |\mathcal{C}|) \). Here, \( \mathcal{C} \) denotes the size of the entire space of colors, i.e., each node \( v \) has a color list \( L_v \subseteq \{1, \ldots, \mathcal{C}\} \).

**Proof:** We describe an algorithm that colors a constant fraction of nodes, in \( O(\log^2 \Delta) \) rounds. The theorem then follows by repeating this procedure \( O(\log n) \) times, each time removing the colors of the colored nodes from the lists of their neighbors.

As a helper tool, at the very beginning, we compute a \( O(\Delta^2) \) proper coloring \( \phi \) of all vertices. This can be done in \( O(\log n) \) time using Linial’s classic algorithm [28].

We now describe how we color a constant fraction of nodes. We start with a simple fractional color assignment and then use the rounding procedure described in the previous section to turn this into an integral color assignment with essentially the same cost. We then see how, thanks to the fact that the overall cost is small, we can properly color a constant fraction of the nodes.

**Fractional Color Assignment:** Consider a node \( v \) with color list \( L_v \) and degree \( d(v) \). Let \( \tilde{d}(v) \) be the largest power of 2 that is less than or equal to \( d(v) \) + 1, i.e., \( d(v) = 2^{\lceil \log_2 \tilde{d}(v) \rceil} + 1 \). Notice that \( \tilde{d}(v) \leq d(v) \) and \( d(v)/2 \leq \tilde{d}(v) \).

Let \( \tilde{L}_v \) be an arbitrary subset of \( L_v \) with size \( \tilde{d}(v) \). Let \( \mathcal{L} = \{1, \ldots, \mathcal{C}\} \). Consider the fractional assignment \( x_{v,\ell} \) where for each color \( \ell \in \tilde{L}_v \), we set \( x_{v,\ell} = \frac{1}{d(v)} \) and for every color \( \ell \notin \tilde{L}_v \), we set \( x_{v,\ell} = 0 \). Notice that this is a \( 1/Q \)-integral assignment for a value of \( Q = 2^{\lceil \log_2 (\Delta + 1) \rceil} \) and \( Q = O(\Delta) \).

Moreover, for each edge \( e = \{u, v\} \), define the cost function

\[
c((u, \ell_u), (v, \ell_v)) = \begin{cases} 1 & \text{if } \ell_u = \ell_v \\ 0 & \text{if } \ell_u \neq \ell_v \end{cases}
\]

Observe that for these fractional assignment and edge costs, we can upper bound the total cost as

\[
C = \sum_{\{u,v\} \in E} \sum_{(\ell_u, \ell_v) \in \mathcal{L}^2} x_{u,\ell_u} \cdot x_{v,\ell_v} \cdot c((u, \ell_u), (v, \ell_v))
\]

\[
= \sum_{\{u,v\} \in E} \sum_{\ell \in L_u \cap L_v} \frac{1}{\tilde{d}(u) \cdot \tilde{d}(v)}
\]

\[
= \frac{1}{2} \cdot \sum_{u \in V} \sum_{\ell \in N(u)} \tilde{L}_u \cdot \tilde{L}_v \cdot \frac{1}{\tilde{d}(u) \cdot \tilde{d}(v)}
\]

\[
\leq \frac{1}{2} \cdot \sum_{u \in V} \sum_{\ell \in N(u)} \tilde{d}(v) \cdot \frac{1}{\tilde{d}(u) \cdot \tilde{d}(v)}
\]

\[
= \frac{1}{2} \cdot \sum_{v \in V} \frac{d(v)}{d(u)} \leq \frac{1}{2} \cdot \sum_{v \in V} 2 = n.
\]

As a side comment, one can view the above fractional assignment as a randomized algorithm where each node \( v \) picks a uniformly random color in \( \tilde{L}_v \) and the cost function is simply the expected number of monochromatic edges — those edges that both endpoints get the same color — under this random color assignment.

**Integral Color Assignment:** By applying Corollary III.6, with \( \varepsilon = 1 \) and \( Q = O(\Delta) \), we get an integral assignment with cost at most \( 2n \), in \( O(\log^2 \Delta) \) rounds and using messages of size \( O(\Delta \log |\mathcal{C}|) \) bits. Once we have integral assignments, for each node \( v \), we have \( x_{v,\ell} = 1 \) for exactly one color \( \ell \in L_v \) and we have \( x_{v,\ell'} = 0 \) for any other color \( \ell' \). Hence, each node \( v \) is assigned exactly one color \( \ell \in \tilde{L}_v \subseteq L_v \). Moreover, in this case, the cost

\[
\sum_{\{u,v\} \in E} \sum_{(\ell_u, \ell_v) \in \mathcal{L}^2} x_{u,\ell_u} \cdot x_{v,\ell_v} \cdot c((u, \ell_u), (v, \ell_v))
\]

is simply the total number of monochromatic edges, i.e., the total number of edges \( e = \{u, v\} \) whose both endpoints \( u \) and \( v \) received the same color \( \ell \) as their integral color assignment. Hence, we have a color allocation to the nodes such that the number of monochromatic edges is at most \( 2n \). But this coloring might be improper and neighboring nodes might have the same color.
**Proper Coloring.** We now show that we can compute a proper coloring for at least a $1/10$ fraction of the nodes. Let $S$ be the set of nodes that have at most 4 monochromatic edges incident on them. We can conclude that $|S| \geq n/2$. Let $E'$ be the set of monochromatic edges with both points in $S$, and let $H' = (S, E')$. We compute a maximal independent set $S' \subseteq S$ in $H'$, in $O(\log^* \Delta)$ rounds. This can be done easily by using Linial’s algorithm [28] for this constant degree graph $H'$. The runtime is $O(\log^* \Delta)$ rounds, instead of $O(\log n)$ rounds, because we employ the $O(\Delta^2)$ coloring $\phi$ computed at the very beginning as the initial set of colors in Linial’s algorithm [28]. This is exactly the point that we discussed in footnote 3.

Notice that since $H'$ has maximum degree at most 4, we have $S' \geq |S|/5 \geq n/10$. We consider the integral assignments of nodes of $S'$ as their permanent color. Observe that two nodes $u, v \in S'$ might be neighboring in $G$ but then their integral color assignment must be on different colors, as otherwise the edge between them would be included in $E'$ and would be in contradiction with $S'$ being an independent set of $H' = (S, E')$. Hence, in $O(\log^2 \Delta + \log^* \Delta) = O(\log^* \Delta)$ rounds, we managed to color at least a $1/10$ fraction of the nodes with colors from their lists such that no two neighboring nodes received the same color.

**Wrap Up.** The above finishes the description of the procedure for coloring a constant fraction of the nodes. As mentioned at the beginning, the overall algorithm works by $O(\log n)$ repetitions of this procedure, each time on the nodes that remain uncolored. In each iteration, whenever we find permanent colors for some nodes, we remove the color permanently taken by each node from the list of the neighbors that remain uncolored. Since each permanently colored node takes away one color and one unit from the remaining degree, for each node $v$ that remains uncolored, the condition $|L_v| \geq d(v) + 1$ remains satisfied, where $L_v$ is now updated to be the set of remaining colors and $d(v)$ is the number of remaining neighbors.

**Theorem IV.1** directly implies a $(\Delta + 1)$-list coloring algorithm in $O(\log^2 \Delta \cdot \log n)$ rounds of the LOCAL model. Plugging this into the randomized algorithm of Chang et al. [12] improved the randomized complexity of computing $(\Delta + 1)$-coloring.

**Corollary I.2 (restated).** There is a randomized algorithm in the LOCAL-mode that computes a $(\Delta + 1)$-coloring in any graph with at most $n$ nodes and maximum degree at most $\Delta$ in $O(\log^3 \log n)$ rounds, with high probability.

**Proof Sketch:** The randomized part of the $(\Delta + 1)$-coloring algorithm of Chang et al. [12] runs in $O(\log^* n)$ rounds and colors most of the nodes, with the following guarantee: with high probability, each of the connected components in the subgraph induced by the nodes that remain uncolored has size $O(\log n)$. Then, we run the deterministic algorithm **Theorem IV.1** on each of these components separately. Notice that for each remaining node $v$, its list of remaining colors $L_v$ — i.e., those colors in $\{1, \ldots, \Delta + 1\}$ that are not already used by colored neighbors of $v$— has size $|L_v| \geq d(v) + 1$, where $d(v)$ denotes the number of neighbors of $v$ that remain uncolored. This is because, at the start $v$ had at most $\Delta$ neighbors and among the $\Delta + 1$ colors of $\{1, \ldots, \Delta + 1\}$, each already colored neighbor of $v$ used one color. Hence, the coloring problem in each connected component is an instance of $(\deg + 1)$-list coloring. Since the component size is $N = O(\log n)$, the algorithm of **Theorem IV.1** runs in $O(\log^3 N) = O(\log^3 \log n)$ rounds. Overall, this is a round complexity of $O(\log^* n + \log^2 \log n) = O(\log^3 \log n)$.

**B. CONGEST Model Algorithm**

**Theorem IV.2.** There is a deterministic distributed algorithm that solves any $(\deg + 1)$-list coloring problem in $O(\log^2 C \cdot \log n)$ rounds, using messages of size $O(\log C)$ bits. Here, $C$ denotes the size of the entire space of colors, i.e., each node $v$ has a color list $L_v \subseteq \{1, \ldots, C\}$.

The proof is done in a similar way as the proof of **Theorem IV.1**. At the core is a procedure that colors a constant fraction of nodes in $O(\log^2 C)$ rounds. The theorem then follows by repeating this procedure $O(\log n)$ times, each time removing the colors of the colored nodes from the lists of their neighbors. The details appear in the full version [18].

**Corollary IV.3.** There is a deterministic distributed algorithm that, on any $n$-node graph with maximum degree at most $\Delta$, computes a $(\Delta + 1)$-vertex coloring in $O(\log^2 \Delta \log n)$ rounds and using messages of size $O(\log n)$ bits.

**Proof:** The claim follows directly from **Theorem IV.2** by defining the list $L_v$ of each node $v$ as $L_v = \{1, \ldots, \Delta + 1\}$, which clearly satisfies $|L_v| \geq d(v) + 1$, and defining the space of colors $\{1, \ldots, C\} = \{1, \ldots, \Delta + 1\}$.

**V. THE ALGORITHM FOR WEIGHTED AVERAGE DEFECTIVE COLORING**

In this section, we give a distributed algorithm to solve the weighted average defective coloring problem and we prove **Lemma II.3**.

**A. Basic Weighted Average Defective Coloring Algorithms**

We start with the weighted version of a standard distributed defective coloring result. Recall that for a parameter $\varepsilon > 0$ and an integer $C \geq 1$, a weighted $\varepsilon$-relative defective $C$-coloring of a weighted graph $G = (V, E, w)$ is a $C$-coloring of the nodes $V$ such that for all nodes $v \in V$, the total weight of $v$’s monochromatic edges is at most an $\varepsilon$-fraction of the total weight of the total weight of $v$’s edges (cf. **Definition II.1**). By adapting an algorithm
and analysis of [25], Kawarabayashi and Schwartzman in [24] show how to efficiently compute a weighted \( \varepsilon \)-relative defective coloring with \( O(1/\varepsilon^2) \) colors.

**Lemma V.1 (Weighted Defective Coloring).** [24] Let \( G = (V, E, w) \) be a weighted graph with non-negative edge weights, let \( \varepsilon \in (0, 1) \) be a parameter, and assume that the nodes of \( G \) are properly colored with colors in \([q]\). Then, there is a distributed \( O(\log^* q)\)-round algorithm to compute a weighted \( \varepsilon \)-relative \( O(1/\varepsilon^2) \)-coloring of \( G \). If initially, every node knows the edge weights of all incident edges, the algorithm requires messages of \( O(\log q) \) bits.

Note that we could directly use Lemma V.1 instead of Lemma II.3 in all our distributed vertex coloring algorithms. This would however lead to coloring algorithms that are slower by two log-factors. In order to get a better trade-off between the defect and the number of colors of the defective coloring, we need to resort to the weaker weighted average defective coloring. Recall that for a parameter \( \varepsilon > 0 \) and an integer \( C \geq 1 \), a weighted \( \varepsilon \)-relative average defective \( C \)-coloring of a weighted graph \( G = (V, E, w) \) is a \( C \)-coloring of the nodes \( V \) such that the total weight of all monochromatic edges is at most an \( \varepsilon \)-factor of the total weight of all the edges of \( G \). We first give a simple iterative algorithm to compute a weighted average defective coloring.

**Lemma V.2.** Let \( G = (V, E, w) \) be a weighted graph with non-negative edge weights and assume that we are given a proper vertex coloring with colors in \([q]\) of \( G \). Then, for every integer \( C \geq 1 \), there is a deterministic \( O(q)\)-round algorithm to compute a weighted average \( 1/C \)-relative defective \( C \)-coloring of \( G \). The algorithm requires messages of size \( O(\log C) \).

**Proof:** The algorithm consists of \( q \) phases, where in phase \( i \in [q] \), the nodes with color \( i \) (in the initial proper \( q \)-coloring) choose their color of the defective coloring. For the sake of the analysis, assume that all edges in the graph are oriented from the node with the larger initial color to the node with the smaller initial color. Note that because the initial \( q \)-coloring is a proper coloring, the orientation of every edge is well-defined. Consider some node \( v \in V \) that chooses its color in \([C]\) in phase \( i \). When \( v \) chooses its color, all outgoing neighbors of \( v \) have already chosen their color in previous phases (because they have a smaller initial color). Node \( v \) chooses a color in \([C]\) such that the total weight of its monochromatic outgoing edges is minimized. Because \( v \) choose among \( C \) different colors, there exists a color such that the total weight of the monochromatic outgoing edges is at most a \( 1/C \)-fraction of the total weight of all outgoing edges. This directly implies that the computed coloring is weighted average \( 1/C \)-relative defective. Clearly, one phase of the algorithm can be implemented in a single communication round and in order to choose its color, a node only needs to know the colors (of the defective coloring) of the already processed neighbors. Thus, the algorithm only requires messages of \( O(\log C) \) bits.

**B. Time-Efficient Weighted Average Defective Coloring Algorithms**

As a next step, we give a recursive algorithm that achieves a similar trade-off between relative average defect and number of colors more efficiently.

**Lemma V.3.** Let \( G = (V, E, w) \) be a weighted graph with non-negative edge weights and assume that we are given a proper vertex coloring with colors in \([q]\) of \( G \). Then, for every \( \varepsilon \geq 1 \), there is a deterministic \( O(1/\varepsilon+\log(1/\varepsilon)\cdot\log^* q)\)-round algorithm to compute a weighted \( \varepsilon \)-relative average defective \( [2/\varepsilon] \)-coloring of \( G \). The algorithm requires messages of size \( O(\log q) \).

**Proof:** First note that we can assume that \( q > [2/\varepsilon] \) as otherwise, we can just output the initial proper \( q \)-coloring as the defective coloring. For \( \varepsilon > 0 \), let \( T(\varepsilon) \) denote the time that is required to compute a weighted \( \varepsilon \)-relative average defective \( [2/\varepsilon] \)-coloring of a given properly \( q \)-colored weighted graph. To prove the claimed time bound of the lemma, we thus need to show that \( T(\varepsilon) = O(1/\varepsilon+\log(1/\varepsilon)\cdot\log^* q) \). We argue that \( T(\varepsilon) \) can be phrased recursively as follows. There exists a constant \( c > 0 \) such that

\[
T(\varepsilon) = \begin{cases} 
T(2\varepsilon) + c/\varepsilon + O(\log^* q) & \text{if } \varepsilon < 1/2 \\
O(\log^* q) & \text{if } \varepsilon \geq 1/2
\end{cases}
\]

(2)

Let us first consider the case \( \varepsilon \geq 1/2 \). In this case, we can first compute a weighted \( (\varepsilon/2) \)-relative defective \( O(1) \)-coloring in time \( O(\log^* q) \) and with messages of size \( O(\log q) \) bits by using Lemma V.1. Let \( H \) be the subgraph of \( G \) that only consists of the bichromatic edges of \( G \) w.r.t. this \( O(1) \)-coloring. By using Lemma V.2 (for \( C = [2/\varepsilon] \)), we can then compute an \( (\varepsilon/2) \)-relative average defective \( [2/\varepsilon] \)-coloring of \( H \) in time \( O(1/\varepsilon) = O(1) \) with messages of size \( O(\log(1/\varepsilon)) = O(1) \) bits. This coloring has a weighted average relative defect of at most \( \varepsilon \) and we have thus shown that \( T(\varepsilon) = O(\log^* q) \) for \( \varepsilon \geq 1/2 \). As discussed, the algorithm only requires messages of \( O(\log q) \) bits.

For \( \varepsilon < 1/2 \), we first use Lemma V.1 to compute a weighted \( (1/4) \)-relative defective coloring of \( G \) with \( s = O(1) \) colors. This can be done in time \( O(\log^* q) \) with messages of size \( O(\log q) \) bits. Let \( G_1, \ldots, G_s \) be the subgraphs induced by \( s \) color classes of this coloring. Note that by definition, the total weight of all the edges in graph \( G_1, \ldots, G_s \) is at most \( W(G)/4 \), where \( W(G) \) is the total weight of all edges in \( G \). For each graph \( G_i \), we now recursively compute a weighted \( 2\varepsilon \)-relative average defective \( [1/\varepsilon] \)-coloring in time \( T(2\varepsilon) \). When combining the \( s \)-coloring of \( G \) with these \( [1/\varepsilon] \)-colorings of each graph \( G_i \), we obtain a \( s \cdot [1/\varepsilon] \)-coloring of \( G \). Let \( F \subseteq E \) be the set of monochromatic edges of this coloring. Note an edge \( e \in F \) if \( e \) is an edge of one of the graphs \( G_i \).
for $i \in [s]$ and if $e$ is monochromatic in the weighted $2\varepsilon$-relative average defective $\lfloor 1/\varepsilon \rfloor$-coloring of $G_i$. Because the total weight of the graphs $G_1, \ldots, G_s$ is at most $W(G)/4$, the total weight of all edges in $F$ is at most $2\varepsilon \cdot W(G)/4 = \varepsilon/2 \cdot W(G)$. Note also that by definition of the edge set $F$, the computed $s \cdot \lfloor 1/\varepsilon \rfloor$-coloring of $G$ is a proper coloring of the graph $H := (V, E \setminus F)$. We can therefore use Lemma V.2 to compute a weighted $(\varepsilon/2)$-relative average defective $\lceil 2/\varepsilon \rceil$-coloring of $H$ in time $O(s/\varepsilon)$ and with messages of size $O(\log(1/\varepsilon)) = O(\log q)$ bits. The total weight of all monochromatic edges of this coloring in $H$ is at most a $(\varepsilon/2)$-fraction of the total weight of all edges in $H$ and thus also at most $\varepsilon/2 \cdot W(G)$. Together with the edges in $F$, which also could be monochromatic, we therefore get an $\lceil 2/\varepsilon \rceil$-coloring, where the total weight of all monochromatic edges is at most $\varepsilon \cdot W(G)$ as required. This therefore proves the Equation (2).

The claim of the lemma now follows directly from Equation (2) and from the fact that, as discussed above, throughout the recursive algorithm, we only used messages of $O(\log q)$ bits.

Comment.: The argument in the above lemma can be seen as an adaptation of the recursive approach to solve $(\Delta + 1)$-coloring in $O(\Delta + \log^* n)$ time in [25]. We remark that alternatively, an essentially equivalent result to Lemma II.3 could also be proven by adapting the arbdecoloring algorithm of Barenboim, Elkin, and Goldenberg [8] to the weighted setting.

We now have the tools to prove Lemma II.3. For completeness, we first restate the lemma.

**Lemma II.3** (Weighted Average Defective Coloring). Let $G = (V, E, w)$ be a weighted graph with non-negative edge weights and assume that we are given a proper vertex coloring of $G$ with colors in $[q]$. Then, for every integer $C \geq 1$ and every $\delta > 0$, there is a deterministic $O(C/\delta + \log^* q)$-round algorithm to compute a weighted average $(1 + \delta)/C$-relative defective $C$-coloring of $G$. The algorithm requires messages of size $O(\log q)$ bits.

**Proof:** First note that w.l.o.g., we can assume that $q > C/\delta$, as otherwise, the claim of the lemma directly follows from Lemma V.2. In the following, we use $W(G) := \sum_{e \in E} w(e)$ to denote the total weight of all edges of a weighted graph $G$. Recall that to prove the lemma, we need to give a distributed algorithm to compute a $C$-coloring of $G$ such that the total weight of all the monochromatic edges is at most $\frac{1 + \delta}{C} \cdot W(G)$.

As a first step, we use Lemma V.1 to compute an $O((C/\delta)^2)$-coloring of the nodes of $G$ such that the total weight of all monochromatic edges is at most $\frac{\delta}{C} \cdot W(G)$. By Lemma V.1, such a coloring can be computed in $O(\log^* q)$ time and by using messages of $O(\log q)$ bits. Let $F \subseteq E$ be the set of monochromatic edges of this coloring and let $H := (V, E \setminus F)$ be the subgraph of $G$ consisting of the edges that are not monochromatic. Note that because $H$ only has bichromatic edges, the defective $O((C/\delta)^2)$-coloring that we have computed for $G$ is a proper coloring of $H$. For convenience, we define $\delta' := \delta/2$. To prove the lemma, we first apply Lemma V.3 to compute a weighted $(1 + \delta')/C$-relative average defective $C$-coloring of $H$. Even if all the edges in $F$ end up being monochromatic w.r.t. to this coloring, the total weight of all monochromatic edges is still at most $\frac{1 + \delta'}{C} \cdot W(H) + w(F)$. From $W(H) \leq W(G)$, $\delta' = \delta/2$ and $w(F) \leq \frac{\delta}{C} \cdot W(G)$, this immediately implies that the number of monochromatic edges is at most $\frac{1 + \delta}{C} \cdot W(G)$.

It therefore remains that we can compute a weighted $(1 + \delta')/C$-relative average defective coloring of a graph $H$ with an initial proper $O((C/\delta)^2)$-coloring in the required round complexity and with the required message size. The algorithm to achieve this consists of two steps. First, we apply Lemma V.4 to compute a weighted $(1 + \delta')/C$-relative average defective coloring of $H'$ in time $O(\log(C/\delta))$ and with messages of size $O(\log(C/\delta)) = O(\log q)$ bits. When using this coloring on $H$, the total weight of all monochromatic edges is at most $\frac{1 + \delta'}{C} \cdot W(H)$. Let $H''$ be the subgraph of $H$ that only consists of the bichromatic edges of this $\gamma$-coloring of $H$. We next apply Lemma V.2 to compute a weighted $1/C$-relative average defective $C$-coloring of $H''$ in time $O(\gamma) = O(C/\delta)$ and with messages of size $O(\log(C/\delta)) = O(\log q)$ bits. When using this coloring on $H$, the total weight of all monochromatic edges is at most $\frac{1 + \delta}{C} \cdot W(H)$ as required.

**References**

[1] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986.

[2] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network decomposition and locality in distributed computation. In Proc. 30th IEEE Symp. on Foundations of Computer Science (FOCS), pages 364–369, 1989.

[3] A. Balliu, F. Kuhn, and D. Olivetti. Distributed edge coloring in time quasi-polynomial in Delta. In Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC), pages 289–298, 2020.

[4] P. Bamberger, F. Kuhn, and Y. Maus. Efficient deterministic distributed coloring with small bandwidth. In Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC), pages 243–252, 2020.

[5] L. Barenboim. Deterministic $(\Delta + 1)$-coloring in sublinear (in $\Delta$) time in static, dynamic, and faulty networks. *Journal of the ACM*, 63(5):1–22, 2016.

[6] L. Barenboim and M. Elkin. Deterministic distributed vertex coloring in polylogarithmic time. In Proc. 29th Symp. on Principles of Distributed Computing (PODC), pages 410–419, 2010.
[7] L. Barenboim and M. Elkin. Distributed Graph Coloring: Fundamentals and Recent Developments. Morgan & Claypool Publishers, 2013.

[8] L. Barenboim, M. Elkin, and U. Goldenberg. Locally-iterative distributed \((\Delta + 1)\)-coloring below Szegedy-Vishwanathan barrier, and applications to self-stabilization and to restricted-bandwidth models. In Proc. 37th ACM Symp. on Principles of Distributed Computing (PODC), pages 437–446, 2018.

[9] L. Barenboim, M. Elkin, and F. Kuhn. Distributed \((\Delta + 1)\)-coloring in linear (in \(\Delta\)) time. SIAM Journal on Computing, 43(1):72–95, 2015.

[10] L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. The locality of distributed symmetry breaking. Journal of the ACM, 63(3):20, 2016.

[11] K. Censor-Hillel, M. Parter, and G. Schwartzman. De-randomizing local distributed algorithms under bandwidth restrictions. Distributed Comput., 33(3-4):349–366, 2020.

[12] Y.-J. Chang, W. Li, and S. Pettie. An optimal distributed \((\Delta + 1)\)-coloring algorithm? In Proc. 50th ACM Symp. on Theory of Computing (STOC), 2018.

[13] Y.-J. Chang and S. Pettie. A time hierarchy theorem for the LOCAL model. In Proc. 58th IEEE Symp. on Foundations of Computer Science (FOCS), pages 156–167, 2017.

[14] M. Fischer. Improved deterministic distributed matching via rounding. Distributed Computing, 33(3):279–291, 2020.

[15] M. Fischer, M. Ghaffari, and F. Kuhn. Deterministic distributed edge-coloring via hypergraph maximal matching. In Proc. 58th IEEE Symp. on Foundations of Computer Science (FOCS), 2017.

[16] P. Fraigniaud, M. Heinrich, and A. Kosowski. Local conflict coloring. In Proc. 57th IEEE Symp. on Foundations of Computer Science (FOCS), pages 625–634, 2016.

[17] M. Ghaffari, C. Grunau, and V. Rozhon. Improved deterministic network decomposition. In Proc. 33rd ACM-SIAM Symp. on Discrete Algorithms (SODA), pages 2984–2993, 2021.

[18] M. Ghaffari and F. Kuhn. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. CoRR, abs/2011.04511, 2020.

[19] M. Ghaffari, F. Kuhn, Y. Maus, and J. Uitto. Deterministic distributed edge-coloring with fewer colors. In Proc. 50th ACM Symp. on Theory of Computing (STOC), 2018.

[20] M. M. Halldörsson, F. Kuhn, Y. Maus, and T. Tonoyan. Efficient randomized distributed coloring in CONGEST. In Proc. 53rd ACM Symp. on Theory of Computing (STOC), 2021.

[21] M. M. Halldörsson, A. Nolin, and T. Tonoyan. Ultra-fast distributed coloring of high degree graphs. CoRR, abs/2105.04700, 2021.

[22] D. G. Harris. Distributed local approximation algorithms for maximum matching in graphs and hypergraphs. In Proc. 60th IEEE Symp. on Foundations of Computer Science (FOCS), pages 700–724, 2019.

[23] D. G. Harris, J. Schneider, and H.-H. Su. Distributed \((\Delta + 1)\)-coloring in sublogarithmic rounds. In Proc. 48th ACM Symp. on Theory of Computing (STOC), 2016.

[24] K. Kawarabayashi and G. Schwartzman. Adapting local sequential algorithms to the distributed setting. In Proc. 32nd Int. Symp. on Distributed Computing (DISC), pages 35:1–35:17, 2018.

[25] F. Kuhn. Local weak coloring algorithms and implications on deterministic symmetry breaking. In Proc. 21st ACM Symp. on Parallelism in Algorithms and Architectures (SPAA), 2009.

[26] F. Kuhn. Faster deterministic distributed coloring through recursive list coloring. In Proc. 32st ACM-SIAM Symp. on Discrete Algorithms (SODA), pages 1244–1259, 2020.

[27] F. Kuhn and R. Wattenhofer. On the complexity of distributed graph coloring. In Proc. 25th ACM Symp. on Principles of Distributed Computing (PODC), pages 7–15, 2006.

[28] N. Linial. Distributive graph algorithms – global solutions from local data. In Proc. 28th IEEE Symp. on Foundations of Computer Science (FOCS), pages 331–335, 1987.

[29] N. Linial. Locality in distributed graph algorithms. SIAM Journal on Computing, 21(1):193–201, 1992.

[30] M. Luby. A simple parallel algorithm for the maximal independent set problem. SIAM Journal on Computing, 15:1036–1053, 1986.

[31] M. Luby. Removing randomness in parallel computation without a processor penalty. J. of Computer and System Sciences, 47(2):250–286, 1993.

[32] Y. Maus and T. Tonoyan. Local conflict coloring revisited: Linial for lists. In Proc. 34th Int. Symp. on Distributed Computing (DISC), pages 16:1–16:18, 2020.

[33] A. Panconesi and A. Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In Proc. 24th ACM Symp. on Theory of Computing (STOC), pages 581–592, 1992.

[34] D. Peleg. Distributed Computing: A Locality-Sensitive Approach. SIAM, 2000.

[35] V. Rozhoň and M. Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In Proc. 52nd ACM Symp. on Theory of Computing (STOC), pages 350–363, 2020.