List Defective Colorings: Distributed Algorithms and Applications

Marc Fuchs
University of Freiburg, Germany

Fabian Kuhn
University of Freiburg, Germany

Abstract

The distributed coloring problem is at the core of the area of distributed graph algorithms and it is a problem that has seen tremendous progress over the last few years. Much of the remarkable recent progress on deterministic distributed coloring algorithms is based on two main tools: a) defective colorings in which every node of a given color can have a limited number of neighbors of the same color and b) list coloring, a natural generalization of the standard coloring problem that naturally appears when colorings are computed in different stages and one has to extend a previously computed partial coloring to a full coloring.

In this paper, we introduce *list defective colorings*, which can be seen as a generalization of these two coloring variants. Essentially, in a list defective coloring instance, each node v is given a list of colors $x_{v,1}, \dots, x_{v,p}$ together with a list of defects $d_{v,1}, \dots, d_{v,p}$ such that if v is colored with color $x_{v,i}$, it is allowed to have at most $d_{v,i}$ neighbors with color $x_{v,i}$.

We highlight the important role of list defective colorings by showing that faster list defective coloring algorithms would directly lead to faster deterministic $(\Delta+1)$-coloring algorithms in the LOCAL model. Further, we extend a recent distributed list coloring algorithm by Maus and Tonoyan [DISC '20]. Slightly simplified, we show that if for each node v it holds that $\sum_{i=1}^{p} \big(d_{v,i}+1\big)^2 > \deg_G^2(v) \cdot \operatorname{poly}\log\Delta$ then this list defective coloring instance can be solved in a communication-efficient way in only $O(\log\Delta)$ communication rounds. This leads to the first deterministic $(\Delta+1)$-coloring algorithm in the standard CONGEST model with a time complexity of $O(\sqrt{\Delta}\cdot\operatorname{poly}\log\Delta+\log^* n)$, matching the best time complexity in the LOCAL model up to a $\operatorname{poly}\log\Delta$ factor.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Mathematics of computing → Graph coloring

Keywords and phrases distributed coloring, list coloring, defective coloring

Digital Object Identifier 10.4230/LIPIcs.DISC.2023.22

Related Version *Full Version*: https://arxiv.org/abs/2304.09666 [15]

Funding This work was partially supported by the German Research Foundation (DFG) under the project number 491819048.

1 Introduction and Related Work

Distributed graph coloring is one of the core problems in the area of distributed graph algorithms. One typically assumes that the graph $G=(V,E)$ to be colored represents a communication network of n nodes with maximum degree Δ and that the nodes (or edges) of G must be colored in a distributed way by exchanging messages over the edges of G. The nodes typically interact with each other in synchronous rounds. If the size of messages is not restricted, this is known as the LOCAL model and if in every round, every node can send an $O(\log n)$-bit message to every neighbor, it is known as the CONGEST model [33]. The problem was first studied by Linial in a paper that pioneered the whole area of distributed graph algorithms [26]. Linial in particular showed that coloring a ring network with $O(1)$

colors (and thus coloring a graph with $f(\Delta)$ colors) requires $\Omega(\log^* n)$ rounds and that in $O(\log^* n)$ rounds, one can color any graph with $O(\Delta^2)$ colors. Subsequently, there has been a plethora of work on distributed coloring algorithms, e.g., [19, 1, 31, 36, 32, 25, 11, 7, 12, 5, 14, 22, 13, 24, 30, 29, 18, 20, 21]. The most standard variant of the distributed coloring problem asks for a proper coloring of the nodes V of G with $\Delta + 1$ colors. Note that this is what can be achieved by a simple sequential greedy algorithm.

Over the last approximately 15 years, we have seen remarkable progress on randomized and on deterministic distributed coloring algorithms. Much of the progress on deterministic algorithms (which are the focus of the present paper) has been achieved by studying and using two generalizations of the standard coloring problem, *defective colorings* and *list colorings*. In the following, we briefly discuss the history and significance of defective colorings and of list colorings in the context of deterministic distributed coloring algorithms. For lack of space, we do not discuss, the very early work on deterministic distributed coloring [26, 19, 36, 25], the deterministic algorithms that directly result from computing network decomposition [1, 32, 34, 17, 16], or the vast literature on randomized distributed coloring algorithms, e.g., [28, 35, 12, 22, 13, 20, 21].

Defective Coloring. Given integers $d \geq 0$ and $c > 0$, a d-defective c-coloring of a graph $G = (V, E)$ is an assignment of colors $\{1, \ldots, c\}$ to the nodes in V such that the subgraph induced by each color class has a maximum degree of at most d [27]. Defective colorings were introduced to distributed algorithms independently by Barenboim and Elkin [6] and by Kuhn [23] in 2009. Both papers give distributed algorithms to compute d-defective colorings with $O(\Delta^2/d^2)$ colors. The algorithm of [23] extends the classic $O(\Delta^2)$-coloring algorithm by Linial to achieving this in $O(\log^* n)$ time. Both papers use defective colorings to compute proper colorings in a divide-and-conquer fashion, leading to algorithms to compute a $(\Delta + 1)$-coloring in $O(\Delta + \log^* n)$ rounds.[1] This idea was pushed further by Barenboim and Elkin in [7] and [8]. In [7], they introduce the notion of *arbdefective colorings*: Instead of decomposing a graph into color classes of bounded degree, the aim is to decompose a graph into color classes of bounded arboricity. More specifically, the output of an arbdefective c-coloring algorithm with arbdefect d is a coloring of nodes with colors $\{1, \ldots, c\}$ together with an orientation of the edges such that every node has at most d outneighbors of the same color. For this more relaxed version of defective coloring, they show that for a given oriented graph with maximum outdegree β, one can efficiently compute a d-arbdefective coloring with $O(\beta/d)$-colors (in time $O(\beta^2/d^2 \cdot \log n)$). Applying this recursively, for example allows us to obtain a $\Delta^{1+o(1)}$-coloring in time $\operatorname{poly}\log\Delta \cdot \log n$. In [8], it is shown that for a family of graphs that includes line graphs, one can efficiently compute a standard d-defective coloring with only $O(\Delta/d)$ colors in time $O(\Delta^2/d^2 + \log^* n)$. This in particular implies that a $\Delta^{1+o(1)}$-edge coloring can be computed in time $\operatorname{poly}\log\Delta + O(\log^* n)$. All the algorithms of [6, 23, 7, 8] use defective coloring in the following basic way. If a computed defective coloring has p colors, the space of available colors is divided into p parts that can then be assigned to the p color classes and handled in parallel on the respective lower degree/arboricity graphs. When doing this, one inherently has to use more than $\Delta + 1$ colors because in each defective coloring step, the maximum degree (or outdegree) goes down at a factor that is somewhat smaller than the number of colors of the defective coloring. In [6, 23], this is compensated by reducing the number of colors at the end of each recursion level. However, this leads to algorithms with time complexity at least linear in Δ.

[1] Earlier algorithms were based on simple round-by-round color reduction schemes and required $O(\Delta^2 + \log^* n)$ [26, 19] and $O(\Delta \log \Delta + \log^* n)$ rounds [36, 25], respectively.

List Coloring. The key to obtaining $(\Delta+1)$-coloring algorithms with a better time complexity is to explicitly consider the more general $(\textit{degree}+1)$-list coloring problem. In this problem, every node v receives a list L_v of at least $\deg(v)+1$ colors as input and an algorithm has to properly color the graph in such a way that each node v is colored with a color from its list L_v. Note that this problem can still be solved by a simple sequential greedy algorithm. Note also that the problem appears naturally when solving the standard $(\Delta+1)$-coloring problem in different phases. If a subset $S \subseteq V$ is already colored, then each node $v \in V \setminus S$ needs to be colored with a color that is not already taken by some neighbor in S. If v has degree Δ and all already colored neighbors of v have chosen different colors, the list of the remaining available colors for v is exactly of length $\deg(v)+1$. The first paper that explicitly considered list coloring in the context of deterministic distributed coloring is by Barenboim [5]. In combination with the improved arbdefective coloring algorithm of [10], the algorithm of the paper obtains a $(1+\varepsilon)\Delta$-coloring in $O(\sqrt{\Delta}+\log^* n)$ rounds by first computing a $O(\sqrt{\Delta})$-arbdefective $O(\sqrt{\Delta})$-coloring and by afterwards iterating over the $O(\sqrt{\Delta})$ color classes of this arbdefective coloring and solving the corresponding list coloring problem in $O(1)$ time. With the same technique, the paper also gets an $O(\Delta^{3/4}+\log^* n)$-round algorithm for $(\Delta+1)$-coloring. This algorithm also works in the CONGEST model, i.e., by exchanging messages of at most $O(\log n)$ bits. For algorithms with a round complexity of the form $f(\Delta)+O(\log^* n)$, this still is the fastest known $(\Delta+1)$-coloring algorithm in the CONGEST model. The algorithm was improved by Fraigniaud, Kosowski, and Heinrich [14]. In combination with the subsequent results of [10, 30], the algorithm of [14] leads to an $O(\sqrt{\Delta \log \Delta}+\log^* n)$-round distributed algorithm for $(\textit{degree}+1)$-list coloring and thus also for $(\Delta+1)$-coloring. As one of the main results of this paper, we give a CONGEST algorithm that almost matches this and that has a time complexity of $O(\sqrt{\Delta}\operatorname{poly}\log\Delta+\log^* n)$. List colorings and defective colorings have also been explicitly used in all later deterministic distributed coloring algorithms [24, 4, 18, 3]. We next discuss an idea that was introduced in [24] and that is particularly important in the context of the present paper.

Distributed Color Space Reduction. The objective of [24] was to extend the coloring algorithms of [7, 8] to list colorings. The algorithms of [7, 8] are based on computing arbdefective or defective colorings to recursively divide the graph into low (out)degree parts that use disjoint sets of colors. This leads to fast coloring algorithms, however, the number of required colors grows exponentially with the number of recursion levels. While it is not clear how to efficiently turn a standard distributed coloring algorithm that uses significantly more than $\Delta+1$ colors into a $(\Delta+1)$-coloring algorithm, by using the techniques introduced in [5, 14], we can do this if we have a list coloring algorithm. Essentially, if we have a list coloring algorithm that uses lists of size $O(\alpha(\Delta+1))$, it can be turned into a $(\textit{degree}+1)$-list coloring algorithm in only $\tilde{O}(\alpha^2)$ rounds (and in some cases even in $O(\alpha)$ rounds). However, if the nodes have different lists, a defective coloring does not easily split the graph into independent coloring problems. As a generalization of defective colorings, [24] introduces a tool called *color space reduction*. Assuming that all lists consist of colors of some color space $\mathcal{C}$. For a given partition of $\mathcal{C}$ into disjoint parts $\mathcal{C}_1,\dots,\mathcal{C}_p$, a color space reduction algorithm partitions the nodes V into p parts $V_1,\dots,V_p$ such that for every node v, with $v \in V_i$ and v has $\deg_i(v)$ neighbors in V_i, the algorithm tries to keep the ratio $|L_v \cap \mathcal{C}_i|/\deg_i(v)$ as close as possible to the initial list-degree ratio $|L_v|/\deg(v)$. In [24], it is shown that the arbdefective and defective coloring algorithms of [7, 8] can be generalized to compute a color space reduction. If the size of the color space is polynomial in Δ, this lead to $(\textit{degree}+1)$-coloring algorithms with time complexities of $2^{O(\sqrt{\log\Delta})}\log n$ in general graphs and of $2^{O(\sqrt{\log n})}+O(\log^* n)$ in

graphs of bounded neighborhood independence, a family of graphs that includes line graphs of bounded rank hypergraphs. The complexity of the $(degree+1)$-edge coloring problem was later improved to $(\log\Delta)^{O(\log\log\Delta)}+O(\log^* n)$ in [4] and to $\operatorname{poly}\log\Delta+O(\log^* n)$ in [3]. In both cases, this was achieved by designing better distributed color space reduction algorithms for line graphs. The $(\Delta+1)$-coloring algorithm of [24] for general graphs was later subsumed by a deterministic $O(\log^2\Delta\cdot\log n)$-round algorithm for the $(\Delta+1)$-coloring problem in [18].

List Defective Colorings. Color space reductions can be seen as a special case of the following list variant of defective colorings. Each node v has a list of possible colors that it can choose (e.g., which color subspace $\mathcal{C}_i$ to use). Depending on what color v chooses, it can tolerate different defects (e.g., depending on the size $|L_{v,i}\cap\mathcal{C}_i|$ of the remaining color list when choosing color subspace $\mathcal{C}_i$). In the following, we formally define *list defective colorings*. One of the objectives of this paper is to understand the relation of list defective colorings to each other and to other coloring problems, and we will see in particular that better list defective coloring algorithms can directly lead to better algorithms for standard coloring problems.

In a list defective coloring problem, as input, each node v obtains a *color list* $L_v\subseteq\mathcal{C}$, where $\mathcal{C}$ is the space of possible colors. Each node v further has a defect function $d_v:L_v\to\mathbb{N}_0$ that assigns a non-negative integral defect value to each color in v's list L_v. Given vertex lists L_v, a list vertex coloring is an assignment $\varphi:V\to\mathcal{C}$ that assigns each node $v\in V$ a color $\varphi(v)\in L_v$. In the following, we formally define three variants of list defective coloring.

▶ **Definition 1** (List Defective Coloring). *Let $G=(V,E)$ be a graph, let $\mathcal{C}$ be a color space, and assume that each node $v\in V$ is given a color list $L_v\subseteq\mathcal{C}$ and a defect function $d_v:L_v\to\mathbb{N}_0$. Further, assume that we are given a list vertex coloring $\varphi:V\to\mathcal{C}$.*

- *The coloring φ is a* list defective coloring *iff every $v\in V$ has at most $d_v(\varphi(v))$ neighbors of color $\varphi(v)\in L_v$.*
- *If G is a directed graph, φ is called an* oriented list defective coloring *iff every $v\in V$ has at most $d_v(\varphi(v))$ out-neighbors of color $\varphi(v)\in L_v$.*
- *In combination with an edge orientation σ, φ is called a* list arbdefective coloring *iff it is an oriented list defective coloring w.r.t. the directed graph induced by the edge orientation σ.*

An (oriented) list (arb)defective coloring is called an (oriented) p-list (arb)defective coloring *for some integer $p>0$ if for all $v\in V$, $|L_v|\le p$.*

Note that the difference between an oriented list defective coloring and a list arbdefective coloring is that in an oriented list defective coloring, the edge orientation of G is given as part of the input and in a list arbdefective coloring, the edge orientation is a part of the output. By a result of Lovász [27], it is well-known that a d-defective c-coloring of a graph G with maximum degree Δ always exists if $c(d+1)>\Delta$. Note that this condition is also necessary if $G=K_{\Delta+1}$. By computing a balanced orientation of the edges of each color class of such a coloring, one can also deduce that a d-arbdefective c-coloring always exists if $c(2d+1)>\Delta$. Again, this condition is necessary if $G=K_{\Delta+1}$. By generalizing the potential function argument of [27], in the full version of this paper [15], we prove that the natural generalization of both existential statements also holds for the respective list defective coloring variants. Specifically, we show that for given color lists L_v and defect functions d_v, a list defective coloring always exists if

$$\forall v\in V:\sum_{x\in L_v}\big(d_v(x)+1\big)>\Delta \tag{1}$$

and a list arbdefective coloring always exists if

$$\forall v \in V : \sum_{x \in L_v} \big(2d_v(x) + 1\big) > \Delta. \tag{2}$$

Both conditions are necessary if the graph is a $(\Delta + 1)$-clique and if all nodes have the same color list and the same defect function. For arbdefective colorings, it has further been shown in [2] that Condition (1) is necessary and sufficient to compute such colorings in time $f(\Delta) + O(\log^* n)$. Whenever (1) does not hold, there is an $\Omega(\log_\Delta n)$-round lower bound for deterministic distributed list arbdefective coloring algorithms.

1.1 Our Contributions

In the following, whenever we consider a graph $G = (V, E)$, we assume that n denotes the number of nodes of G, Δ denotes the maximum degree of G, and $\deg(v)$ denotes the degree of a node v. Also, if G is a directed graph, β_v refers to the outdegree of node v and β denotes the maximum outdegree. Further, if we discuss any list defective coloring problem, unless stated otherwise, we assume that the colors come from space $\mathcal{C}$, $L_v \subseteq \mathcal{C}$ denotes the list of node v, and d_v denotes the defect function of nodes v. We further assume that $\Lambda := \max_{v \in V} |L_v|$ denotes the maximum list size. As it is common in the distributed setting, we do not analyze the complexity of internal computations at nodes. We briefly discuss the complexity of internal computations for our algorithms in the full version [15].

Oriented List Defective Coloring. As our main technical contribution, we give an efficient deterministic distributed algorithm for computing oriented list defective colorings. This algorithm is an adaptation of the techniques developed by Maus and Tonoyan [30] to obtain a 2-round algorithm for proper vertex colorings in directed graphs of small maximum outdegree.

▶ **Theorem 2.** *Let $G = (V, E)$ be a properly m-colored directed graph and assume that we are given an oriented list defective coloring instance on G. Assume that for every node $v \in V$, for a sufficiently large constant $\alpha > 0$, it holds that*

$$\sum_{x \in L_v} \big(d_v(x) + 1\big)^2 \geq \alpha \cdot \beta_v^2 \cdot \kappa(\beta, \mathcal{C}, m), \tag{3}$$

where $\kappa(\beta, \mathcal{C}, m) = (\log \beta + \log\log |\mathcal{C}| + \log\log m) \cdot (\log\log \beta + \log\log m) \cdot \log^2 \log \beta$.

Then, there is a deterministic distributed algorithm that solves this oriented list defective coloring instance in $O(\log \beta)$ rounds using $O\big(\min\{|\mathcal{C}|, \Lambda \cdot \log |\mathcal{C}|\} + \log \beta + \log m\big)$-bit messages.

Recursive Color Space Reduction. We have already discussed that we can use list defective colorings to recursively divide the color space. We next elaborate on power of doing recursive color space reduction directly for list defective coloring problems. The following theorem shows that in this way, at the cost of solving a somewhat weaker problem, we can sometimes significantly improve the time complexity or the required message size. In the following, we assume that we have an oriented list defective coloring algorithm $\mathcal{A}$, where the complexity is in particular a function of the maximum list size Λ. More specifically, the following theorem specifies the properties of $\mathcal{A}$ by an arbitrary parameter $\nu \geq 0$ and by arbitrary non-decreasing functions $\kappa(\Lambda)$, $T(\Lambda)$, and $M(\Lambda)$. Note that the functions $\kappa(\Lambda)$, $T(\Lambda)$, and $M(\Lambda)$ can in principle also depend on other global properties such as the maximum degree Δ, the maximum outdegree β, or the number of nodes n. When applying $\mathcal{A}$ to obtain algorithm $\mathcal{A}'$, we then however treat those other parameters as fixed quantities.

▶ **Theorem 3.** *Let $\nu \geq 0$ be a parameter and let $\kappa(\Lambda)$, $T(\Lambda)$, and $M(\Lambda)$ be non-decreasing functions of the maximum list size Λ. Assume that we are given a deterministic distributed algorithm $\mathcal{A}$ that solves oriented list defective coloring instances for which*

$$\forall v \in V \,:\, \sum_{x \in L_v} \big(d_v(x)+1\big)^{1+\nu} \geq \beta_v^{1+\nu} \cdot \kappa(\Lambda).$$

Assume further that the round complexity of $\mathcal{A}$ is $T(\Lambda)$ and that $\mathcal{A}$ requires messages of $M(\Lambda)$ bits.

Then, for any integer $p \in (1, |\mathcal{C}|]$, there exists a deterministic distributed algorithm $\mathcal{A}'$ that solves oriented list defective coloring instances for which

$$\forall v \in V \,:\, \sum_{x \in L_v} \big(d_v(x)+1\big)^{1+\nu} \geq \beta_v^{1+\nu} \cdot \kappa(p)^{\lceil \log_p |\mathcal{C}| \rceil}$$

in time $O(T(p) \cdot \log_p |\mathcal{C}|)$ and with $M(p)$-bit messages.

When replacing β_v by $\deg(v)$, the same theorem also holds for list defective colorings (in undirected graphs). One can easily see this as a list defective coloring can be turned into an equivalent oriented list defective problem, by replacing every edge $\{u, v\}$ of an undirected graph by the two directed edges (u, v) and (v, u).

Note that the number of colors of a standard defective coloring corresponds to the maximum list size Λ of a list defective coloring, i.e., a standard defective coloring with c colors is a special case of list defective coloring with lists of size c. Many of the existing defective and arbdefective coloring algorithms have a round complexity that is of the form $\text{poly}(c) + O(\log^* n)$ [8, 11, 10]. For a concrete application of Theorem 3, we therefore assume that the time complexity of algorithm $\mathcal{A}$ is of the form $T(\Lambda) = \text{poly}(\Lambda) + O(\log^* n)$. For simplicity, we further assume that the size of the color space $\mathcal{C}$ is at most polynomial in β. By setting $p = 2^{O(\sqrt{\log \beta \cdot \log \kappa(\Lambda)})}$, we then get an algorithm that solves oriented list defective coloring problems with $\forall v \in V \,:\, \sum_{x \in L_v} (d_v(x)+1)^{1+\nu} \geq \beta_v^{1+\nu} \cdot 2^{O(\sqrt{\log \beta \cdot \log \kappa(\Lambda)})}$ in time $2^{O(\sqrt{\log \beta \cdot \log \kappa(\Lambda)})} + O(\log^* n)$ rounds. For details, we refer to Corollary 13 in Section 4.

As a second application of Theorem 3, consider the oriented list defective coloring algorithm given by Theorem 2. The round complexity of this algorithm is $O(\log \beta)$ and we cannot hope to get a time improvement by recursively subdividing the color space. We can however improve the necessary message size. The message size of the algorithm is essentially linear in the maximum list size Λ. If we choose $p \ll \Lambda$, the message size becomes essentially linear in p. Assume for example that Λ and the color space are both polynomial in β. We then only need a constant number of recursion levels to reduce the message size to $O(\beta^\varepsilon + \log m)$ for any constant $\varepsilon > 0$ (see Corollary 14). We will apply this idea to obtain our new CONGEST algorithm for the $(\Delta+1)$-coloring problem.

Degree + 1 and List Arbdefective Colorings. The remaining contributions deal with applying (oriented) list defective coloring algorithms to solve the standard ($degree$+1)-coloring problem and more general other coloring problems. The following theorem shows that in combination with (oriented) list defective coloring algorithms, the general technique of [5, 14] cannot only be used to solve standard ($degree+1$)-coloring instances, but more generally also to solve list arbdefective coloring instances for which for all nodes v, $\sum_{x \in L_v} (d_v(x)+1) \geq \deg(v)+1$. Further, if we assume (oriented) list defective coloring algorithms of a certain quality (which is better than what we currently know), we directly obtain algorithms that potentially significantly improve the state of the art for the standard ($degree+1$)-coloring problem. For the following theorem, we assume that for two parameters $\nu > 0$ and

$\kappa > 0$, we have an oriented list defective coloring algorithm $\mathcal{A}^O_{\nu,\kappa}$ or a list defective coloring algorithm $\mathcal{A}^D_{\nu,\kappa}$ to solve instances for which for all v, $\sum_{x\in L_v}(d_v(x)+1)^{1+\nu} \geq \beta_v^{1+\nu}\cdot\kappa$ or $\sum_{x\in L_v}(d_v(x)+1)^{1+\nu} \geq \deg(v)^{1+\nu}\cdot\kappa$. We use $T^O_{\nu,\kappa}$ and $T^D_{\nu,\kappa}$ to denote the time complexities of the two algorithms. Note that the parameter κ can depend (monotonically) on global properties such as the maximum list size Λ, the maximum degree Δ, or the maximum outdegree β. We then however treat those global parameters as fixed quantities when applying the algorithms $\mathcal{A}^O_{\nu,\kappa}$ and $\mathcal{A}^D_{\nu,\kappa}$ recursively.

▶ **Theorem 4.** *Let $\nu \geq 0$ and $\kappa > 0$ be two parameters, let $G = (V, E)$ be an undirected graph with maximum degree Δ, and assume that we are given a list arbdefective coloring instance of G for which $\forall v \in V : \sum_{x\in L_v}(d_v(x)+1) > \deg(v)$. Using the oriented list defective coloring algorithm $\mathcal{A}^O_{\nu,\kappa}$, the given list arbdefective coloring problem can be solved in $O\big(\Lambda^{\frac{\nu}{1+\nu}}\cdot\kappa^{\frac{1}{1+\nu}}\cdot\log(\Delta)\cdot T^O_{\nu,\kappa}+\log^* n\big)$ rounds. Using the list defective coloring algorithm $\mathcal{A}^D_{\nu,\kappa}$, the given list arbdefective coloring problem can be solved in $O\big(\Lambda^\nu\cdot\kappa^2\cdot\log(\Delta)\cdot T^D_{\nu,\kappa}+\log^* n\big)$ rounds. If $\nu \geq \nu_0$ for some constant $\nu_0 > 0$, in both time bounds, the $\log(\Delta)$ term can be substituted by $\log(\Delta/\Lambda)$. If $\mathcal{A}^D_{\nu,\kappa}$ (or $\mathcal{A}^O_{\nu,\kappa}$) uses messages of at most B bits, then the resulting list arbdefective coloring algorithm uses messages of $O(B+\log n)$ bits.*

Note that the algorithm of Theorem 2 satisfies the requirements of algorithm $\mathcal{A}^O_{\nu,\kappa}$ for $\nu = 1$. If we assume that we first compute an $O(\Delta^2)$-coloring of G in time $O(\log^* n)$ by using a standard algorithm of [26] and if we assume the size of the color space is at most exponential in Δ, then $\kappa = O(\log\Delta\cdot\log^3\log\Delta)$ and $T^O_{\nu,\kappa} = O(\log\Delta)$. When using the algorithm of Theorem 2 as algorithm $\mathcal{A}^O_{\nu,\kappa}$ in Theorem 4, Theorem 4 therefore implies that the given arbdefective coloring instance can be solved in $O\big(\sqrt{\Lambda}\cdot\log^{3/2}\Delta\cdot\log^{3/2}\log\Delta+\log^* n\big)$ rounds. Hence, the theorem in particular entails that a d-arbdefective $\lfloor\frac{\Delta}{d+1}+1\rfloor$-coloring can be computed in $O(\sqrt{\Delta/(d+1)}\cdot\log^{3/2}\Delta\cdot\log^{3/2}\log\Delta+\log^* n)$ rounds. Even when using $O(\Delta/d)$ colors, the best previous algorithm for this problem required $O(\Delta/d+\log^* n)$ rounds [10]. Theorem 4 further shows that if we could get a fast oriented list defective coloring algorithm for a condition of the form $\sum_{x\in L_v}(d_v(x)+1)^{2-\varepsilon} \geq \beta_v^{2-\varepsilon}\operatorname{poly}\log\Delta$, we would already significantly improve the existing $O(\sqrt{\Delta\log\Delta}+\log^* n)$-round algorithm of [14, 10, 30] to compute a $(\Delta+1)$-coloring. The same would be true in case we could get a fast list defective coloring algorithm for a condition of the form $\sum_{x\in L_v}(d_v(x)+1)^{3/2-\varepsilon} \geq \deg(v)^{3/2-\varepsilon}\cdot\operatorname{poly}\log\Delta$. This indicates that the current obstacle for significantly improving the $O(\sqrt{\Delta\log\Delta}+\log^* n)$-round algorithm (in case this is possible) is to improve our understanding of the complexity of computing defective colorings and possible list defective colorings.

Faster Coloring in the CONGEST Model. Finally, we show that by combining Theorems 2–4, we obtain a faster (*degree* + 1)-list coloring for the CONGEST model.

▶ **Theorem 5.** *Let $G = (V, E)$ be an n-node graph and assume that we are given a (degree+1)-list coloring instance on G. If the color space $\mathcal{C}$ of the problem is of size $|\mathcal{C}| \leq \operatorname{poly}(\Delta)$, there exists a deterministic* CONGEST *algorithm for solving the (degree + 1)-list coloring instance in time $\sqrt{\Delta}\cdot\operatorname{poly}\log\Delta+O(\log^* n)$. If the color space is of size $|\mathcal{C}| = O(\Delta)$ (such as, e.g., for the standard $(\Delta+1)$-coloring problem), the time complexity of the algorithm is $O(\sqrt{\Delta}\cdot\log^2\Delta\cdot\log^6\log\Delta+\log^* n)$.*

Note that there is a $O(\log^2\Delta\cdot\log n)$-round CONGEST algorithm for solving (*degree* + 1)-list coloring instances [18]. Further, by [14, 10, 30] there is an $O(\sqrt{\Delta\log\Delta}+\log^* n)$ algorithm for the problem that uses messages of size $\tilde{O}(\Delta)$. Thus, $(\Delta+1)$-coloring algorithms running in $\sqrt{\Delta}\cdot\operatorname{poly}\log\Delta+O(\log^* n)$ rounds in the CONGEST model are already known as long

as $\Delta = O(\log n)$ or $\Delta = \Omega(\log^2 n)$. Our results fill this gap and give such an algorithm for $\Delta \in [\omega(\log n), o(\log^2 n)]$. A rough explanation of why the previous deterministic CONGEST algorithms fail to compute $(\Delta+1)$-colorings efficiently when $\Delta \in [\omega(\log n), o(\log^2 n)]$ is the following. In the algorithm of [14, 10, 30], every node has to *learn* the color lists of its neighbors, which requires that $\Omega(\Delta \cdot \log \Delta)$ bits have to be sent over every edge (which only works in CONGEST if $\Delta = O(\log n)$). For other algorithms (such as for the algorithm of [18]), the round complexity of the algorithms is at least $\Omega(\log n)$, which only leads to efficient time complexities in $\tilde{O}(\sqrt{\Delta})$ if $\Delta = \Omega(\log^2 n)$.

1.2 Organization of the paper

The remainder of the paper is organized as follows. In Section 2, we formally define the communication model and we introduce the necessary mathematical notations and definitions. Section 3 is the main technical section. It discusses our oriented list defective coloring algorithms, leading to the proof of Theorem 2. Subsequently, Section 4 discusses how to improve existing list (defective) coloring algorithms by recursively reducing the color space. Section 5 then shows how (oriented) list defective coloring algorithms can be applied to efficiently solve the standard $(\mathit{degree}+1)$-coloring problem and even list arbdefective coloring problems. The section also shows how this, in combination with the results in Section 3 and Section 4, leads to our new $(\mathit{degree}+1)$-coloring algorithm for the CONGEST model. Due to lack of space, all formal proofs are deferred to the full version of the paper [15].

2 Model and Preliminaries

Communication Model. In the LOCAL model and the CONGEST model [33], the network is abstracted as an n-node graph $G = (V, E)$ in which each node is equipped with a unique $O(\log n)$-bit identifier. Communication happens in synchronous rounds. In every round, every node of G can send a potentially different message to each of its neighbors, receive the messages from the neighbors and perform some arbitrary internal computation. Even if G is a directed graph, we assume that communication can happen in both directions. All nodes start an algorithm at time 0 and the time or round complexity of an algorithm is defined as the total number of rounds needed until all nodes terminate (i.e., output their color in a coloring problem). In the LOCAL model, nodes are allowed to exchange arbitrary messages, whereas in the CONGEST model, messages must consist of at most $O(\log n)$ bits.

Mathematical Notation. Let $G = (V, E)$ be a graph. Throughout the paper, we use $\deg_G(v)$ to denote the degree of a node $v \in V$ in G and $\Delta(G)$ to denote the maximum degree of G. If G is a directed graph, we further use $\beta_{v,G}$ to denote the outdegree of a node $v \in V$. More specifically, for convenience, we define $\beta_{v,G}$ as the maximum of 1 and the outdegree of v, i.e., we also set $\beta_{v,G} = 1$ if the outdegree of v is 0. The maximum outdegree β_G of G is defined as $\beta_G := \max_{v \in V} \beta_{v,G}$. We further use $N_G(v)$ to denote the set of neighbors of a node v and if G is a directed graph, we use $N_G^{out}(v)$ to denote the set of outneighbors of v. In all cases, if G is clear from the context, we omit the subscript G. When discussing one of the list defective coloring problems on a graph $G = (V, E)$, we will typically assume that $\mathcal{C}$ denotes the space of possible colors, and we use L_v and d_v for $v \in V$ to denote the color list and defect function of node v. Throughout the paper, we will w.l.o.g. assume that $\mathcal{C} \subseteq \mathbb{N}$ is a subset of the natural numbers. When clear from the context, we do not explicitly introduce this notation each time. Further, for convenience, for an integer $k \geq 1$, we use $[k] := \{1, \dots, k\}$ to denote the set of integers from 1 to k. Further, for a finite set A and an integer $k \geq 0$, we use 2^A to denote the power set of A and $\binom{A}{k}$ to denote the set of subsets of size k of A. Finally, we use $\log(x) := \log_2(x)$ and $\ln(x) := \log_e(x)$.

3 Distributed Oriented List Defective Coloring Algorithms

3.1 Fundamentals

Our algorithm in based on the list coloring approach of Maus and Tonoyan [30] that we sketch next. As input, each node of $G = (V, E)$ obtains a color list $L_v \subseteq \mathcal{C}$ of size $|L_v| \geq \alpha\beta^2\tau$ for a sufficiently large constant $\alpha > 0$ and some integer parameter $\tau > 0$. In addition, the nodes are equipped with an initial proper m-coloring of G. The "highlevel" idea is based on the classic one-round $O(\beta^2 \log m)$-coloring algorithm of Linial [26]. As an intermediate step of the algorithm of [30], every node v chooses a subset $C_v \subseteq L_v$ of size $|C_v| = \beta\tau$ of its list such that for every outneighbor u of v, it holds that $|C_u \cap C_v| < \tau$. To obtain a proper coloring of G, node v can then choose a color $x \in C_v$ that does not appear in any of the sets C_u of one of the $\leq \beta$ outneighbors u of v. If all nodes have to pick a color from $\{1, \dots, \alpha\beta^2\tau\}$ and if $\tau = O(\log\beta + \log m)$ is chosen sufficiently large, Linial shows that such sets C_v for all nodes v can be computed in 0 rounds without communication. However, this is not true for the list coloring variant of the problem considered in [30].

Before we show how the authors of [30] overcome the problems of list, we introduce some terminology. Let P_0 be the original list coloring problem that we need to solve and let P_1 be the intermediate problem of choosing a set C_v from $\binom{L_v}{\beta\cdot\tau}$ s.t. $|C_v \cap C_u| < \tau$ for all outneighbors u of v. As discussed, after solving P_1, P_0 can be solved in a single round, each node v just needs to learn the sets C_u of all its outneighbors u. To solve P_1, the authors of [30] introduce a new problem P_2, that can be seen as a "higher-dimensional" variant of Linial's algorithm. P_2 is defined in such a way that it can be solved without communication in 0 rounds and such that after solving P_2, P_1 can be solved in a single round. In problem P_2, every node v computes a set of possible candidates for the set C_v. For a more detailed description we need to define the following conflict relation.

▶ **Definition 6** (Conflict relation $\Psi(\tau', \tau)$)**.** *Let $\tau', \tau > 0$ be two parameters. The relation $\Psi(\tau', \tau) \subseteq 2^{2^{\mathcal{C}}} \times 2^{2^{\mathcal{C}}}$ is defined as follows. For any $K_1, K_2 \in 2^{2^{\mathcal{C}}}$, we have*

$$(K_1, K_2) \in \Psi(\tau', \tau) \Leftrightarrow \exists \text{ distinct } C_1, \dots, C_{\tau'} \in K_1 \text{ s.t.}$$
$$\forall i \in \{1, \dots, \tau'\}\ \exists C \in K_2 \text{ for which } |C_i \cap C| \geq \tau.$$

In a solution to problem P_2, every node v outputs a set $K_v \subseteq 2^{\binom{L_v}{\beta\tau}}$ such that $|K_v| = \beta\tau'$ (for some integer parameter $\tau' > 0$) and such that for every outneighbor u of v, $(K_v, K_u) \notin \Psi(\tau', \tau)$. Note that this implies that for every outneighbor u, K_v contains at most $\tau' - 1$ sets C for which there is a set $C' \in K_u$ for which $|C \cap C'| \geq \tau$. Because K_v has size $\beta\tau'$, there exists some $C_v \in K_v$ such that for every $C' \in K_u$ for every outneighbor u, we have $|C_v \cap C'| < \tau$. A solution of P_2 can be transformed into a solution of P_1 in a single round (each node v has to communicate it's set K_v to all its outneighbors u). Maus and Tonoyan [30] showed that for appropriate choices of the parameters τ and τ', P_2 can be solved in 0 rounds. To see this, consider the following technical lemma, which follows almost directly from Lemmas 3.3 and 3.4 in [30].

▶ **Lemma 7** (adapted from [30])**.** *Let $\gamma, \tau, \tau' \geq 1$ be three integer parameters such that $\tau \geq 8\log\gamma + 2\log\log|\mathcal{C}| + 2\log\log m + 16$ and $\tau' = 2^{\tau - \lceil\log(2e\gamma^2)\rceil}$. For every color list $L \in \binom{\mathcal{C}}{\ell}$ of size ℓ for some $\ell \geq 2e\gamma^2\tau$, we further define $S(L) := \binom{\binom{L}{\gamma\tau}}{\gamma\tau'}$. Then, there exists $\bar{S}(L) \subseteq S(L)$ such that $|\bar{S}(L)| \geq |S(L)|/2$ and such that for every $K \in \bar{S}(L)$ and every $L' \in \binom{\mathcal{C}}{\ell}$, there are at most $d_2 < \frac{1}{4m|\mathcal{C}|^\ell} \cdot |S(L)|$ different $K' \in S(L')$ such that $(K, K') \in \Psi(\tau', \tau)$ or $(K', K) \in \Psi(\tau', \tau)$. Further, for every $K \in S(L)$ and every $L' \in \binom{\mathcal{C}}{\ell}$, there are at most d_2 different $K' \in S(L')$ for which $(K, K') \in \Psi(\tau', \tau)$.*

Let us sketch how Lemma 7 implies that for appropriate choices of the parameters, P_2 can be solved without communication. In the following, we set the parameters of Lemma 7 as $\gamma := \beta$ and τ and τ' as given by the lemma statement. Assume that initially, every node v has a list L_v of size ℓ, where $\ell \geq 2e\beta^2\tau$. We define the type T_v of a node as the tuple (c, L_v), where c is the color of v in the initial proper m-coloring of G and $L_v \in \binom{\mathcal{C}}{\ell}$ is the color list of v. Let $T_1, \dots, T_t$ be a fixed ordering of the $t = m\binom{|\mathcal{C}|}{\ell} \leq m|\mathcal{C}|^\ell$ types and let L_i be the color list of nodes of type T_i. If we assign a set $K_i \in S(L)$ to each type T_i so that for any two sets K_i and K_j, $(K_i, K_j) \notin \Psi(\tau', \tau)$ and $(K_j, K_i) \notin \Psi(\tau', \tau)$, then if each node v of type T_i (for $i \in \{1, \dots, t\}$) outputs K_i, this assignment solves problem P_2. We assign the sets K_i greedily, where for every type T_i, we choose some $K_i \in \bar{S}(L_i)$ such that $\bar{S}(L_i)$ is the subset of $S(L_i)$ that is guaranteed to exist by Lemma 7. Assume for any $i \geq 1$ each type T_j for $j \in \{1, \dots, i-1\}$ already picked K_j. Then type T_i will pick some list $K_i \in \bar{S}(L)$ that does not conflict with choices of the $i-1$ previous types. By the lemma, for any type T_j, $j \neq i$, there are at most $d_2 < \frac{1}{4m|\mathcal{C}|^\ell} \cdot |S(L_i)| \leq \frac{1}{2t} \cdot |\bar{S}(L_i)|$ sets in $\bar{S}(L_i)$ that conflict. Because there are only t types, we can always choose an appropriate K_i that does not conflict with already assigned sets K_j for $j < i$. Consequently, P_2 can be solved in 0 rounds, and thus the original list coloring problem can be solved in 2 rounds.

3.2 Basic Oriented List Defective Coloring Algorithm

In the following, we first give a basic algorithm that solves a slightly generalized version of the OLDC problem. Concretely, the algorithm assigns a color $x_v \in L_v$ with defect $d_v(x_v)$ to each node v s.t. at most $d_v(x_v)$ outneighbors w of v choose a color x_w with $|x_v - x_w| \leq g$, where $g \geq 0$ is some given parameter. Recall that we assume that all colors are integers and therefore, the value $|x_v - x_w|$ is defined. Note that for $g = 0$, this is the OLDC problem as defined in Definition 1. We give a basic algorithm for this more general problem because we will need it as a subroutine in the algorithm for proving our main technical result, Theorem 2. The steps for solving the generalized OLDC problem are similar to the approach described in Section 3.1. We however in particular have to adapt the algorithm to handle the case where each node comes with an individual list size.

A single defect per node. At the core of our basic (generalized) OLDC algorithm is an algorithm that solves the following weaker variant of the problem. Instead of having color-specific defects, every node v has a fixed defect value $d_v \geq 0$, i.e., we have $d_v(x) = d_v$ for all $x \in L_v$. Based on an algorithm for this *single-defect* case, one can solve the general OLDC problem by using a reduction explained in the proof of Lemma 12. For that reason, assume during the following section that each node v has three given inputs, a color list L_v, a defect value $d_v \geq 0$ and the number of outneighbors β_v. For each node v, the algorithm then requires lists of size $|L_v| \geq \alpha(\beta_v/(d_v+1))^2 \cdot \tau$ for some constant $\alpha > 0$ and some parameter $\tau > 0$. Note that the required list size only depends on the ratio between β_v and $d_v + 1$ and not on their actual values. In the following section we therefore do not work with the defect value d_v or the outdegree β_v of some node, but with a value γ_v that is essentially equal to the ratio $\beta_v/(d_v+1)$ such that the list size of v is proportional to γ_v^2. More formally, we partition the nodes into so-called γ-classes such that nodes in the same class have the same value γ_v and hence a similar $\beta_v/(d_v+1)$ ratio. The details appear in the subsequent section.

3.2.1 γ-Classes and Parameters

Each node v comes with some parameter $\gamma_v = 2^i$ for some $i \in [h]$, where h is a parameter. We call i the γ-class of v. Since these γ-classes have a natural order, we call a node u to be in a *lower* (respectively *higher*) γ-class than v if $i_u < i_v$ (respectively $i_u > i_v$). We define the following two parameters, which both depend on the maximum γ-class index h, the color space $\mathcal{C}$, and the initial (proper) m-coloring of G.

$$\tau(h,\mathcal{C},m) := \lceil 8h + 2\log\log|\mathcal{C}| + 2\log\log m + 16\rceil, \tag{4}$$

$$\tau'(h,\mathcal{C},m) := 2^{\tau(h,\mathcal{C},m)-\lceil 2h+\log(2e)\rceil}. \tag{5}$$

Note that these choices are consistent with the parameter setting in Lemma 7. If clear from the context, we omit the parameters $h,\mathcal{C}$ and m for simplicity and denote $\tau(h,\mathcal{C},m)$ by τ and $\tau'(h,\mathcal{C},m)$ by τ'. A node v of γ-class i_v is equipped with a color list L_v of size $|L_v| = \ell_{i_v} := \alpha \cdot 4^{i_v}\tau(2g+1) = \alpha\gamma_v^2\tau(2g+1)$ for some sufficiently large $\alpha > 0$ and $g > 0$. Because we solve a generalized version of the OLDC problem, we have to use a more general form for our conflict relation Ψ. Let $x \in \mathcal{C}$ be a color and $C \subseteq \mathcal{C}$ a set of colors, we denote the number of conflicts of x with colors in C regarding some given $g \geq 0$ by $\mu_g(x,C) := |\{c \in C \mid |x-c| \leq g\}|$.

▶ **Definition 8** (τ&g-conflict). *Two lists $C, C' \subseteq \mathcal{C}$ do τ&g-conflict if $\sum_{x\in C}\mu_g(x,C') \geq \tau$.*

Note that $\sum_{x\in C}\mu_g(x,C') = \sum_{x\in C'}\mu_g(x,C)$ is always true. The Ψ conflict relation from Definition 6 is adapted accordingly.

▶ **Definition 9** (Conflict relation $\Psi_g(\tau',\tau)$). *Let $\tau', \tau > 0$ be two parameters. The relation $\Psi_g(\tau',\tau) \subseteq 2^{2^{\mathcal{C}}} \times 2^{2^{\mathcal{C}}}$ is defined as follows. For any $K_1, K_2 \in 2^{2^{\mathcal{C}}}$, we have*

$$(K_1,K_2) \in \Psi_g(\tau',\tau) \Leftrightarrow \exists \text{ distinct } C_1,\dots,C_{\tau'} \in K_1 \text{ s.t.}$$
$$\forall i \in \{1,\dots,\tau'\}\ \exists C \in K_2 \text{ for which } C_i \text{ and } C \text{ do } \tau\&g\text{-conflict.}$$

We adapt the definitions of problem P_1 and P_2 to what we need for the generalized OLDC problem. We define $k_i := 2^i \cdot \tau$ and $k' := 2^h \cdot \tau'$. Subsequently, we denote the γ-class of a node v by i_v.

▶ **Definition 10** (Problems P_1 and P_2).

P_1: *Every node v has to output $C_v \subseteq L_v$ of size $|C_v| = k_{i_v}$ s.t. there are at most $d_v/2$ outneighbors u of v s.t. u is in γ-class $i_u \leq i_v$, C_u and C_v do τ&g-conflict.*

P_2: *Every node v has to output a list $K_v \in 2^{\binom{L_v}{k_{i_v}}}$ of size $|K_v| = k'$ s.t. for each outneighbor u in γ-class $i_u \leq i_v$, $(K_v,K_u) \notin \Psi_g(\tau',\tau)$.*

The next lemma states that P_2 can be solved in zero rounds. The high-level idea is to first adapt Lemma 7 such that we can apply it even if the size of the initial color lists differs. We start by using a simple trick to make sure that the color list L_v of each node v does not contain colors that are *close* to each other i.e., there are no distinct colors $x_1, x_2 \in L_v$ s.t. $|x_1 - x_2| \leq g$. After doing this, the conflict relation Ψ_g behaves almost the same as Ψ behaves in the fundamental problem. For details, we refer to the full version of the paper [15].

▶ **Lemma 11.** *P_2 can be solved without communication given an initial m-coloring.*

3.2.2 Algorithm

Assume each node is equipped with a color list of size $|L_v| \geq \alpha \left(\beta_v/(d_v+1)\right)^2 \tau(2g+1)$ and some defect value $d_v > 0$. The γ-class of a node v is defined as the smallest i_v s.t. $2^{i_v} \geq \frac{2\beta_v}{d_v+1}$. Based on the individual γ-class, each node solves P_2 (Lemma 11) and forwards the solution to the neighbors. The knowledge gained by that is used to solve P_1 without additional communication. In more detail, node v comes with list K_v s.t. $(K_v, K_u) \notin \Psi_g(\tau', \tau)$ for any outneighbor u with $i_u \leq i_v$. This implies that at most $\tau' - 1$ lists $C \in K_v$ do $\tau\&g$-conflict with some list in K_u. Hence, there are at most $\beta_v(\tau'-1)$ many $C \in K_v$ that $\tau\&g$-conflict. By the pigeonhole principle there is some $C_v \in K_v$ with at most[2] $\beta_v(\tau'-1)/k' < (d_v+1)/2$ many such conflicts within all the C_u of outneighbors u of smaller γ-classes. Hence, C_v is a valid solution for P_1 that is then forwarded to the neighbors.

To solve the list coloring problem itself we iterate through the γ-classes in descending order. Each node v has to decide on a color $x \in C_v$ s.t. in the end at most d_v outneighbors are colored with the same color x. Let us fix a node v. By design, in the iteration v decides on a color, all outneighbors of higher γ-classes already decided on a color and v knows the P_1 solution lists C_u of the outneighbors u with $i_u \leq i_v$. Let $f_v(x)$ be the frequency of color x within the outneighbors u of v i.e., the sum over all occurrences of x in C_u's of neighbors with the same or smaller γ-class plus the number of outneighbors of higher γ-classes that are already colored with x. The color x with the lowest frequency in C_v will be the final color of v for the following reason: There are at most $d_v/2$ outneighbors that have an unbounded number of $\tau\&g$-conflicts, while C_v shares at most $\tau - 1$ colors with the remaining C_u's (note that with outneighbors of higher γ-class at most one color in C_v is in conflict, hence, for worst-case observation we can ignore that case). By the pigeonhole principle there exist a color $x \in C_v$ with

$$f_v(x) \leq \frac{\sum_{c \in C_v} f_v(c)}{|C_v|} \leq \frac{d_v/2 \cdot |C_v| + \beta_v(\tau-1)}{|C_v|} < d_v + 1.$$

The round complexity of the whole algorithm is $O(h)$, since we iterate through all the γ-classes to assign colors. This completes the algorithm to handle *single* defects. We will now extend this result to solve the OLDC problem.

3.2.3 Multiple Defects

In the general case, each node can have lists that are composed of colors of different defects. In order to reduce the general case to the case, where each node only has a single defect, every node v first rounds all its defects to the next smaller power of 2. Every node v then can have $h = O(\log \beta)$ different defect values. The node v only keeps the colors in its list for one of those defect values. This value is chosen such that the sum $\sum_{x \in L'_v} (d_v(x)+1)^2$ is maximized, where L'_v is the reduced list of v. Note that when doing this, the sum $\sum_{x \in L_v} (d_v(x)+1)^2$ for the original list L_v is at most by an $O(h) = O(\log \beta)$ factor larger.

▶ **Lemma 12.** *Given a graph with an initial m-coloring. There is an $O(h)$-round* LOCAL *algorithm that assigns each node v a color $x_v \in L_v$ such that every node $v \in V$ has at most $d_v(x_v)$ outneighbors w with a color x_w for which $|x_w - x_v| \leq g$ if for each node $v \in V$*

$$\sum_{x \in L_v} (d_v(x)+1)^2 \geq \alpha \beta_v^2 \cdot \tau(h, \mathcal{C}, m) \cdot h \cdot (2g+1)$$

for some sufficiently large constant α, some integer $h \geq \max_v \lceil \log(\frac{\beta_v}{\min_{x \in L_v} d_v(x)+1}) \rceil$ and color space $\mathcal{C}$. Messages are of size at most $O(\min\{\Lambda \cdot \log|\mathcal{C}|, |\mathcal{C}|\} + \log\log\beta + \log m)$-bits.

[2] By definition of the γ-class, we have $(d_v+1)/2 \geq \beta_v/2^h$

To apply this lemma one can use that $h = O(\log \beta)$ and for some color space of size poly Δ and initial $O(\Delta^2)$-coloring (e.g., by [26]) we have $\tau(h, \mathcal{C}, m) = O(\log \Delta)$. Hence, Lemma 12 solves the OLDC problem (where $g = 0$) in $O(\log \Delta)$ communication rounds if the initial color list L_v for each node v fulfills the condition $\sum_{x \in L_v} (d_v(x) + 1)^2 \geq \alpha \beta_v^2 \cdot \log^2 \Delta$ for some sufficient large constant α. Note that the OLDC algorithm mentioned in Theorem 2 requires a stronger condition on the color lists L_v than Lemma 12. However, to reach this better result, we apply Lemma 12 as subroutine. The details of this more involved analysis are stated in Appendix A.

4 Recursive Color Space Reduction

Distributed list defective coloring algorithms first implicitly appeared in [24] as a tool to recursively reduce the color space of a distributed coloring problem. In this section, we show that the idea of using list defective colorings to recursively reduce the color space can also directly be applied to the (oriented) list defective coloring problem. In this way, at the cost of requiring slightly larger lists, we can turn a given distributed (oriented) list defective coloring algorithm into another distributed (oriented) list defective coloring algorithm that is faster and/or needs smaller messages. The high-level idea is the following. Assume that we are given an (oriented) list defective coloring problem with colors from a color space $\mathcal{C}$. We can arbitrarily partition $\mathcal{C} = \mathcal{C}_1 \cup \dots \cup \mathcal{C}_p$ into p approximately equal parts. Instead of directly choosing a color, each node v now first just selects the color subspace $\mathcal{C}_i$ from which v chooses its color. If v starts with color list L_v, then after choosing the color subspace $\mathcal{C}_i$, v's color list reduces to $L_{v,i} = L_v \cap \mathcal{C}_i$ (with the original defects on those colors). However, v now only has to compete with neighbors that also pick the same color subspace $\mathcal{C}_i$. The choices of color subspaces by the nodes can itself be phrased as an (oriented) list defective coloring instance for a color space of size p and thus also with lists of size at most p. Theorem 3 in Section 1.1 formalizes this idea.

It has been well-known since Linial's seminal work in [26] that in directed graphs of outdegree at most β, one can compute a proper $O(\beta^2)$-coloring in $O(\log^* n)$ rounds (or in $O(\log^* m)$ rounds if an initial proper m-coloring is provided). In [23], it was shown that in the same way, one can also compute an oriented d-defective coloring with $O((\beta/d)^2)$ colors. In [30], the coloring result of [26] was extended to the list coloring problem and in Section 3 of this paper (and to a limited extent also in [30]), the defective coloring result of [23] is extended to the oriented list defective coloring problem. While there has been progress on solving the natural list and defective coloring variants of $O(\beta^2)$ coloring, it is still unknown if a coloring with $O(\beta^{2-\varepsilon})$ colors (for some constant $\varepsilon > 0$) can be computed in time $f(\beta) + O(\log^* n)$.[3] Even if a moderately fast distributed algorithm for better oriented list defective colorings exists, we directly also get much faster algorithms for computing proper colorings with $o(\beta^2)$ colors. In the following, we assume that there exists an oriented defective coloring algorithm with a round complexity that is polynomial in the number of colors per node plus $O(\log^* n)$. Such algorithm for example exist for (list) defective colorings in graphs of neighborhood independence at most Δ^ε [9, 24].

[3] Note that oriented graphs with maximum outdegree β have colorings with $O(\beta)$ colors. However, the best distributed algorithm to compute an $O(\beta)$-coloring requires time $O(\log^3 \beta \cdot \log n)$ [18]. It is not known if colorings with $O(\beta^{2-\varepsilon})$ colors can be computed with an n-dependency $o(\log n)$.

▶ **Corollary 13.** *Let $\nu \geq 0$ be a parameter and let $\kappa(\Lambda)$ be a non-decreasing functions of the maximum list size Λ. Assume that we are given a deterministic distributed algorithm $\mathcal{A}$ that solves oriented list defective coloring instances for which*

$$\forall v \in V \,:\, \sum_{x \in L_v} \big(d_v(x)+1\big)^{1+\nu} \geq \beta_v^{1+\nu} \cdot \kappa(\Lambda).$$

Assume further that if an initial proper m-coloring is given, $\mathcal{A}$ has a round complexity of $\mathrm{poly}(\Lambda)+O(\log^* m)$. Then, there exists a $\big(2^{O(\sqrt{\log\beta \log\kappa(\Lambda)})}+O(\log^* m)\big)$-round deterministic distributed list coloring algorithm $\mathcal{A}'$ to solve list coloring instances with colors from a color space of size $\mathrm{poly}(\beta)$ for which

$$\forall v \in V \,:\, \sum_{x \in L_v} (d_v(x)+1)^{1+\nu} \geq \beta_v^{1+\nu} \cdot 2^{O(\sqrt{\log\beta \cdot \log\kappa(\Lambda)})}.$$

Note that the $2^{O(\sqrt{\log\Delta})}+O(\log^* n)$-round algorithm for computing a $(\Delta+1)$-coloring in graphs of bounded neighborhood independence and thus in particular in line graphs of bounded rank hypergraphs is based on the same idea as Corollary 13. The corollary shows how in some cases, recursive color space reduction can be used to significantly speed up a given (oriented) list defective coloring problem. The following corollary shows that recursive color space reduction can sometimes also be used to significantly reduce the required message size of an (oriented) list defective coloring algorithm. In the oriented list defective coloring algorithm of Section 3, all nodes need to learn the lists and defect vectors of their neighbors and this dominates the required communication. A list L_v of length $|L_v| \leq \Lambda$ consisting of colors from a color space of size $|\mathcal{C}|$ can be represented by $\min\{|\mathcal{C}|, \Lambda\log|\mathcal{C}|\}$ bits and a corresponding defect vector can be represented by $\Lambda\log\beta$ bits, or even by $\Lambda\log\log\beta$ bits if we assume that all defects are integer powers of 2 (which can usually be assumed at the cost of a factor 2 in the required list size). In the following, we assume that we have an algorithm that requires $O(|\mathcal{C}|\cdot B+\log n)$ bits for some parameter $B \geq 1$ (the $\log n$ is included to cover things like exchanging initial colors, unique IDs, etc.).

▶ **Corollary 14.** *Let $\nu \geq 0$ be a parameter and let $\kappa(\Lambda)$ be a non-decreasing functions of the maximum list size Λ. Assume that we are given a deterministic distributed algorithm $\mathcal{A}$ that solves oriented list defective coloring instances for which*

$$\forall v \in V \,:\, \sum_{x \in L_v} \big(d_v(x)+1\big)^{1+\nu} \geq \beta_v^{1+\nu} \cdot \kappa(\Lambda).$$

Assume further that $\mathcal{A}$ has a round complexity of $T(\Lambda)$ and requires messages of $O(|\mathcal{C}|\cdot B+\log n)$ bits, where $\mathcal{C}$ the color space and $B \geq 1$ is some parameter. Then, for every integer $r \geq 1$, there exists an $O(T(\Lambda)\cdot r)$-round deterministic distributed list coloring algorithm $\mathcal{A}'$ to solve list coloring instances with colors from the same color space and for which

$$\forall v \in V \,:\, \sum_{x \in L_v} (d_v(x)+1)^{1+\nu} = \beta_v^{1+\nu} \cdot \kappa(\Lambda)^r.$$

The algorithm $\mathcal{A}'$ requires messages of size $O(|\mathcal{C}|^{1/r}\cdot B+\log n)$.

As for Theorem 3, when replacing β_v by $\deg(v)$, Corollary 13 and Corollary 14 both also hold for the list defective coloring problem in undirected graphs.

5 Applying List Defective Colorings

In [5] and [14], Barenboim, and Fraigniaud, Heinrich, and Kosowski developed a technique to transform fast, but relaxed (oriented) list coloring into efficient algorithms for the (*degree*+1)-list coloring problem. The same technique has later also been used by the algorithms in [24, 4, 3]. The high-level idea of this transformation is as follows. Assume that for some $\alpha > 1$, we have a T-round algorithm $\mathcal{A}$ that solves list coloring instances with lists of size $> \alpha\Delta$ in graphs of maximum degree Δ. We can then first use a defective k-coloring to decompose the graph into k subgraphs of maximum degree $\leq \Delta/(2\alpha)$. One then iterates over those color classes and extends a given partial (*degree* + 1)-list coloring. When working on the nodes of some color class, all nodes that still have at least $\Delta/2$ uncolored neighbors also still have a list of size $> \Delta/2$. This is more than α times the maximum degree $\Delta/(2\alpha)$ in the current color class, and we can therefore color such nodes by using algorithm $\mathcal{A}$. In $k \cdot T$ rounds, we can therefore reduce the maximum degree of our (*degree* + 1)-list coloring problem from Δ to $\Delta/2$ and by repeating $O(\log \Delta)$ times, we can solve the (*degree* + 1)-list coloring problem. If the algorithm $\mathcal{A}$ works on directed graphs of maximum outdegree β and requires lists of size $> \alpha\beta$, the same idea also works if we decompose the graph by using an arbdefective coloring instead of a defective coloring.

The contribution of this section is two-fold. Firstly, we show that if we assume the existence of (oriented) list coloring algorithms that are significantly better than the current state of the art, we would directly obtain significantly faster algorithms for the standard $(\Delta + 1)$-coloring problem. Moreover, we show that by replacing the algorithm $\mathcal{A}$ in the description above by an (oriented) list defective coloring algorithm, the technique cannot only be used for the (*degree*+1)-list coloring problem, but it also works for computing arbdefective colorings and more generally list arbdefective colorings. In fact, it works for list arbdefective colorings with lists L_v and defects d_v such that for all $v \in V$, $\sum_{v \in V}(d_v(x) + 1) > \deg(v)$. In the following, we refer to such instances as (*degree*+1)-list arbdefective coloring instances. We subsequently assume that $\mathcal{A}^D_{\nu,\kappa}$ is a deterministic distributed list defective coloring algorithm that operates on undirected graphs and $\mathcal{A}^O_{\nu,\kappa}$ is a deterministic distributed oriented list defective coloring algorithm that operates on directed graphs. For real values $\nu \geq 0$ and $\kappa > 0$ we assume that $\mathcal{A}^D_{\nu,\kappa}$ and $\mathcal{A}^O_{\nu,\kappa}$ solve all (oriented) list defective coloring problems for which for all $v \in V$,

$$\sum_{x \in L_v} \big(d_v(x) + 1\big)^{1+\nu} \geq \deg(v)^{1+\nu} \cdot \kappa \text{ and} \tag{6}$$

$$\sum_{x \in L_v} \big(d_v(x) + 1\big)^{1+\nu} \geq \beta_v^{1+\nu} \cdot \kappa, \tag{7}$$

respectively. We assume that the round complexity of algorithm $\mathcal{A}^D_{\nu,\kappa}$ is $T^D_{\nu,\kappa}$ and that the round complexity of algorithm $\mathcal{A}^O_{\nu,\kappa}$ is $T^O_{\nu,\kappa}$. Theorem 4 in Section 1.1 shows that by using $\mathcal{A}^O_{\nu,\kappa}$, one can solve (*degree* + 1)-list arbdefective coloring instances in time $O\big(\Lambda^{\frac{\nu}{1+\nu}} \cdot \kappa^{\frac{1}{1+\nu}} \cdot \log(\Delta) \cdot T^O_{\nu,\kappa} + \log^* n\big)$ and by using $\mathcal{A}^D_{\nu,\kappa}$ one can solve such list arbdefective colorings in time $O\big(\Lambda^\nu \cdot \kappa^2 \cdot \log(\Delta) \cdot T^O_{\nu,\kappa} + \log^* n\big)$.

Implications of Theorem 4

We first discuss two immediate implications of Theorem 4, and we afterwards show how the theorem can be used to improve the best current deterministic complexity of the $(\Delta + 1)$-coloring problem in the CONGEST model.

Complexity of Computing (List) Arbdefective Colorings. For a first immediate implication of Theorem 4, we can use the algorithm of Theorem 2 as the oriented list defective coloring algorithm $\mathcal{A}^O_{\nu,\kappa}$. If we assume that the color space that we have is of size $|\mathcal{C}| = \text{poly}(\beta)$, in this case, $\nu = 1$ and $\kappa = O(\log\beta \cdot \log^3 \log\beta)$. This results in an arbdefective coloring algorithm that solves instances with lists L_v for which $\forall v \in V : \sum_{x\in L_v}(d_v(x)+1) > \deg(v)$ in time $O\big(\sqrt{\Lambda}\cdot\log^{5/2}\Delta\cdot\log^{3/2}\log\Delta + \log^* n\big)$. In particular, this implies that for any $d \geq 0$ and any $q > \frac{\Delta}{d+1}$, a d-arbdefective q-coloring can be computed in time $O\big(\sqrt{\frac{\Delta}{d+1}}\cdot\log^{5/2}\Delta\cdot\log^{3/2}\log\Delta + \log^* n\big)$, which significantly improves the previously best algorithms that achieves the same arbdefective coloring in time $O(\Delta + \log^* n)$ [2] or a more relaxed d-arbdefective $O\big(\frac{\Delta}{d+1}\big)$-coloring in time $O\big(\frac{\Delta}{d+1} + \log^* n\big)$. Note also that the condition $\forall v \in V : \sum_{x\in L_v}(d_v(x)+1) > \deg(v)$ is necessary in order to compute a (list) arbdefective coloring in time $f(\Delta) + O(\log^* n)$. If the condition does not hold, any deterministic algorithm for the problem requires at least $\Omega(\log_\Delta n)$ rounds [2].

Better List Defective Coloring Implies Better $(\Delta+1)$-Coloring. The theorem in particular also implies that certain progress on (oriented) list defective coloring algorithms would directly lead to faster algorithms for the standard $(\Delta+1)$-coloring problem. Assume that for an initial m-coloring of the graph, we have an oriented list defective coloring algorithm with a round complexity that is $\text{poly}(\Lambda) + O(\log^* m)$ and that satisfies equation (6) for any constant $\nu < 1$. In combination with Corollary 13, Theorem 4 then implies that we then obtain a $(\textit{degree}+1)$-list coloring (and thus $(\Delta+1)$-coloring) algorithm with a time complexity of $O\big(\Delta^{\frac{\nu}{1+\nu}+o(1)} + \log^* n\big)$, which would be polynomial improvement over the $O(\sqrt{\Delta\log\Delta} + \log^* n)$-round algorithm of [14, 10, 30]. The same would be true if we had a list defective coloring algorithm with a round complexity of $\text{poly}(\Lambda) + O(\log^* m)$ and that satisfies equation (7) for any constant $\nu < 1/2$. We believe that if it is possible to significantly improve the current best $O(\sqrt{\Delta\log\Delta} + \log^* n)$-round of $(\Delta+1)$-coloring, the key will be to better understand the distributed complexity of (oriented) defective colorings and probably also of the more general (oriented) list defective colorings.

Complexity of $(\Delta+1)$-Coloring in the CONGEST Model. Apart from the standard $(\Delta+1)$-coloring problem, in the following, we also consider the general $(\textit{degree}+1)$-list coloring problem. In order to keep the results simple and because this is the most interesting case, we will assume that we have $(\textit{degree}+1)$-list coloring instances with a color space of size at most $\text{poly}(\Delta)$. Note that in the case of the standard $(\Delta+1)$-coloring problem, the color space is of size $\Delta+1$. For small Δ, the best $(\Delta+1)$-coloring algorithm in the LOCAL model has a round complexity of $O(\sqrt{\Delta\log\Delta} + \log^* n)$ [14, 10, 30] and Theorem 5 in Section 1.1 states that this round complexity can almost be matched in the CONGEST model.

References

1 Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In *Proc. 30th Symp. on Foundations of Computer Science (FOCS)*, pages 364–369, 1989. doi:10.1109/SFCS.1989.63504.

2 Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed Δ-coloring plays hide-and-seek. *arXiv preprint arXiv:2110.00643*, 2021.

3 Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed edge coloring in time polylogarithmic in Δ. In *Proc. 41st ACM Symp. on Principles of Distributed Computing (PODC)*, pages 15–25, 2022.

4 Alkida Balliu, Fabian Kuhn, and Dennis Olivetti. Distributed edge coloring in time quasi-polylogarithmic in delta. In *Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 289–298, 2020.

5 Leonid Barenboim. Deterministic (δ+1)-coloring in sublinear (in δ) time in static, dynamic, and faulty networks. *Journal of ACM*, 63(5):1–22, 2016. doi:10.1145/2979675.

6 Leonid Barenboim and Michael Elkin. Distributed (delta+1)-coloring in linear (in delta) time. In *Proc. 41st Annual ACM Symp. on Theory of Computing (STOC)*, pages 111–120, 2009.

7 Leonid Barenboim and Michael Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. *Distributed Comput.*, 22:363–379, 2010. doi:10.1007/s00446-009-0088-2.

8 Leonid Barenboim and Michael Elkin. Deterministic distributed vertex coloring in polylogarithmic time. *Journal of ACM*, 58:23:1–23:25, 2011. doi:10.1145/2027216.2027221.

9 Leonid Barenboim and Michael Elkin. Distributed deterministic edge coloring using bounded neighborhood independence. In *Proc. 30th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 129–138, 2011.

10 Leonid Barenboim, Michael Elkin, and Uri 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.

11 Leonid Barenboim, Michael Elkin, and Fabian Kuhn. Distributed (Δ+1)-Coloring in Linear (in Δ) Time. *SIAM Journal on Computing*, 43(1):72–95, 2014. doi:10.1137/12088848X.

12 Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The Locality of Distributed Symmetry Breaking. *Journal of the ACM*, 63(3):1–45, 2016. doi:10.1145/2903137.

13 Yi-Jun Chang, Wenzheng Li, and Seth Pettie. An optimal distributed (Δ+1)-coloring algorithm? In *Proc. 50th ACM Symp. on Theory of Computing, (STOC)*, pages 445–456, 2018. doi:10.1145/3188745.3188964.

14 Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local conflict coloring. In *Proc. 57th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 625–634, 2016.

15 Marc Fuchs and Fabian Kuhn. List defective colorings: Distributed algorithms and applications. *CoRR*, abs/2304.09666, 2023. doi:10.48550/arXiv.2304.09666.

16 Mohsen Ghaffari, Christoph Grunau, Bernhard Haeupler, Saeed Ilchi, and Václav Rozhon. Improved distributed network decomposition, hitting sets, and spanners, via derandomization. In *Proc. 34th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2023.

17 Mohsen Ghaffari, Christoph Grunau, and Václav Rozhon. Improved deterministic network decomposition. In *Proc. 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2904–2923, 2021. doi:10.1137/1.9781611976465.173.

18 Mohsen Ghaffari and Fabian Kuhn. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. In *Proc. 62nd IEEE Symp. on Foundations of Computing (FOCS)*, 2021.

19 A.V. Goldberg, S.A. Plotkin, and G.E. Shannon. Parallel symmetry-breaking in sparse graphs. *SIAM Journal on Discrete Mathematics*, 1(4):434–446, 1988.

20 Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Tigran Tonoyan. Efficient randomized distributed coloring in CONGEST. In *Proc. 53rd ACM Symp. on Theory of Computing (STOC)*, pages 1180–1193, 2021.

21 Magnús M. Halldórsson, Fabian Kuhn, Alexandre Nolin, and Tigran Tonoyan. Near-optimal distributed degree+1 coloring. *CoRR*, abs/2112.00604, 2021.

22 David G. Harris, Johannes Schneider, and Hsin-Hao Su. Distributed (Δ +1)-coloring in sublogarithmic rounds. *J. ACM*, 65(4):19:1–19:21, 2018.

23 Fabian Kuhn. Local weak coloring algorithms and implications on deterministic symmetry breaking. In *Proc. 21st ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, 2009.

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

25 Fabian Kuhn and Roger Wattenhofer. On the complexity of distributed graph coloring. In *Proc. 25th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 7–15, 2006.

26 Nathan Linial. Distributive graph algorithms – Global solutions from local data. In *Proc. 28th Symp. on Foundations of Computer Science (FOCS)*, pages 331–335. IEEE, 1987. doi:10.1109/SFCS.1987.20.

27 L. Lovász. On decompositions of graphs. *Studia Sci. Math. Hungar.*, 1:237–238, 1966.

28 Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986. doi:10.1137/0215074.

29 Yannic Maus. Distributed graph coloring made easy. In *Proc. 33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2021.

30 Yannic Maus and Tigran Tonoyan. Local conflict coloring revisited: Linial for lists. In *Proc. 34th Symp. on Distributed Computing (DISC)*, volume 179 of *LIPIcs*, pages 16:1–16:18, 2020.

31 Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM Journal on Discrete Mathematics*, 4(3):409–412, 1991. doi:10.1137/0404036.

32 Alessandro Panconesi and Aravind Srinivasan. On the complexity of distributed network decomposition. *Journal of Algorithms*, 20(2):356–374, 1996.

33 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000. doi:10.1137/1.9780898719772.

34 Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proc. 52nd ACM Symp. on Theory of Computing (STOC)*, pages 350–363, 2020.

35 Johannes Schneider and Roger Wattenhofer. A new technique for distributed symmetry breaking. In *Proc. 29th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 257–266, 2010. doi:10.1145/1835698.1835760.

36 Mario Szegedy and Sundar Vishwanathan. Locality based graph coloring. In *Proc. 25th ACM Symp. on Theory of Computing (STOC)*, pages 201–207, 1993.

A Main Oriented List Defective Coloring Algorithm

If we have an OLDC instance in which some nodes have colors with constant defect requirements, the number of h of γ-classes can be $\Theta(\beta)$. Because also the value of $\tau(h, |\mathcal{C}|, m)$ is linear in h, this means that even if $g = 0$ and even if $|\mathcal{C}|$ and m are both polynomial in β, the condition in Lemma 12 is of the form $\sum_{x \in L_v} (d_v(x) + 1)^2 \geq \alpha \beta_v^2 \log^2 \beta$. One of the $\log \beta$ factors comes from the fact that at the very beginning of the algorithm, every node v reduces its color list to a list in which all colors have approximately the same defect value. In the following, we show that at the cost of a more complicated algorithm, we can improve this $\log \beta$ factor to a factor of the form $\operatorname{poly} \log \log \beta$. In the following discussion, we assume that $g = 0$, but we note that along the way, we will have to use Lemma 12 with positive g as a subroutine.

In order to obtain the improvement, we first want an algorithm where for computing the 0-round problem P_2, a node v of some γ-class only needs to compete with outneighbors of the same γ-class. For this, we use an iterative approach to solve P_2 and P_1. For each $i \in [h]$, let $V_i \subseteq V$ be the set of nodes in γ-class i. For each node $v \in V_i$ that is colored with some color x, we will make sure that v has at most $d_v(x)/4$ outneighbors of color x in γ-classes j for $j < i$, at most $d_v(x)/4$ outneighbors of color x in the same γ-class and that v has at most $d_v/2$ outneighbors in γ-classes j for $j > i$. Thus, we assume that each node $v \in V$ only uses the part $L_{v,i}$ of its L_v consisting of colors with a defect d_v such that $\gamma_v = 2^i \geq 4\beta_v/(d_v + 1)$. We then iterate over the γ classes $i \in [h]$ in increasing order. In

iteration i, we solve problems P_2 and P_1 for the nodes in V_i. When dealing with nodes in V_i, we can therefore assume that for all outneighbors in $u \in V_j$ for $j < i$, the list C_u (i.e., the output of problem P_1) is already computed. We then remove each color x from the list $L_{v,i}$ for which there are more than $d_v(x)/4$ outneighbors $u \in V_1 \cup \dots \cup V_{i-1}$ for which $x \in C_u$. In this way, we guarantee that v cannot choose a color with defect more than $d_v/4$ to outneighbors in lower γ-classes even before solving P_2 for node v. We can then solve P_2 and P_1 by only considering outneighbors in V_i. Of course, we have to make sure that even after removing colors from $L_{v,i}$, the list of v is still sufficiently large to solve problem P_2. The advantage of only having to consider neighbors in V_i when solving P_2 and P_1 is that in the condition on $\sum_{x\in L_{v,i}}(d_v(x)+1)^2$, we can replace the outdegree β_v of v by the number of outneighbors $\beta_{v,i}$ that v has in V_i. This gives us more flexibility in the choice of v's γ-class i. If $\sum_{x\in L_{v,i}}(d_v(x)+1)^2$ is large, v can choose γ-class i and tolerate many outneighbors in the same γ-class and if $\sum_{x\in L_{v,i}}(d_v(x)+1)^2$ is small, v can only choose γ-class i if a small number of outneighbors choose γ-class i. We will see that the problem of choosing a good γ-class can be phrased as an OLDC problem that can be solved by using Lemma 12 with appropriate parameters. The following technical Lemma 15 assumes that the γ-classes are already assigned and it formally proves under which conditions the above algorithmic idea allows to solve a given OLDC instance.

▶ **Lemma 15.** *Let $G = (V, E)$ be a directed graph that is equipped with an initial proper m-coloring. Let $h \geq 1$ be an integer parameter and assume that every node $v \in V$ is in some γ-class $i_v \in [h]$. For every $i \in [h]$, let V_i be the nodes in γ-class i and let $\beta_{v,i}$ be the number of outneighbors of v in V_i. Each node v has a color list $L_v \subseteq \mathcal{C}$ and one fixed defect value d_v, i.e., $d_v(x) = d_v$ for all $x \in L_v$. We further define $\tau := \tau(h, \mathcal{C}, m)$ and some integer parameter $q \in [\tau]$. We assume that for all $v \in V$*

$$\forall v \in V : \frac{4 \cdot \max\left\{\beta_{v,i_v}, \frac{\beta_v}{q}\right\}}{d_v + 1} \leq 2^{i_v} \quad \text{and} \quad |L_v| \geq \left[\alpha \cdot 4^{i_v} + \frac{4}{d_v+1} \cdot \sum_{j=i_v-\lfloor \log q \rfloor}^{i_v-1} \beta_{v,j} \cdot 2^j\right] \cdot \tau$$

for a sufficiently large constant $\alpha > 0$. Then there is an $O(h)$-round algorithm that assigns each node v a color $x \in L_v$ such that every node $v \in V$ has at most d_v outneighbors of color x. The algorithm requires messages consisting of $O(\min\{\Lambda \log|\mathcal{C}|, |\mathcal{C}|\} + \log\log\beta + \log m)$ bits.

We are now ready to prove Theorem 2, our main contribution. In the following $\hat{\beta}_v$ is the outdegree of v rounded up to the next integer power of 2. Further $\hat{\beta} := \max_{v\in V} \hat{\beta}_v$. Note that for all v, we have $\hat{\beta}_v \leq 2\beta_v$. The following lemma is a rephrasing of the theorem. By adjusting the constant α, Theorem 2 follows from Lemma 16 because for $h = \lceil \log \hat{\beta} \rceil$ and $h' = \lceil \log 4h \rceil$, $\tau(h, \mathcal{C}, m) = O(\log\beta + \log\log|\mathcal{C}| + \log\log m)$ and $\tau(h', [h], m) = O(\log\log\beta + \log\log m)$.

▶ **Lemma 16.** *Let $G = (V, E)$ be a properly m-colored directed graph and let $h := \lceil \log \hat{\beta} \rceil$ and $h' := 4^{\lceil \log_4 \log 8h \rceil}$. Assume that we are given an OLDC instance on G for which*

$$\forall v \in V : \sum_{x\in L_v} \big(d_v(x)+1\big)^2 \geq \alpha^2 \cdot \hat{\beta}_v^2 \cdot \tau \cdot \bar{\tau} \cdot h'^2, \tag{8}$$

where $\tau = 4^{\lceil \log_4 \tau(h,\mathcal{C},m) \rceil}$ and $\bar{\tau} = 4^{\lceil \log_4 \tau(h',[h],m) \rceil}$. Then, there is a deterministic distributed algorithm that solves this OLDC instance in $O(\log\beta)$ rounds using $O\big(\min\{|\mathcal{C}|, \Lambda \cdot \log|\mathcal{C}|\} + \log\beta + \log m\big)$-bit messages.

Proof. First note that w.l.o.g., we can assume that for all v and all $x \in L_v$, $(d_v(x)+1)^2$ and α are both integer powers of 4. We can just round up α and round down $d_v(x)$ to the next value for which this is true. We then just need to choose the constant α slightly larger. With those assumptions, the right-hand side of (8) is then a integer power of 4. For each node v, we define $R_v := \alpha \cdot \hat{\beta}_v^2 \cdot \bar{\tau} \cdot h'^2$. For every $v \in V$ and every $x \in L_v$, we then have $\frac{R_v}{(d_v(x)+1)^2} = 4^\mu$ for some $\mu \in [h]$. We can therefore partition each list L_v in to lists $L_v = L_{v,1} \cup \dots \cup L_{v,h}$ such that for all $\mu \in [h]$, $L_{v,\mu}$ consists of the colors $x \in L_v$ for which $\frac{R_v}{(d_v(x)+1)^2} = 4^\mu$. The algorithm to solve the given OLDC instance consists of two phases. In the first phase, every node $v \in V$ chooses its γ-class, which is an integer $i_v \in [h]$. In the second phase, we then use Lemma 15 to solve the OLDC instance.

We first discuss the objective of the first phase and we consider some node v. For every $\mu \in [h]$, we define $D_{v,\mu} := \sum_{x \in L_{v,\mu}} (d_v(x)+1)^2$ and $D_v := \sum_{\mu=1}^{h} D_{v,\mu} = \sum_{x \in L_v} (d_v(x)+1)^2$. For each $\mu \in [h]$, we further define $\lambda_{v,\mu} \in (0,1]$ as follows

$$\lambda_{v,\mu} := \begin{cases} 0 & \text{if } D_{v,\mu}/D_v < 1/(2h) \\ 4^{\lfloor \log_4(D_{v,\mu}/D_v) \rfloor} & \text{otherwise.} \end{cases}$$

In the next part, we make a case distinction and first assume that $\lambda_{v,\mu} < 1/4$ for all μ. The case when there is some μ with $\lambda_{v,\mu} \geq 1/4$ is a simple case that we discuss later.

Case I: $\forall \mu \in [h] : \lambda_{v,\mu} < 1/4$. Note that the definition of $\lambda_{v,\mu}$ implies that for all μ where $\lambda_{v,\mu} \neq 0$, $\lambda_{v,\mu} > 1/(8h)$ and $\lambda_{v,\mu} = 4^{-r_{v,\mu}}$ for some integer $r_{v,\mu} \in \{0, \dots, \lceil \log_4 4h \rceil\}$. Note also that the values of $D_{v,\mu}$ for which $\lambda_{v,\mu} = 0$ sum up to at most $D_v/2$ and therefore

$$\sum_{\mu=1}^{h} \lambda_{v,\mu} \geq \frac{1}{8}.$$

For every $v \in V$, we next define a function $f_v : [h] \to [h] \cup \{\perp\}$ such that for all $\mu \in [h]$, $f_v(\mu) = \mu - r_{v,\mu} + 2$ if $\lambda_{v,\mu} > 0$ and $f_v(\mu) = \perp$ otherwise. Note that $0 < \lambda_{v,\mu} < 1/4$ implies that $f_v(\mu) \leq h$. For every $\mu \in [h]$, we next also define a second function $i_v : [h] \to [h] \cup \{\perp\}$ as follows. For every μ, we set $i_v(\mu) = f_v(\mu)$ if $f_v(\mu) = \perp$ or if $f_v(\mu) \geq 1$ and there is no $\mu' < \mu$ for which $f_v(\mu') = f_v(\mu)$. Otherwise, we set $i_v(\mu) = \perp$. Note that for any two $\mu, \mu' \in [h]$ with $\mu \neq \mu'$, we either have $i_v(\mu) = i_v(\mu') = \perp$ or we have $i_v(\mu) \neq i_v(\mu')$. We next show that

$$\sum_{\mu \in [h] : i_v(\mu) \neq \perp} \lambda_{v,\mu} \geq \frac{2}{3} \cdot \sum_{\mu \in [h] : f_v(\mu) \neq \perp \wedge f_v(\mu) \geq 1} \lambda_{v,\mu} \geq \frac{2}{3} \cdot \left(\frac{1}{8} - \frac{1}{48} \right) \geq \frac{1}{20}. \tag{9}$$

To see this, consider some μ for which $\lambda_{v,\mu} > 0$. Note that for $f_v(\mu) < 1$, we need $r_{v,\mu} \geq \mu + 2$ and therefore $\lambda_{v,\mu} \leq 4^{-\mu-2}$. Thus, the sum over those $\lambda_{v,\mu}$ is at most $\frac{4}{3} \cdot 4^{-1-2} = 1/48$. It therefore remains to show that the sum over the $\lambda_{v,\mu}$ for which $f_v(\mu) \geq 1$, but $i_v(\mu) = \perp$ is at most a third the sum over the $\lambda_{v,\mu}$ for which $f_v(\mu) \geq 1$. We have $i_v(\mu) = \perp$ and $f_v(\mu) \geq 1$ iff there is a $\mu' < \mu$ for which $f_v(\mu') = f_v(\mu)$. For $f_v(\mu') = f_v(\mu)$, we need to have $\lambda_{v,\mu} = \lambda_{v,\mu'} \cdot 4^{\mu'-\mu}$. Consider some value $z \geq 1$ for which there is a value μ_z with $f_v(\mu_z) = z$ and assume that μ_z is the smallest such value. The sum over all λ_μ for $\mu > \mu_z$ and $f_v(\mu) = f_v(\mu_z)$ is at most $\sum_{\mu=\mu_z+1}^{\infty} \lambda_{\mu_z} \cdot 4^{\mu_z - \mu} = \lambda_{\mu_z}/3$. This concludes the proof of Inequality (9).

In order to assign a γ-class i_v to every node $v \in V$, we define another (generalized) OLDC instance. For this instance, the "color" list of node v is $\mathcal{L}_v = \{i_v(\mu) : i_v(\mu) \neq \perp\}$. For every $i \in \mathcal{L}_v$, we define the inverse function $\mu_v(i)$ to be the value μ for which $i_v(\mu) = i$. For each color $i \in \mathcal{L}_v$, we then define a defect $\delta_{v,i}$ as

$$\delta_{v,i} := \left\lfloor \sqrt{\lambda_{v,\mu_v(i)} \cdot R_v} \right\rfloor .$$

We further define $q := h$ and $g := \lfloor \log h \rfloor$. We then want to find an assignment of values $i_v \in \mathcal{L}_v$ to each node such that for every $v \in V$, the number of outneighbors u for which $i_u \in [i_v - g, i_v]$ is at most δ_{v,i_v}. We next show that such an assignment of γ-classes i_v satisfies the requirement needed by Lemma 15 and we can therefore use it to efficiently solve the original OLDC instance. We afterwards show that the generalized OLDC instance to find the values i_v satisfies the requirement of Lemma 12.

Let us therefore assume that we have an assignment of γ-class i_v to the nodes that solve the above generalized OLDC problem. For each $i \in [h]$, we again use V_i to denote the set of nodes v with $i_v = i$ and we assume $\beta_{v,i}$ is the number of outneighbors of v in V_i. The fact that v has at most δ_{v,i_v} outneighbors u with $i_u \in [i_v - g, i_v]$ implies that $\delta_{v,i_v} \geq \beta_{v,j}$ for all $j \in [i_v - g, i_v]$. For all $v \in V$, we have

$$\delta_{v,i_v} = \left\lfloor \sqrt{\lambda_{v,\mu_v(i)} \cdot R_v} \right\rfloor \overset{(\lambda_{v,\mu_v(i)} \geq 1/(8h))}{\geq} \left\lfloor \sqrt{\frac{R_v}{8h}} \right\rfloor = \left\lfloor \sqrt{\frac{\alpha \cdot \hat{\beta}_v^2 \cdot \bar{\tau} \cdot h'^2}{8h}} \right\rfloor \geq \frac{\hat{\beta}_v}{h}.$$

The last inequality follows because $h, \tau, \bar{\tau}, h' \geq 1$ and if we choose $\alpha \geq 8$. For the following calculations, we define $\mu_v := \mu_v(i_v) = i_v + r_{v,\mu} - 2$. Note that if v chooses γ-class i_v, it uses the colors in L_{v,μ_v}. All those colors have a defect d_v such that $(d_v + 1)^2 = R_v / 4^{\mu_v}$. Using $q = h$, we therefore have

$$\begin{aligned}
\frac{4 \cdot \max\left\{\beta_{v,i_v}, \frac{\beta_v}{q}\right\}}{d_v + 1} &\leq \frac{4 \cdot \delta_{v,i_v}}{d_v + 1} \\
&\leq \sqrt{\frac{16 \lambda_{v,\mu_v} \cdot R_v}{(d_v + 1)^2}} \\
&= \sqrt{16 \lambda_{v,\mu_v} \cdot 4^{\mu_v}} \\
&= \sqrt{4^2 \cdot 4^{-r_{v,\mu_v}} \cdot 4^{i_v + r_{v,\mu_v} - 2}} \;=\; 2^{i_v}.
\end{aligned}$$

The first part of the requirement of Lemma 15 is therefore satisfied. For the second part, recall that v uses the colors in L_{v,μ_v} and that $D_{v,\mu_v} = \sum_{x \in L_{v,\mu_v}} (d_v(x) + 1)^2 = |L_{v,\mu_v}| \cdot (d_v + 1)^2$, where d_v is defined as before. We have

$$|L_{v,\mu_v}| \geq \frac{\lambda_{v,\mu_v} D_v}{(d_v + 1)^2} \geq \frac{\lambda_{v,\mu_v} \cdot \alpha\tau \cdot R_v}{(d_v + 1)^2} = 4^{-r_{v,\mu_v}} \cdot \alpha\tau \cdot 4^{\mu_v} = \frac{\alpha}{16} \cdot 4^{i_v} \cdot \tau. \tag{10}$$

Before we continue, we switch to Case II.

Case II: $\exists \mu \in [h] : \lambda_{v,\mu} \geq 1/4$. Before looking of the problem of assigning the γ-classes, we have a look at the requirements for Lemma 15 in Case II, i.e., if there is a $\mu \in [h]$ for which $\lambda_{v,\mu} \geq 1/4$. Let μ_v be one such value μ. In this case, we set $i_v = \mu_v$, $\mathcal{L}_v = \{i_v\}$, and $\delta_{v,i_v} := \lfloor \sqrt{R_v}/4 \rfloor$. We then have $\delta_{v,i_v} = \left\lfloor \sqrt{\alpha \hat{\beta}_v^2 \bar{\tau} h'^2 / 16} \right\rfloor \geq \beta_v$. The last inequality holds if $\alpha \geq 16$ because $\bar{\tau}$ and h' are positive integers. We therefore clearly have $\delta_{v,i_v} \geq \max\{\beta_{v,i_v}, \beta_v/q\}$ and therefore

$$\frac{4 \cdot \max\left\{\beta_{v,i_v}, \frac{\beta_v}{q}\right\}}{d_v + 1} \leq \frac{4\delta_{v,i_v}}{d_v + 1} \leq \sqrt{\frac{16 R_v}{16 (d_v + 1)^2}} = 2^{\mu_v} = 2^{i_v}.$$

Hence, the first part of the requirement of Lemma 15 also holds in Case II. Similarly to Case I, we can lower bound the size of the color list L_{v,μ_v} that is used by v:

$$|L_{v,\mu_v}| \geq \frac{\lambda_{v,\mu_v} D_v}{(d_v+1)^2} \geq \frac{\alpha\tau \cdot R_v}{4(d_v+1)^2} = \frac{\alpha}{4} \cdot 4^{i_v} \cdot \tau. \tag{11}$$

The bound given by (10) therefore also holds in Case II.

We now continue considering both cases together. It remains to show (to apply Lemma 15) that

$$|L_{v,\mu_v}| \geq \alpha' \cdot 4^{i_v} \cdot \tau + \frac{4}{d_v+1} \cdot \sum_{j=i_v-\lfloor \log q \rfloor}^{i_v-1} \beta_{v,j} \cdot 2^j \cdot \tau \tag{12}$$

for some constant α' that can be chosen as large as needed by choosing the constant α sufficiently large. Note that we have $g = \lfloor \log q \rfloor$ and thus for $j \in [i_v - \lfloor \log q \rfloor, i_v]$, $\beta_{v,j} \leq \delta_{v,i_v}$. We therefore have

$$\frac{4}{d_v+1} \cdot \sum_{j=i_v-\lfloor \log q \rfloor}^{i_v-1} \beta_{v,j} \cdot 2^j \cdot \tau \leq \frac{4\delta_{v,i_v}\tau}{d_v+1} \cdot \sum_{\ell=1}^{g} 2^{i_v-\ell} < \frac{4\delta_{v,i_v}}{d_v+1} \cdot 2^{i_v} \cdot \tau.$$

We have already seen that $4\delta_{v,i_v}/(d_v+1) \leq 2^{i_v}$ and the bound in the above inequality can therefore be upper bounded by $4^{i_v}\tau$. For every constant $\alpha' > 0$, we can therefore choose a constant $\alpha > 0$ such that the bound in (12) is upper bounded by the bound in (10). This shows that if v is in Case I, node v satisfies the requirements to apply Lemma 15.

We next also show that the assignment of γ-classes i_v can be done by using the algorithm of Lemma 12. Recall that every node v needs to pick an $i_v \in \mathcal{L}_v$ such that the total number of outneighbors u that pick $i_u \in [i_v - g, i_v]$ is at most δ_{v,i_v}. To apply Lemma 12, we have to lower bound $\sum_{i\in\mathcal{L}_v}(\delta_{v,i}+1)^2$. We have

$$\begin{aligned}
\sum_{i\in\mathcal{L}_v}(\delta_{v,i}+1)^2 &\geq \min\left\{\frac{1}{16}, \sum_{i\in\mathcal{L}_v}\lambda_{v,\mu_v(i)}\right\} \cdot R_v \\
&\overset{(9)}{\geq} \frac{1}{20} \cdot R_v \\
&= \frac{1}{20} \cdot \alpha \cdot \hat{\beta}_v^2 \cdot \bar{\tau} \cdot h'^2.
\end{aligned}$$

Note that we have $g = \lfloor \log h \rfloor$ and $h' \geq \log(8h)$. We therefore have $2h' \geq 2g+1$. By choosing a sufficiently large constant α, the above inequality therefore implies that the requirements of Lemma 12 are satisfied as long as the value of $\bar{\tau}$ is sufficiently large. We have $\bar{\tau} = \tau(h', [h], m)$. Note that the color space of the OLDC problem that we use to assign the values i_v is $[h]$. Recall that for all v and μ, $\lambda_{v,\mu} = 0$ or $\lambda_{v,\mu} \geq 1/(8h)$. For each node $v \in V$, we therefore have

$$\min_{i\in\mathcal{L}_v}\delta_{v,i} \geq \min\left\{\frac{\sqrt{R_v}}{4}, \sqrt{\lambda_{v,\mu_v(i)}R_v}\right\} \geq \min\left\{\frac{1}{4}, \frac{1}{\sqrt{8h}}\right\} \cdot \sqrt{R_v} \geq \frac{\sqrt{R_v}}{8h} \geq \frac{\beta_v}{8h}.$$

We therefore have

$$\max_{v\in V} \frac{\beta_v}{\min_{i\in\mathcal{L}_v}\delta_{v,i}+1} \leq 8h.$$

Because we have $h' \geq \log(8h)$, the choice $\bar{\tau} = \tau(h', [h], m)$ satisfies the requirements of Lemma 12 and we can therefore compute the γ-classes i_v for all nodes v by using the algorithm of Lemma 12.

We will now analyze the required message size and round complexity of the algorithm. In the first phase the OLDC problem on color lists $\mathcal{L}_v$ and defects δ_{v,i_v} has to be solved. As $|\mathcal{L}_v| \leq h$ and transmitting such a single defect does not need more than $\log h$ bits, by Lemma 12, solving this OLDC instance the maximum message size is $O(h + \log h + \log m) = O(h + \log m)$ bits. The number of rounds needed for this first phase are $O(h')$. In the second phase we have messages of size $O(\min\{|\mathcal{C}| + \Lambda \log |\mathcal{C}|\} + \log\log \beta)$ (the initial color is already known in the second phase) due to Lemma 12. The round complexity of the second phase is $O(h)$ due to Lemma 15. Combining both phases and using that $h' = O(h) = O(\log \beta)$, the maximum message size and the runtime are as stated. ◀