# Distributed Graph Coloring Made Easy

YANNIC MAUS, TU Graz, Austria

In this article, we present a deterministic CONGEST algorithm to compute an $O(k\Delta)$-vertex coloring in $O(\Delta/k) + \log^* n$ rounds, where $\Delta$ is the maximum degree of the network graph and $k \geq 1$ can be freely chosen. The algorithm is extremely simple: each node locally computes a sequence of colors and then it *tries colors* from the sequence in batches of size $k$. Our algorithm subsumes many important results in the history of distributed graph coloring as special cases, including Linial's color reduction [Linial, FOCS'87], the celebrated locally iterative algorithm from [Barenboim, Elkin, Goldenberg, PODC'18], and various algorithms to compute defective and arbdefective colorings. Our algorithm can smoothly scale between several of these previous results and also simplifies the state-of-the-art $(\Delta+1)$-coloring algorithm. At the cost of losing some of the algorithm's simplicity we also provide a $O(k\Delta)$-coloring algorithm in $O(\sqrt{\Delta/k}) + \log^* n$ rounds. We also provide improved deterministic algorithms for ruling sets, and, additionally, we provide a tight characterization for one-round color reduction algorithms.

CCS Concepts: • **Theory of computation** → **Distributed algorithms**; • **Mathematics of computing** → *Graph coloring;*

Additional Key Words and Phrases: LOCAL model, CONGEST model, distributed graph coloring

**ACM Reference format:**
Yannic Maus. 2023. Distributed Graph Coloring Made Easy. *ACM Trans. Parallel Comput.* 10, 4, Article 21 (December 2023), 21 pages.
https://doi.org/10.1145/3605896



## 1 INTRODUCTION

In the *C-vertex coloring* problem the objective is to assign each vertex of an $n$-node graph $G = (V, E)$ one of $C$ colors such that adjacent vertices get different colors. In the distributed setting, graph coloring is considered to be one of the core *symmetry breaking problems* with a huge amount of published work and even a whole book almost exclusively covering the problem [13]. In this setting, the usual goal is to compute a $C$-coloring with $\Delta + 1 \leq C \leq O(\Delta^2)$, where $\Delta$ is the maximum degree of the graph. The bound of $\Delta+1$ stems from the fact that any graph can be colored with the respective number of colors, and this can even be done with a simple sequential greedy algorithm. The bound of $O(\Delta^2)$ colors stems from an algorithm in Linial's seminal paper in which he introduces one of the core models for distributed graph algorithms, i.e., the LOCAL model [50]. In this model, the graph abstracts a communication network in which the nodes communicate through the edges in synchronous rounds and at the end of the computation each node needs to output its

This project was partially supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 755839.
Author's address: Y. Maus, TU Graz, Inffeldgasse 16b/II, Graz, Austria, 8010; email: yannic.maus@ist.tugraz.at.


2329-4949/2023/12-ART21
https://doi.org/10.1145/3605896

own part of the solution, e.g., its own color. The complexity measure is the number of synchronous rounds. Linial gave an extremely fast $O(\Delta^2)$-coloring algorithm that only uses $O(\log^* n)$ rounds. Further, he showed that $\Omega(\log^* n)$ rounds are needed to color rings ($\Delta = 2$) with $O(1) = \Delta^{O(1)}$ colors. Due to the lower bound, a vast amount of published papers, e.g., [9, 14, 16, 29, 49, 51, 60], study the setting that occurs after applying Linial's coloring algorithm; i.e., they ask: *Given an $O(\Delta^2)$-coloring, how fast can one reduce the number of colors where the runtime of the algorithm can only depend on $\Delta$?* The similar question of finding a fast algorithm with complexity $f(\Delta) + \log^* n$ is also sometimes referred to as determining the *truly local complexity* of a problem [51].

In the current article, we devise several results to advance the understanding of this setting. First, we provide a *simple* deterministic algorithm that scales between the two extremes of $\Delta + 1$ (or rather $O(\Delta)$ colors) and $O(\Delta^2)$ colors. In particular, for a parameter $k \geq 1$ of the user's choice the algorithm computes a $O(k\Delta)$-coloring with complexity $f(\Delta) = O(\Delta/k)$. In the algorithm, each node $v$ uses its input color, e.g., provided by Linial's algorithm, to (locally) compute a sequence $p_v(0), p_v(1), \ldots$ of colors. Then, node $v$ *tries* to get colored with one of the first $k$ colors in its sequence by sending these color trials to its neighbors and receiving their trials. If $v$ tries a color $c$ that is not *conflicting* with the neighbors' trials node, $v$ gets permanently colored with color $c$; otherwise $v$ continues to the next round in which it tries the next $k$ colors in its sequence, and so on.

Second, we show that this simple *mother algorithm* either immediately yields the core steps of the aforementioned papers, e.g., the algorithms in [14, 50], or can be mildly adapted to obtain crucial subroutines developed or used in [9, 10, 14, 16, 29, 45, 51]; e.g., a mild adaptation yields *$d$-defective colorings,* which were the crucial ingredient in [10, 16, 45], or *arbdefective colorings,* which were crucial in [9, 14, 29, 51]. A core strength of our result is that the algorithms for each of these results are very similar. To get a feeling for this, let us look at defective colorings. In a *$d$-defective coloring* a node is allowed to have at most $d$ neighbors with the same color. Besides a suitable choice of $k$ and sequences $p_v(0), p_v(1), \ldots$, the only change is in the execution of the algorithm: when deciding whether to keep a color $c$, a node tolerates up to $d$ neighbors with the same color. This algorithm does not yet do the job as the defect of a node $v$ might be larger than $d$ at the end of the execution if one or more neighbors of $v$ choose the same color as $v$ in later rounds, but it still captures the essence of the adaptations that need to be performed.

Third, at the cost of losing some of the algorithm's simplicity, we show how to compute an $O(\Delta^{1+\varepsilon})$-coloring in $O(\Delta^{1/2-\varepsilon/2})$ rounds, and we improve the state-of-the-art runtime for computing so-called *ruling sets*.

Fourth, we provide a full characterization of one-round coloring algorithms. Informally, we determine the maximum number $q_{m,\Delta}$ of colors that can be reduced by a one-round coloring algorithm that works for any graph with a given maximum degree $\Delta$ and an input $m$-coloring.

Next, we describe why this result, combined with our $O(k\Delta)$-coloring algorithm in $O(\Delta/k)$ rounds, might be of additional interest. Already [44, 49, 60] studied one-round color reduction algorithms and showed lower bounds like our fourth contribution; then in [60] these one-round lower bounds on the number of colors were used to prove a heuristic $\Omega(\Delta \log \Delta)$ runtime lower bound for computing a $(\Delta + 1)$-coloring. As already pointed out by [60], the bound is *heuristic* in the following sense (the following example uses the result of our article): given a coloring with at most $2\Delta$ colors, we can reduce exactly one color in a single round. By applying this tight bound twice, one would wish to claim that one cannot go from a $2\Delta$ coloring to a $2\Delta - 2$ coloring in two rounds. However, this claim cannot be deduced via this method, as the second application of the one-round lower bound assumes that the *intermediate* $2\Delta - 1$ *coloring* is *worst case*. But, instead, a two-round algorithm might not produce an intermediate coloring at all, or it might output a very specific intermediate coloring that enables it to reduce more than one color in its second round. The heuristic lower bound in [60] is obtained by applying one-round lower bounds

iteratively, purposely ignoring this important subtlety. Since the publication of [60], at least five different algorithms for $(\Delta + 1)$-coloring that beat this lower bound were published: given an $O(\Delta^2)$-coloring, a $(\Delta + 1)$-coloring was computed in $O(\Delta)$ rounds [10, 16, 45], in $O(\Delta^{3/4})$ rounds in [9], in $O(\sqrt{\Delta \log \Delta} \log^* \Delta)$ rounds in [14, 29], and in $O(\sqrt{\Delta \log \Delta})$ rounds in [51]. Formally, only the *locally iterative* $O(\Delta)$-round algorithm by Barenboim et al. [14] beats this lower bound and we explain why. It starts with an $O(\Delta^2)$ coloring and maintains a feasible coloring in each round. The color of a node in the next round only depends on the node's own color and the colors of its neighbors. The algorithm was celebrated as it is significantly simpler than the aforementioned faster algorithms and it breaks the aforementioned heuristic lower bound in a *clean way*, due to maintaining a feasible coloring in each round.

While we formally do not maintain a feasible coloring in each round in our algorithms,[1] we provide a different insight. Our *tight* lower bounds for one-round algorithm provides a heuristic (false) argument that shows that one needs $\Omega(\Delta)$ rounds to go from a $O(\Delta^2)$ coloring to a $\Delta^2/5$-coloring (the 5 is chosen somewhat arbitrarily), and for a suitable $k = \Omega(\Delta)$, our mother algorithm provides a $O(\Delta/k) = O(1)$-round algorithm to perform such a color reduction. Thus, the heuristic lower bound that is based on the repeated application of *tight* one-round lower bounds can be beaten significantly in the number of colors by a simple $O(1)$-round algorithm. In contrast, all the previous algorithms that provide such an insight require $\Delta^{\Omega(1)}$ rounds. This is in particular interesting, as there has been almost no progress in proving lower bounds for the $(\Delta + 1)$-coloring problem since Linial's initial seminal $\Omega(\log^* n)$ lower bound. Just as we do in this article, the only other known lower bounds in the LOCAL model study one-round algorithms [44, 49, 60]. Our article suggests that even in constant-time coloring algorithms there are still results to be discovered. In fact, we do not even know a lower bound on the number of colors that a two-round algorithm must use. Surprisingly, there is not even a lower bound that rules out that one can go from a $O(\Delta^2)$ coloring to a $\Delta + 1$ coloring in two rounds. Further evidence that understanding constant-time or even just two-round algorithms is given by [51]. It provides a two-round algorithm for a *list coloring* variant of Linial's color reduction. In *list coloring* each node $v$ has a list $L(v)$ of colors and needs to output a color from this list. Basically, the authors show that one can compute a list coloring in two rounds if lists are of size $\tilde{\Omega}(\Delta^2)$—the "equivalent" of the $O(\Delta^2)$ colors in Linial's coloring—and interestingly for their choice of parameters the exact same problem cannot be solved in one round. Of course, just as applying a lower bound for one-round algorithms twice does not give a tight lower bound for two rounds, it is unclear whether understanding two-round algorithms will yield a result for the holy grail, a tight runtime lower bound for $(\Delta + 1)$-coloring.

### 1.1 Our Contributions

While we have already explained our contributions from a high-level point of view, we use this section to state them formally; additional related work is presented afterwards. Our results hold in the LOCAL model and in the CONGEST model (both lower and upper bounds).

*The* LOCAL *and* CONGEST *model of distributed computing [50, 55].* In both models the graph is abstracted as an $n$-node network $G = (V, E)$ with maximum degree at most $\Delta$. Communication happens in synchronous rounds. Per round, each node can send one message to each of its neighbors. At the end, each node has to know its own part of the output, e.g., its own color. In the LOCAL model there is no bound on the message size, and in the CONGEST model messages can contain at most $O(\log n)$ bits. Usually, in both models nodes are equipped with unique $O(\log n)$-bit IDs and

[1] It is straightforward to tweak the algorithms to actually achieve this at the cost of losing some of the simplicity of the algorithm, e.g., by encoding the state of a node into a proper vertex coloring.

initially, nodes know their own ID or their own color in an input coloring but are unaware of the IDs of their neighbors.

Linial's algorithm treats the unique IDs as an input coloring to compute an $O(\Delta^2)$-coloring in $O(\log^* n)$ rounds; merely in one *color reduction step* he reduces an input $m$-coloring to an $O(\Delta^2 \log^{O(1)} m)$-coloring, which then serves as the input coloring for the next step. All of our algorithms do not make use of unique IDs but work in the more general setting where nodes are only equipped with some input coloring with $m$ colors. Similarly to most previously known results, we assume that $m$ and $\Delta$ (and sometimes additional parameters) are global knowledge. It is easiest to grasp our results when setting $m = O(\Delta^2)$; that is, one first applies Linial's algorithm. For an integer $x$ we use $[x]$ to denote the set $\{0, \dots, x-1\}$; we use this notation frequently throughout this article. Our main technical result is the following theorem.

Theorem 1.1. *There exists a distributed deterministic algorithm that performs as follows in any undirected graph $G = (V, E)$:*

**Input:** *At every node $v \in V$, the algorithm takes as input an integer $m \geq 1$, a color $c_v \in [m]$ such that the colors of the vertices form an $m$-coloring of $G$, the maximum degree $\Delta$ of $G$, and two integers $d, k$, where $0 \leq d \leq \Delta - 1$ and $1 \leq k \leq X$ for $Z = \frac{\Delta}{(d+1)}$ and $X = 4 \cdot Z \cdot \lceil \log_Z m \rceil$.*

**Output:** *At every node $v \in V$, the algorithm outputs a color in $[kX]$ such that*

*(1) the set of monochromatic edges can be oriented such that no node is the source of more than $d$ of them, and*

*(2) the set of nodes can be partitioned into $R = \lceil \frac{X}{k} \rceil$-induced subgraphs $P_1, \dots, P_R$ such that every node is incident on at most $d$ monochromatic intra-part edges.*

*This algorithm runs in $R = \lceil \frac{X}{k} \rceil$ rounds in the* CONGEST *model. The orientation and the partition can be computed with no additional cost in the round complexity.*

Note that whenever $d \neq 0$, the coloring computed by the algorithm of Theorem 1.1 may not be proper; i.e., neighboring vertices may output the same color. Corollary 1.2 summarizes the most important parameter settings for Theorem 1.1, including settings to compute proper colorings ($d = 0$). While the algorithm for Theorem 1.1 is extremely simple (locally compute a permutation of the output colors, try them in batches of size $k$, and tolerate up to $d$ conflicts), the theorem, stated in its general form, has a rather technical appearance to fit various choices of parameters at once. But we believe that it is very approachable as soon as one considers precise choices for its parameters; e.g., we can first use Linial's algorithm (or an algorithm derived from Theorem 1.1) to compute a $O(\Delta^2)$-coloring in $O(\log^* n)$ rounds. If we treat this coloring as an input coloring with $m = O(\Delta^2)$ colors and if also $d = 0$ or $d = \Delta^\varepsilon$ holds for some constant $0 < \varepsilon < 1$, one can replace the term $\lceil \log_Z m \rceil$ with a constant.

One step of Linial's color reduction [50] is based on a suitable so-called *low-intersecting set family* $S_1, \dots, S_m$; Linial uses the probabilistic method to show that the respective families exist. In the algorithm a node with input color $i$ simultaneously tries all colors in $S_i$, and as $S_i \cap S_j$ is *small* for each neighbor's input color $j \neq i$, it is guaranteed that at least one color in $S_i$ is not tried by any neighbor. The low-intersecting set families obtained by the probabilistic method are not strong enough to go from a $\Delta^{O(1)}$-coloring to an $O(\Delta^2)$-coloring. Hence, Linial uses a different construction for such families, based on polynomials.[2] At its core is the fundamental theorem that for two distinct polynomials $p_1$ and $p_2$ with degree $d$ over a suitable finite set $\mathbb{F}_q$, the sets

[2]Szegedy and Vishwanathan show how to compute a coloring with $\Delta^{O(1)}$ colors in $0.5 \log^* n$ rounds; if this algorithm is followed by $O(1)$ iterations of Linial's color reduction based on polynomials, this implies an $O(\Delta^2)$-coloring algorithm in $0.5 \log^* n + O(1)$ rounds [60].

$S_i = \{(x, p_i(x)) \mid x \in \mathbb{F}_q\}$, $i = 1, 2$ intersect in at most $d$ elements; see [25, Example 3.2]. Choosing $m$ distinct polynomials yields the respective set family with $m$ sets. This argument is also the core of our main result; in particular, Linial's one-round color reduction is a special case of our more general Theorem 1.1. Next, we discuss various settings for the parameters in Theorem 1.1 and explain which results it subsumes. An approachable summary is contained in Corollary 1.2. If $d = 0$, the computed coloring is proper by point (1) of Theorem 1.1 and (2) can be ignored. The parameter $k$ trades the number of rounds versus the number of colors. For the extreme choice of $k = X = O(\Delta)$ we obtain the aforementioned color reduction by Linial (the one built on top of polynomials), and for $k = 1$ we obtain a generalization of the locally iterative algorithm of [14]. Other values of $k$ scale between both algorithms and provide an extremely simple way to compute $O(k \cdot \Delta)$-coloring in $O(\Delta/k)$ rounds. While our algorithm for $k = 1$ only computes an $O(\Delta)$-coloring in $O(\Delta)$ rounds, we can use an additional $O(\Delta)$ rounds, in each of which we remove a single color class to transform it into a $(\Delta + 1)$-coloring.

We now explain Theorem 1.1 in the case of $d > 0$. A *$\beta$-outdegree $c$-coloring* is a vertex coloring with $c$ colors together with an orientation of the edges between neighbors with the same color such that each node has at most $\beta$ outgoing edges. Note that the edges between vertices with different colors do not need to be oriented.[3] Consider the setting of $k = 1$ and $d = \beta = \Delta^\varepsilon$ for a constant $0 < \varepsilon < 1$. Due to the first condition of Theorem 1.1, we obtain a simple $\beta$-outdegree $O(\frac{\Delta}{\beta})$-coloring algorithm that runs in $O(\frac{\Delta}{\beta})$ rounds. Further, by assigning a vertex $v$ with output color $\phi(v)$ the color tuple $(\phi(v), i)$, where $i$ is the index of the subgraph $P_i$ that it belongs to in (2), we obtain a $d$-defective $O((\frac{\Delta}{d})^2)$-coloring in $O(\frac{\Delta}{d})$ rounds. For $k = X = O(\Delta)$, the same defective coloring can be computed in one round. This simplifies and subsumes several results in the literature.

A $\beta$-outdegree $O(\frac{\Delta}{\beta})$-coloring algorithm is one of the two crucial components in the state-of-the-art $(\Delta + 1)$-coloring algorithm in [51]. Our simpler algorithm to compute such a coloring thus also simplifies the overall algorithm; see Section 3.1 for details.

The next corollary summarizes the parameter settings in Theorem 1.1 that are most interesting with our current knowledge. In the future, other settings of parameters might be of interest.

Corollary 1.2. *There are the following deterministic CONGEST algorithms that compute the stated proper colorings in the stated runtimes on any input $\Delta^4$-colored graph with maximum degree $\Delta$, given a globally known parameter $k \geq 1$ (in 2. and 3.):*

*(1) $256\Delta^2$-coloring in one round (Linial's color reduction [50])*
*(2) $16\Delta \cdot k$-coloring in $O(\frac{\Delta}{k})$ rounds (subsumes results in [10, 16, 45])*
*(3) $\Delta^2$-coloring in $O(1)$ rounds*

*There are the following deterministic CONGEST algorithms that compute the stated improper colorings in the stated runtimes on any input $\Delta^4$-colored graph, given a globally known parameter $\beta = \Delta^\varepsilon$ (in 4.) or $d = \Delta^\varepsilon$ (in 5. and 6.) for a constant $0 < \varepsilon < 1$:*

*(4) $\beta$-outdegree $O(\frac{\Delta}{\beta})$-coloring in $O(\frac{\Delta}{\beta})$ rounds (subsumes a result in [14])*
*(5) $d$-defective $O((\frac{\Delta}{d})^2)$-coloring in one round (subsumes a result in [16, 45])*
*(6) $d$-defective $O((\frac{\Delta}{d})^2)$ coloring in $O(\frac{\Delta}{d})$ rounds (subsumes some results in [10, 16])*

The required input $\Delta^4$-coloring can be computed with Linial's algorithm for a sufficiently large constant $\Delta$. The precise choice of the constants in the $O$-notation in the defective coloring in

[3] These colorings with a bound on the outdegree are closely related to *arbdefective* colorings, which were introduced in [11] and have played a significant role in the development of sublinear in $\Delta$ algorithms (more details in [51]).

Corollary 1.2 depends linearly on the exponent $\varepsilon$. The algorithm of (6) is clearly inferior to the one in (5), as it computes a $d$-defective coloring with the same number of colors but is slower. We merely state (6) for its proof (see Section 2), which gives a slightly different perspective on Theorem 1.1.

We also provide algorithms that are faster than the previous state of the art.

THEOREM 1.3. *For any constant $\varepsilon > 0$, there is a deterministic* CONGEST *algorithm that computes a $O(\Delta^{1+\varepsilon})$-coloring in $O(\Delta^{1/2-\varepsilon/2}) + \log^* n$ rounds on any graph with maximum degree $\Delta$.*

Instantiating Theorem 1.3 with $\varepsilon = \log_\Delta k$ yields the following corollary.

COROLLARY 1.4. *For any $1 \leq k \leq \Delta$, there is a deterministic* CONGEST *algorithm that computes an $O(k\Delta)$-coloring in $O(\sqrt{\Delta/k}) + \log^* n$ rounds on any graph with maximum degree $\Delta$.*

*Ruling sets.* For an integer $r \geq 1$, a $(2, r)$-ruling set of a graph $G = (V, E)$ is a subset $S \subseteq V$ of the vertices that is an independent set and satisfies that for any vertex $v \in V$ there is a vertex $s \in S$ in hop distance at most $r$ [1]. Ruling sets and their extensions (larger distance between nodes in $S$) have played an important role as subroutines in several algorithms, e.g., [1, 24, 34, 54]. We provide a faster algorithm for $(2, r)$-ruling sets.

THEOREM 1.5. *For any constant integer $r \geq 2$, there is a deterministic* CONGEST *algorithm that computes a $(2, r)$-ruling set in $O(\Delta^{\frac{2}{r+2}}) + \log^* n$ rounds on any graph with maximum degree $\Delta$.*

The runtime guarantee of the previously fastest algorithm is $O(\Delta^{2/r}) + \log^* n$ rounds [57]. So, e.g., for $r = 2$ the $\Delta$-dependency improves from $O(\Delta)$ to $O(\sqrt{\Delta})$ and for $r = 3$ it improves from $O(\Delta^{2/3})$ to $O(\Delta^{2/5})$. For $r = 1$ the problem is equivalent to the *maximal independent set problem* and has a $\Omega(\Delta)$ lower bound, if the dependency on $n$ is limited to $O(\log^* n)$ rounds [3]. For (possibly non-constant) $r \geq 1$ there is a very recent lower bound of $\Omega(r\Delta^{1/r})$ rounds for the problem, even if an initial $O(\Delta)$ coloring is given [4]. Thus, the bound of Theorem 1.5 is tight for $r = 2$.

*Lower bounds for color reduction.* We give tight characterization for one-round color reduction algorithms, given an input $m$-coloring and no unique IDs.

THEOREM 1.6. *For any integer $\Delta \geq 1$ and $\Delta + 1 \leq m \leq \frac{\Delta^2}{4} + \frac{3\Delta}{2} + \frac{9}{4}$ let $1 \leq k \leq \frac{\Delta}{2} + \frac{3}{2}$ be the largest integer such that $m \geq k(\Delta - k + 3)$. Then, there is a one-round* CONGEST *algorithm that on any input $m$-colored graph with maximum degree $\Delta$ computes an $(m-k)$-coloring. Additionally, if also $k \leq \Delta - 2$ holds, then there is no one-round* LOCAL *algorithm that outputs a proper $(m - k - 1)$-coloring on every input $m$-colored graph with maximum degree $\Delta$.*

Theorem 1.6 roughly states that reducing $k$ colors requires $k\Delta - \Theta(k^2)$ input colors. For concrete choices of $k$ the bound in Theorem 1.6 says that to reduce one color one needs at least $\Delta + 2$ input colors, to reduce two colors one needs $2\Delta + 2$ input colors, to reduce three colors one needs $3\Delta$ input colors, to reduce four colors one needs $4\Delta - 4$ input colors, and so on.

The fastest randomized algorithms compute $O(\Delta)$ colorings in $O(\log^* n)$ rounds for $\Delta \geq \log^{O(1)} n$ [23, 58] and they can be adapted to also compute $(1+\varepsilon)\Delta$-colorings. However, it seems that the hardest part of $(\Delta + 1)$-coloring is to reduce a $(1 + \varepsilon)\Delta$ coloring to a $(\Delta + 1)$-coloring. We show that an algorithm with runtime $T$ that reduces an input coloring with $(1 + \varepsilon)\Delta$ colors to a $(\Delta + 1)$ coloring can be used with $O(\log_{1+\varepsilon} \Delta)$ overhead to reduce a $O(\Delta^2)$-coloring to a $(\Delta + 1)$-coloring. If $T = \Delta^{\Omega(1)}$, there is only a constant factor overhead (for details see Section 5).

### 1.2 Related Work

The state of the art for $(\Delta + 1)$-coloring when the runtime is expressed as $f(\Delta) + \log^* n$ is $O(\sqrt{\Delta \log \Delta}) + \log^* n$ rounds and given by [51]. Just like the slightly slower algorithms [9, 14, 29],

the result of [51] works for the more general $(deg + 1)$-list coloring problem in which the size of the list of each node exceeds its degree. The result in [29] applies for the even more general *local conflict coloring problem* in which one can specify for each edge of the graph which colors are not allowed to be adjacent. For an extensive overview on algorithms whose runtime is $f(\Delta) + \log^* n$ as well as an overview on the influence of arbdefective colorings during the last decade we refer to the related work section in [51]. Further, almost all published papers until 2013 are discussed in the excellent monograph by Barenboim and Elkin [13], and another very detailed overview on more recent results on coloring is contained in [46]. Detailed overviews on randomized algorithms are contained in [23, 37]. Due to the sheer amount of published papers on distributed coloring, we focus on selected results that have not been discussed in detail in [13, 23, 37, 46, 51], are most related to the current work, or indicate in which direction future research should continue or should probably not continue.

The objective in the edge coloring problem is to assign a color to each edge of a graph such that adjacent edges obtain different colors. Even though Vizing's Theorem [61] states that any graph with maximum degree $\Delta$ can be colored with $\Delta+1$ colors and several randomized and deterministic algorithms get close to this bound, e.g., [18, 36, 42, 59] in LOCAL and [39] in CONGEST, the classic objective is to use $2\Delta - 1$ colors. The reason is that $(2\Delta - 1)$-edge coloring is a (Maxdegree + 1)-vertex coloring of the line graph. The problem has a $f(\Delta) + \log^* n$ LOCAL algorithm with $f(\Delta) = \log^{O(1)} \Delta$ [5]. Obtaining such a runtime for computing a $(\Delta + 1)$-vertex coloring would be a major breakthrough. In the CONGEST model, the state of the art for $(2\Delta-1)$-edge coloring is an algorithm using $O(\Delta + \log^* n)$-round deterministic algorithm [14]. An edge coloring with $O(\Delta)$ colors can be computed in the CONGEST model with $f(\Delta) = \log^{O(1)} \Delta$ [5].

Besides optimizing the dependency on $\Delta$ after spending only $\log^* n$ rounds on Linial's coloring algorithm, another big branch of research has tried to settle the complexity of the problem as a function of $n$. For almost 30 years the best deterministic algorithm in this regime was a $2^{O(\sqrt{\log n})} \gg \log^{O(1)} n$-round algorithm [1, 54], which has been improved to $O(\log^5 n)$ rounds [32, 56] in the LOCAL model and in the CONGEST model [8, 32] (slightly slower). The crucial building block of all of these results is a decomposition of the graph into $O(\log n)$ classes $C_1, \ldots, C_{O(\log n)}$ of small-diameter clusters. To solve the $(\Delta + 1)$-coloring problem one iterates through the $O(\log n)$ classes and solves each cluster $C \in C_i$ in parallel in time that is (at least) linear in the cluster diameter. Even existentially, such decompositions require that the cluster diameter is at least $\Omega(\log n)$, and as a result these methods can probably not yield runtimes that are $o(\log^2 n)$. Thus, the fastest algorithm [35] that needs $O(\log n \log^2 \Delta)$ rounds uses a different approach: similar to [8], it derandomizes a simple randomized one-round algorithm. Output colors are represented as bit strings of length $O(\log \Delta)$ and in one round of the algorithm each node flips a (suitably weighted) coin to determine the next bit in the string. In expectation, after all $O(\log \Delta)$ bits are fixed, a constant fraction of the vertices can be colored. Bamberger et al. derandomize this algorithm for each cluster of a given network decomposition [8]. In contrast, instead of computing a network decomposition and derandomizing within a cluster, Ghaffari and Kuhn derandomize the algorithm *globally* with the help of a special kind of a defective coloring [35]. Their derandomization step takes $O(\log \Delta)$ rounds for each of the $O(\log \Delta)$ bits and the $O(\log n)$ factor follows as only a constant fraction of the vertices gets colored in each phase, yielding a total runtime of $O(\log^2 \Delta \log n)$ rounds. Similar methods, also yielding $\log n \cdot \log^{O(1)} \Delta$ runtimes, have been successful for edge-coloring [33, 42] and computing maximal matchings [27]. Very recently these techniques have also been extended to provide new algorithms for computing maximal independent sets [26] and even network decompositions [31]. If one allows $O(\Delta^{1+\varepsilon})$ colors for a constant $\varepsilon > 0$, a $O(\log \Delta \log n)$-round algorithm has been known for more than a decade [12].

As shown in [20, 22], a logarithmic dependency on $n$ ($\log_\Delta \log n$-dependency for randomized algorithms) is unavoidable if one colors with fewer than $\Delta + 1$ colors; that is, $\Delta$-coloring requires at least $\Omega(\log_\Delta n)$ rounds. Similar bounds hold for the edge coloring problem for coloring with fewer than $(2\Delta - 1)$-colors [7] and for coloring trees and bounded arboricity graphs with significantly fewer than $\Delta$ colors [11, 50]. Very recently it was shown that the $\Delta$ term in the base of the randomized lower bound is necessary as the problem can be solved in $O(\log^* n)$ rounds with a randomized CONGEST algorithm if $\Delta$ is lower bounded by $C \cdot \log^{21} n$ for a sufficiently large constant $C$ [28].

Little is known on lower bounds for $C$-coloring when $C \geq \Delta + 1$ (in contrast to other symmetry breaking problems, e.g., maximal matching, MIS, or ruling sets [3, 6, 48]). Linial's $\Omega(\log^* n)$ deterministic lower bound has recently been re-proven in a topological framework [30]. A $\Omega(\Delta^{1/3})$ lower bound for $O(\Delta)$-coloring holds in a weak variant of the LOCAL model [44]. Several papers analyzed special cases of coloring algorithms that can only spend a single communication round [44, 49, 60]. Just like the lower bounds in this article, none of these results gives anything non-trivial for two rounds. Also, the *speedup* technique (e.g., [2, 3, 6, 19–21]), which proved very successful, e.g., for MIS and ruling set lower bounds, is not yet helpful for graph coloring. To make full use of the technique, one uses a computer program [53] to automatically transfer a problem $P_0$, e.g., the $(\Delta+1)$-coloring problem, into a problem $P_1$ that requires exactly one communication round less in the LOCAL model. Then, one iterates the process to obtain problems $P_0, P_1, \ldots, P_t$, and if $P_t$ cannot be solved with a 0-round algorithm, problem $P_0$ has a lower bound of $t$ rounds. Usually, the program is applied for small values of $\Delta$, and in a second step, the gained insights are transferred into a formal proof for general $\Delta$. For graph coloring the description of the problems grows so quickly with $t$ that even for small values of $\Delta$ one cannot even compute $P_1$ with current computers.

There has also been a lot of progress in randomized coloring algorithm, e.g, [17, 23, 37, 43], where the state of the art for $(\Delta + 1)$-vertex coloring is a $O(\log^3 \log n)$ algorithm in the LOCAL model [23, 35] and $O(\log^6 \log n)$ in the CONGEST model [37]. Remarkably, there is a randomized $O(\log^* \Delta)$-round algorithm to compute a coloring with $\Delta + \log^\gamma n$ colors for a large enough constant $\gamma > 0$ [23]. Prior to this, Schneider and Wattenhofer [58] showed that one can compute a $O(\Delta + \log^{1.1} n)$-coloring in $O(\log^* \Delta)$ rounds of LOCAL. Very recently, Halldórsson and Nolin showed that these results can be extended to the CONGEST model [41]. All of these latter randomized algorithms make use of the concept of trying several colors in one round, similar to our algorithm for $k > 1$. In 2021, Halldórsson et al. [38] showed that $(deg + 1)$-list coloring can be solved in $\log^{O(1)} \log n$ rounds in the randomized LOCAL model. In 2022, Halldórsson et al. obtained similar result in the CONGEST model [40]. These randomized algorithms run in $O(\log^* n)$ time if (additionally) it is guaranteed that each list is of size at least $\log^{2+\Omega(1)} n$) or $\Omega(\log^7 n)$, respectively; otherwise they run in $O(\log^3 n)$ LOCAL rounds. While Naor extended Linial's $\Omega(\log^* n)$ lower bound to randomized algorithms [52] (on rings with $\Delta = 2$), it is not known whether the bounds for $O(\Delta)$-coloring for large $\Delta$ are tight. When $\Delta \geq \log^{\Omega(1)} n$ holds, our current knowledge does not rule out $O(1)$-round algorithms for $O(\Delta)$-coloring. This question is even more of interest as in this setting a $\Delta^{O(1)}$-coloring can be computed in one round from unique IDs from a space of size $n^{O(1)}$.

Additionally, we want to point out that, independently from this work, Barenboim et al. have extended their clever algorithm [14] to compute $O(k\Delta)$-colorings in $O(\Delta/k + \log^* n)$ rounds. These results appear in the journal version [15].

### 1.3 Roadmap

In Section 2 we present the $O(\Delta/k)$-round $O(k\Delta)$-coloring algorithm and its implications and modifications to compute defective and outdegree colorings. In Section 3 we explain how this simplifies the state of the art for $(\Delta + 1)$-vertex coloring, our $O(k\Delta)$-coloring in $O(\sqrt{\Delta/k})$-rounds, and our

results on computing ruling sets. In Section 4 we analyze one-round color reduction algorithms. In Section 5 we conclude and explain why reducing an input $(1+\varepsilon)\Delta$-coloring to a $(\Delta+1)$-coloring might be the hardest part of the $(\Delta+1)$-coloring problem.

## 2 MAIN ALGORITHM: COLORING MADE EASY

The objective of this section is to prove Theorem 1.1, where the emphasis is on the fact that the algorithm is extremely simple if one ignores the precise choice of parameters. Before we prove Theorem 1.1, we prove Corollary 1.2 with useful settings of the respective parameters in Theorem 1.1; while Theorem 1.1 is the technical result, the corollary is supposed to be the interface to the outer world. To formally state the corollary, we begin with two definitions.

Proof of Corollary 1.2. In each of the results we apply Theorem 1.1 with different parameters. **Proof of (1).** Choose $d = 0$, which implies $X = 16\Delta$. With $k = X$ we obtain a proper $C = X \cdot k = 256\Delta^2$-coloring in one ($R = \frac{X}{k} = 1$) round. **Proof of (2).** Choose $d = 0$, which implies $X = 16\Delta$. Then we obtain a $16\Delta \cdot k$-coloring in $R = \frac{16\Delta}{k}$ rounds. **Proof of (3).** Choose all parameters as in 2., but set $k = \lceil \frac{\Delta}{16} \rceil$, which implies $\Delta^2$ colors in $R = \frac{16\Delta}{k} = O(1)$ rounds.

In (4)–(6). the condition $d = \beta = \Delta^\varepsilon$ implies $X = O\left(\frac{\Delta \cdot \log \frac{\Delta}{\beta+1} \Delta^4}{\beta+1}\right) = O\left(\frac{\Delta}{\beta}\right)$.

**Proof of (4).** Let $k = 1$, which implies the claimed number of colors ($X \cdot k = O(\frac{\Delta}{\beta+1})$ and the claimed round complexity ($R = O(\Delta/\beta)$. The coloring is a $\beta$-outdegree coloring due to part (1) of Theorem 1.1. **Proof of (5).** With $k = X$ the runtime is $R = \frac{X}{k} = 1$ rounds and we obtain $C = X \cdot k = O((\frac{\Delta}{d})^2))$ colors. Theorem 1.1 says that the coloring has defect at most $d$ as there is only one subgraph ($R = 1$). **Proof of (6).** Choose $k = 1$. Let $P_1, \ldots, P_R$ be the partition of part (2). If vertices consider their color and the index of their part of the partition as a color tuple, i.e., if a vertex $v \in P_j$ with color $\phi(v)$ colors itself with color $(\phi(v), j)$, we obtain a $d$-defective coloring with $O((\frac{\Delta}{d})^2)$ colors in $O(\frac{\Delta}{d})$ rounds. □

Corollary 1.2 shows that one algorithm is sufficient for many of the essential steps of several previous important papers, and it further allows to smoothly scale between these results. Additionally, the algorithm for computing $\beta$-outdegree colorings is simpler and more direct than previous algorithms. The algorithm in [14] first needs to compute a certain defective coloring and only then can proceed to compute a low outdegree coloring. The slower algorithm in [9] uses a more involved recursive approach. The algorithm(s) in [11, 12] are more involved and require $\Omega(\log n)$ rounds.

We continue with explaining the algebraic basics to construct the sequences for the algorithm for Theorem 1.1 (Algorithm 1). Given a prime $q$, let $\mathbb{F}_q$ denote the field of size $q$ over the elements $[q]$ and let

$$P_q^f = \{p : \mathbb{F}_q \to \mathbb{F}_q \mid p \text{ is polynomial of degree} \leq f\}$$

be the set of all polynomials over $\mathbb{F}_q$ of degree at most $f$. Recall that $Z = \frac{\Delta}{d+1}$. To run the algorithm on a graph with maximum degree $\Delta$, an input $m$-coloring and a *defect parameter* $d$ fix $f = \lceil \log_Z m \rceil$ and a prime $q$ with

$$2f \cdot Z < q < 4f \cdot Z, \tag{1}$$

which exists due to Bertrand's postulate. Then we can locally and without communication assign each input color $i \in [m]$ a distinct polynomial $p_i \in P_q^f$ as $m \leq |P_q^f| = q^{f+1}$ and since all vertices know $m$ and $f$. For example, we can represent each element $p(x) = \sum_{i=0}^{f} a_i x^i$ of $P_q^f$ as a tuple

$(a_0, \ldots, a_f)$, order the tuples lexicographically, and assign the polynomial corresponding to the $i$th tuple with input color $i$. Given these polynomials, uncolored nodes try the colors in their sequences in batches of size $k$ ($k$ is an integer parameter that can be freely chosen). To *try a color* it is sent to its neighbors and nodes evaluate for each of their tried colors how many neighbors try the same color or are already colored with it. Then a node gets permanently colored with a color $c$ or *adopts* color $c$ if it causes conflicts with at most $d$ neighbors. To this end call a color $c$ tried by some node $v$ *$d$-proper* for node $v$ in some iteration if at most $d$ neighbors try color $c$ in the same iteration or are already permanently colored with color $c$. Note that the property of a color of being $d$-proper for a node $v$ in some iteration depends on the batches of tried colors of $v$ and all of its neighbors, as well as the colors that are already adapted by $v$ already permanently colored neighbors. More formally, fix some iteration and let $D \subseteq$ be the nodes that have already determined their final output color in one of the previous iterations; for $v \in D$ denote that permanent output color by $\psi(v)$. For $v \in V \backslash D$ let $B(v)$ be the batch of colors tried by node $v$ in that iteration. Then, a color $c \in B(v)$ is *$d$-proper* for node $v$ in that iteration if $|\{u \in N(v) \cap V \backslash D \mid c \in B(u)\}| + |\{u \in N(v) \cap D \mid c = \psi(u)\}| \leq d$. The details are given by Algorithm 1 and the following paragraph.

**ALGORITHM 1:** For vertex $v$ with color $i$. Parameters $d, k, m, \Delta$. (LOCAL model)

**Locally compute:**
polynomial $p_i : \mathbb{F}_q \to \mathbb{F}_q$ with $q$ chosen by Equation (1)
sequence $s_i$: $(x \mod k, p_i(x) \mod q)$, $x = 0, \ldots, q-1$
Split $s_i$ into $\lceil q/k \rceil$ consecutive batches $B_1(v), B_2(v), \ldots$, each of size $k$ (except for the last)
**For** $j = 1, \ldots, \lceil q/k \rceil$
**Send** the colors in batch $B_j(v)$ (in a single round) to neighbors
**if** $\exists$ *($d$-proper $c \in B_j(v)$)* **then** adopt $c$, join $P_j$, and **return**;

Observation 2.1. *For each $v \in V$, $1 \leq j \leq \lceil q/k \rceil$, the colors tried in one batch $B_j(v)$ are distinct.*

Proof. The colors in a batch are of the form $(x \mod k, y)$, where $x$ ranges through (at most) $k$ consecutive integers. □

Sending the colors of one batch takes one round of communication in the LOCAL model. We reason at the end of the proof of Theorem 1.1 that processing one batch also can be done in a single round of CONGEST. If $k$ does not divide $q$, a node that is uncolored before the last iteration tries less than $k$ tuples in the last iteration, i.e., $|B_{\lceil q/k \rceil}| < k$. In fact, it will try $q - k\lfloor q/k \rfloor$ tuples. When a vertex picks a tuple as its permanent color $c$, it orients all edges toward neighbors that have previously chosen $c$ as a permanent color. If two neighbors both pick the same permanent color $c$ in the same iteration, the edge between them is oriented arbitrarily (e.g., using the input coloring for symmetry breaking from smaller input color to larger input color). A node joins the subgraph $P_j$, where $j$ is the index of the iteration in which it decides for a permanent color.

Note that vertices with the same input color compute the same sequence and that all steps that are related to orientations and subgraphs are obsolete when $d = 0$. We will show that the algorithm is well defined, i.e., that every vertex is colored before it reaches the end of its sequence. To prove the result, we need the following well-known algebraic result on the number of intersections of two degree-bounded polynomials over finite fields.

Lemma 2.2. *Let $q$ be a prime, $f \in \mathbb{N}_0$, and let $p_1, p_2 \in P_q^f$ be distinct polynomials of degree $f_1, f_2$, respectively. Then there are at most* $\max\{f_1, f_2\}$ *points in which $p_1$ and $p_2$ intersect, i.e.,* $|\{x \in \mathbb{F}_q \mid p_1(x) = p_2(x)\}| \leq \max\{f_1, f_2\}$ .

Proof of Theorem 1.1. Recall that the prime number $q$ is the size of the field $\mathbb{F}_q$ from which the coefficients of the polynomials are taken. Due to the choice of $q$ and $f$ (see Equation 1), the set $P_q^f$ contains at least one distinct polynomial for each input color $i \in [m]$. Recall the choice of parameters: $Z = \frac{\Delta}{d+1}$, $X = 4 \cdot Z \cdot \lceil \log_Z m \rceil$, and $R = \Delta/k$. Let $C = X \cdot k$ and note that $X \geq q$. We first prove all statements under the assumption that all vertices are colored after the $R$ iterations of the loop; afterward we show that this is indeed the case.

**Bounding # colors:** Each color is of the form $(x \mod k, p(x) \mod q)$ for some polynomial $p$ that is evaluated over $\mathbb{F}_q$. Thus, the number of colors $C$ can be upper bounded by $k \cdot q \leq k \cdot X$.

**Proof of (1):** When a vertex $v$ is colored with a color $\phi(v)$ in iteration $j$, there are at most $d$ other vertices that try to get the same color in this round or are already colored with this color. Since only edges to these vertices are oriented outward from $v$, the outdegree of $v$ is bounded by $d$. Further, each edge between vertices with the same permanent color is oriented: either they picked the color in the same iteration and the edge is oriented from the node with the smaller input color to the vertex with larger input color, or the edge is oriented outward from the vertex that got permanently colored in a later iteration.

**Proof of (2):** A vertex $v$ joins $P_j$ if it is colored in iteration $j$ of the loop. As a vertex $v$ only gets colored with a color $\phi$ in iteration $j$ if there are at most $d$ neighbors of $v$ that want to get colored with color $\phi$ in iteration $j$, the maximum degree of the graph induced by all vertices in $P_j$ with color $\phi$ is at most $d$.

**All vertices are colored after the $R$ iterations of the loop:** A node is not colored in one iteration only if for all of the $k$ tuples, i.e., colors of the form $(x, p_i(x)) \in [k] \times [q]$, that it tries in that iteration there are strictly more than $d$ neighbors that try the same tuple in the current iteration or are already colored with the tuple. Before we proceed with the proof that all vertices are colored at the end, we bound the number of conflicts that a node experiences during the execution of the algorithm. We consider two types of conflict.

*Bounding the number of conflicts with an active node by $f$:* Let us bound the number of times in which two neighbors $u$, $v$ with polynomials $p_u$ and $p_v$, respectively, try the same tuple in some iteration $j \in [\lceil q/k \rceil]$ (conditioned on both nodes not being permanently colored yet). In iteration $j$, the nodes simultaneously try all of the following tuples (where we omit the $j \cdot k \mod k = 0$ term in the first coordinate):

$$u \text{ tries: } \Big(l, p_u(j \cdot k + l)\Big) \text{ with } l \in [k] \text{ and} \tag{2}$$

$$v \text{ tries: } \Big(l, p_v(j \cdot k + l)\Big) \text{ with } l \in [k]. \tag{3}$$

Two tuples tried by $u$ and $v$ in iteration $j$ can only cause a conflict, i.e, be the same, if they are the same in both coordinates. As all $k$ tuples that are simultaneously tried by a node differ in the first coordinate, any conflict in iteration $j$ between $u$ and $v$ implies that $p_u(j \cdot k + l) = p_v(j \cdot k + l)$ holds for some $l \in [k]$. Since $p_u$ and $p_v$ are polynomials of degree at most $f$, Lemma 2.2 implies that there are at most at most $f$ combinations of $j$ and $l$ for which this holds.

*Bounding the number of conflicts with an inactive node by $f$:* Consider a neighbor $u$ that chose some permanent color $(x_u, y_u) \in [k] \times [q]$. For $v$ to try this tuple in iteration $j \in [q/k]$ we need

$$\Big((j \cdot k + l) \mod k, p_v(j \cdot k + l) \mod q\Big) = \Big(x_u, y_u\Big)$$

for some $l \in [k]$. This can only be the case if $p_v(j \cdot k + l)$ equals the fixed number $y_u$, which is the case for at most $f$ different choices of $j$ and $l$ due to Lemma 2.2 ($y_u$ is a polynomial of degree 0).

Thus, for fixed $u$ and $v$, there are at most $f$ tuples causing a *conflict* while $u$ and $v$ are active, and at most $f$ tuples causing a *conflict* after (at least) one of the nodes has chosen a permanent

color. A node $v$ cannot get permanently colored with a tuple $(l, p_v(j \cdot k + l))$ if there are strictly more than $d$ conflicts for the tuple, i.e., strictly more than $d$ neighbors try the same tuple in the same iteration or have already permanently adapted the color. In this case we call the tuple *blocked*. As each of the at most $\Delta$ neighbors contributes at most $2f$ such conflicts, there can be at most $z = 2f \cdot Z$ blocked tuples. As the length $q$ of the sequence (of tried tuples) is strictly larger than $z$, there is at least one tuple that is not blocked and each node is colored at the end of the algorithm.

CONGEST **implementation:** During the execution of the algorithm all nodes have knowledge of $m$, $q$, $f$, $d$, and $k$ and all nodes can construct the set of polynomials $P_q^f$ locally according to the same lexicographic order. Thus, for an uncolored node to send $k$ trials in iteration $j$, it is sufficient to send its input color (together with $k$ and $j$, which are global knowledge). A node that gets colored can inform its neighbors about the choice in one round. Hence, a node never needs to send more than a single color per round and the message size is upper bounded by $O(\log \Delta) = O(\log n)$ bits and can be executed in the CONGEST model. □

We remark that it becomes significantly easier to read the algorithm if $m = \Delta^{O(1)}$ and if $\Delta/d = \Delta^{\Omega(1)}$, as this implies $f = O(1)$, which simplifies many of the parameters. However, we chose to present the result in a more general form.

*Remark 2.3.* With a tighter analysis for special cases one can reduce the constants in Corollary 1.2; e.g., in the case of $k = X$, the size of the field $\mathbb{F}_q$ can be chosen smaller. Due to such a tighter analysis and by assuming $m = \Delta^3$, the leading constant in the $O(\Delta^2)$-coloring by Linial is some $1 < \alpha < 10$ [50]. In contrast, the lower bound for the one-round algorithms from Section 4 only provides impossibilities below $\Delta^2/2 + \Theta(\Delta)$ colors, that is, for a constant $\alpha < 1$. Thus, there is a large regime for $\alpha$ where we have neither one-round upper bounds nor lower bounds. As even optimized constants in Theorem 1.1 and Corollary 1.2 leave a gap for the regime of $\alpha$ where lower bounds are known, we focus on having simple proofs that cover all cases of the theorem, instead of optimizing these constants.

The condition $d = \beta = \Delta^\epsilon$ in Corollary 1.2: One would wish to use a variant of Corollary 1.2 to compute a $\Delta/2$-defective $O(1)$-coloring in one round, given a $O(\Delta^2)$-coloring. Note that this setting would require the finite field over which we operate—the field size essentially determines the number of colors—to contain only $q = O(1)$ elements, and to obtain a distinct polynomial for each input color we would have to choose polynomials of degree $f = O(\log \Delta) \gg q$. This immediately violates Equation 1. Also, in that case, the proof of Theorem 1.1 breaks as we might have $\Omega(f \cdot \frac{\Delta}{d}) = \Omega(f) = \omega(1)$ blocked tuples while only having $q = O(1)$ tuples in the sequence. While slightly weaker requirements on $d$ are possible without breaking the proof, our requirement ensures that we do not run into these issues. See [16, 45] for the parameter-heavy details on how to iterate the result of Theorem 1.1 for $O(\log^* \Delta)$ iterations to obtain a $d$-defective $O((\frac{\Delta}{d})^2)$-coloring with no condition on $d$ (essentially Corollary 1.2 (5) can also take a defective coloring as input coloring, and then defects add up).

We point out that the sequences required for Theorem 1.1 need not be constructed via polynomials. The proof only requires that the elements of the sequence are from a small enough domain, sequences are long enough, there is one sequence for each input color, and any two sequences intersect in few positions. In [51, arxiv version] such sequences are constructed greedily. Here, we chose to use a construction based on polynomials as the dependency on the input $m$-coloring is better; in particular, when $m = \Delta^{O(1)}$, it implies that $f = O(1)$, instead of $f = O(\log \Delta)$ for the greedy-based construction.

## 3 APPLICATIONS

### 3.1 Simplifying $(\Delta+1)$-coloring Algorithms

Corollary 1.2 provides a simpler algorithm to compute so-called $\beta$-outdegree colorings with $O(\Delta/\beta)$ colors in $O(\Delta/\beta)$ rounds matching the state of the art of [14]. Recall that an $\beta$-outdegree $c$-coloring is an improper coloring with $c$ colors in which each monochromatic edge is equipped with an orientation such that the outdegree of each vertex is at most $\beta$. These colorings are used in all known $(\Delta+1)$-coloring algorithms whose round complexity is sublinear in $\Delta$ [9, 14, 29, 51], and hence our simplifications carry over to these algorithms. The core message of our work is that all results in Corollary 1.2 can be obtained through modifications of Linial's algorithm. Next, we sketch the algorithm in [51] to explain that the same is true for their result. The algorithms of [9, 14, 29, 51] use the following high-level scheme: First compute a $\beta$-outdegree $z$-coloring for a suitable choice of $\beta$ such that $z = O(\Delta/\beta) = o(\Delta)$. Its color classes yield a partition $V_1, \ldots, V_z$ of the vertex set. In a second step, the partition is used as a schedule. In order to compute the nodes' final output color, we iterate through the schedule. For the purpose of this exposition we may assume that all nodes of $V_i$ are colored with their final output color after processing them (formally, this is only true for certain nodes of $V_i$ and an additional recursion is required to color all nodes of the graph). When processing $V_i$, we ensure that a node does not get colored with a color of any of its already colored neighbors in $V_1 \cup \ldots \cup V_{i-1}$; that is, when a node is processed it does not have all of the $\Delta+1$ output colors available, but instead its *list of available colors* does not include the colors of its already colored neighbors. Thus, the resulting problem that we need to solve on $G[V_i]$ is a so-called *list coloring* problem. The additional lever, when coloring $V_i$, is that $G[V_i]$ is equipped with an orientation with a small outdegree. The core result of the paper with the subtitle "Linial for Lists" [51] is a generalization of Linial's one-round color reduction algorithm to the list coloring problem that works in two rounds in graphs with small outdegrees. Hence, the crucial coloring step and the algorithm to compute the necessary schedule, i.e., the $\beta$-outdegree coloring, are generalizations of Linial's algorithm.

### 3.2 Improved $O(\Delta^{1+\varepsilon})$-coloring Algorithms

If one aims for $O(\Delta)$ colors (instead of $(\Delta+1)$ colors), the scheme that we explained in Section 3.1 works for a suitable choice of $\beta = \Theta(\sqrt{\Delta})$, yielding a runtime of $O(\Delta/\beta) = O(\sqrt{\Delta})$ rounds.

Theorem 3.1 ([9, 14]). *There is a deterministic* CONGEST *algorithm that computes a $O(\Delta)$-coloring in $O(\sqrt{\Delta} + \log^* n)$ rounds on any graph with maximum degree $\Delta$.*

We use this algorithm and our defective coloring algorithm from Corollary 1.2 to improve the tradeoff between the number of colors and the runtime from $O(k\Delta)$ vs. $O(\Delta/k)$ (Corollary 1.2) to $O(k\Delta)$ vs. $O(\sqrt{\Delta/k})$.

Theorem 1.3. *For any constant $\varepsilon > 0$, there is a deterministic* CONGEST *algorithm that computes a $O(\Delta^{1+\varepsilon})$-coloring in $O(\Delta^{1/2-\varepsilon/2}) + \log^* n$ rounds on any graph with maximum degree $\Delta$.*

Proof. First, compute an $O(\Delta^2)$-coloring in $\log^* n + O(1)$ rounds using Linial's algorithm [50]. Set $d = \Delta^{1-\varepsilon}$, and then use Corollary 1.2 (part 6) to compute a $d$-defective coloring $\psi$ with $O((\frac{\Delta}{d})^2)$ colors in $O(\Delta/d) = O(\Delta^{\varepsilon})$ rounds. Then, on each color class in parallel compute a $O(d)$-coloring in $O(\sqrt{d}) = \Delta^{1/2-\varepsilon/2}$ rounds (the internals of [9, 14] show that the $\log^* n$ term from the application of Theorem 3.1 actually is $\log^* C$ when we are given an input $C$-coloring; as in our case we are given an input coloring with $C = (\Delta^2)$ colors, the $\log^* C$ term is subsumed in the remaining runtime) via Theorem 3.1 using a distinct color space for each color class of $\psi$; that is, each node $v$ gets a color $\phi(v)$ from this second step and the final output color of node $v$ is set to be the tuple $(\psi(v), \phi(v))$. In total, we use $O((\frac{\Delta}{d})^2 \cdot d) = O(\frac{\Delta^2}{d}) = O(\Delta^{1+\varepsilon})$ colors. □

While the above way is a simple way to prove Theorem 1.3 when using Theorem 3.1 as a black-box, an alternative algorithm can be obtained by using the $\beta$-outdegree coloring result from Corollary 1.2 (with $\beta = \Delta^{1/2+\varepsilon/2}$) and carefully choosing the remaining parameters in the framework of [9]. The corresponding parameters do not appear in the scheme in Section 3.1.

### 3.3 Ruling Sets

A $(2,r)$-ruling set of a graph $G = (V,E)$ is a subset $S \subseteq V$ of the vertices that is an independent set and satisfies that for any vertex $v$ in $V$ there is a vertex $s \in S$ in hop distance at most $r$ [1]. The following result uses colorings to compute a ruling set.

Lemma 3.2 ([47, arxiv version]). *For any $B \geq 2$ there exists a deterministic distributed* CONGEST *algorithm that, given a $C$-coloring, computes a $(2, \lceil \log_B C \rceil)$-ruling set in $O(B \log_B C)$ rounds.*

We use Lemma 3.2 and Theorem 1.3 to compute $(2,r)$-ruling sets by adjusting the number of colors such that the runtime of computing the coloring and using it via Lemma 3.2 are balanced.

Theorem 1.5. *For any constant integer $r \geq 2$, there is a deterministic* CONGEST *algorithm that computes a $(2,r)$-ruling set in $O(\Delta^{\frac{2}{r+2}}) + \log^* n$ rounds on any graph with maximum degree $\Delta$.*

Proof. Set $\varepsilon = \frac{r-2}{r+2}$ and use Theorem 1.3 to compute an $O(\Delta^{1+\varepsilon})$-coloring in $O(\Delta^{1/2-\varepsilon/2}) + \log^* n$ rounds. Let $C = O(\Delta^{1+\varepsilon}) = O(\Delta^{\frac{2r}{r+2}})$ be the number of colors of this coloring. Now, set $B$ such that $\lceil \log_B C \rceil = r$ and apply Lemma 3.2 to compute a $(2,r)$-ruling set in $O(B \log_B C) = O(B \cdot r) = O(C^{1/r})$ rounds. Ignoring the $\log^* n$ term, the total runtime is upper bounded by

$$O(\Delta^{\frac{1}{2}-\frac{r-2}{2r+4}} + C^{1/r}) = O(\Delta^{\frac{1}{2}-\frac{r-2}{2r+4}} + \Delta^{\frac{2}{r+2}}) = O(\Delta^{\frac{2}{r+2}}) \,. \qquad \square$$

Interestingly, for $r = 2$ the state-of-the-art runtimes for the $(2,r)$-ruling set is the same as the complexity for computing an $O(\Delta)$ coloring. Note that the runtime bound of Theorem 1.5 cannot be achieved for $r = 1$. In that case, the ruling sets are better known under the name *maximal independent sets* for which a $\Omega(\Delta)$-round lower bound is known if the runtime's $n$-dependency is limited to $O(\log^* n)$ [3].

An $(\alpha, r)$-*ruling set* is a subset $S \subseteq V$ that is an independent set in the power graph $G^{\alpha-1}$ that satisfies that each vertex $v \in V$ has a vertex in $S$ in distance at most $r$. In the LOCAL model the results of Theorem 1.5 can be extended to $(\alpha, r)$-ruling sets as one can simulate any algorithm on $G^{\alpha-1}$ in the original network graph. For details on these black-box extensions see, e.g., [17, 47].

## 4 ONE-ROUND COLOR REDUCTION

The objective of this section is to show the following theorem:

Theorem 1.6. *For any integer $\Delta \geq 1$ and $\Delta+1 \leq m \leq \frac{\Delta^2}{4} + \frac{3\Delta}{2} + \frac{9}{4}$ let $1 \leq k \leq \frac{\Delta}{2} + \frac{3}{2}$ be the largest integer such that $m \geq k(\Delta - k + 3)$. Then, there is a one-round* CONGEST *algorithm that on any input $m$-colored graph with maximum degree $\Delta$ computes an $(m-k)$-coloring. Additionally, if also $k \leq \Delta - 2$ holds, then there is no one-round* LOCAL *algorithm that outputs a proper $(m-k-1)$-coloring on every input $m$-colored graph with maximum degree $\Delta$.*

In addition to the values for $k$ that are stated in Section 1 ($k = 1, 2, 3, 4$), note that one requires $5\Delta - 10$ input colors to reduce five colors, and $6\Delta - 18$ input colors to reduce six colors. The proof is split into two lemmas. In Lemma 4.1, we provide a one-round color reduction algorithm, and in Lemma 4.3 we show that the algorithm is tight up to each single color. For a function $f : V \to [m]$ and a set $S \subseteq V$ we denote $f(S) = \{f(v) \mid v \in S\}$. For a node $v \in V$ of a given graph $G = (V,E)$ we denote the set of its neighbors by $N(v)$.

The following result reduces more colors than the one-round algorithms in [49, 60].

Lemma 4.1 (Color Reduction). *For an integer* $1 \leq k \leq \frac{\Delta}{2} + \frac{3}{2}$ *there is a one-round color reduction procedure from* $m \geq k(\Delta - k + 3)$ *to* $k(\Delta - k + 2)$ *(reduces* $k$ *colors).*

The intuitive idea of the algorithm is that only vertices in the $k$ largest color classes change their color. Each of these *recoloring colors* has its own small hardcoded output color regime from which it can pick a *free color*. However, the size of the regime is smaller than $\Delta + 1$ and it might be that all of its colors are blocked by neighbors that do not recolor themselves. But this implies that there are some recoloring colors that do not appear in the node's neighborhood and it can steal colors from those recoloring colors' regimes to gain the desired freedom (one stolen color per regime).

Recall that for an integer $x$, we use the notation $[x]$ to refer to the set $\{0, \ldots, x-1\}$.

Proof of Lemma 4.1. We may assume that $m = k(\Delta - k + 3)$ is the number of input colors as for $m' \geq m$ input colors, one can leave $m' - m$ colors unchanged and apply the algorithm to the remaining $m$ colors. Let $\ell = k(\Delta - k + 2) \geq \Delta + 1$ be the number of (desired) output colors, $\phi : V \to [m]$ the input coloring, and $\psi : V \to [\ell]$ the to-be-computed output coloring. Observe that $m = \ell + k$.

Fix $k$ disjoint color regimes of output colors $R_0, \ldots R_{k-1}$, each of size $\Delta - k + 2$, as follows: for $i \in [k]$ and $j \in [\Delta - k + 2]$, let $r_i(j) = i \cdot (\Delta - k + 2) + j \in [\ell]$ and $R_i = \{r_i(j) \mid j \in [\Delta - k + 2]\} \subseteq [\ell]$.

We refer to $R_i$ as the *ith regime*. The regimes are disjoint and each regime $R_i$ is of size $|R_i| = \Delta - k + 2 \geq k - 1$ because $k \leq \frac{\Delta}{2} + \frac{3}{2}$ holds. Additionally, for each of the $k$ regimes, let $f_i : [m] \backslash ([\ell] \cup \{\ell + i\}) \to R_i$ be an abitrary injective function into the regime. Note that $f_i$ exists as its domain is of size $m - \ell - 1 = k - 1 \leq |R_i|$. Nodes execute the following algorithm that takes one round as a node needs to learn its neighbors' input colors.

**ALGORITHM 2:** Executed at each node $v$, output coloring $\psi$, input coloring $\phi$

**Send** input color $\phi(v)$ to neighbors; **Receive** set of neighbors' input colors $\phi(N(v))$;
**Case** $\phi(v) < \ell$**:** $\psi(v) := \phi(v)$; **exit**;
**Case** $\max_{u \in N(v)}\{\phi(u)\} < \ell$**:** $\psi(v) := \min([\Delta + 1] \backslash \phi(N(v)))$; **exit**;
**Case (else):** $F(v) := R_{\phi(v)-\ell} \cup \{f_j(\phi(v)) \mid 0 \leq j < k, \ell + j \notin \phi(N(v))\}$
$\psi(v) := \min(F(v) \backslash \phi(N(v)))$

Let $v$ be a node. If $v$ executes the first case, it does not change its color and neighbors ensure to not output the same color as $v$. If $v$ executes the second case, all of its neighbors execute the first case and do not change their colors; $v$ selects a color not conflicting with any of these. Hence, for the rest of the proof assume that $v$ executes the third case. By the next lemma we have $F(v) \cap F(w) = \emptyset$ for any neighbor $w$ of $v$, regardless of which case $w$ executes.

Claim 4.2. *For any two neighbors* $v$ *and* $w$ *we have* $F(v) \cap F(w) = \emptyset$.

Proof. Since $\phi(w) \neq \phi(v)$, we obtain that the regimes $R_{\phi(v)-\ell}$ and $R_{\phi(w)-\ell}$ are disjoint. By the definition of $F(v)$ and $F(w)$ (more particular by the co-domains of the $f_j$s used in the definition of $F(v)$ and $F(w)$, respectively), we additionally obtain that $F(v)$ does not intersect $R_{\phi(w)-\ell}$ and $F(w)$ does not intersect $R_{\phi(v)-\ell}$. Hence, if $F(v)$ and $F(w)$ intersect, the intersection must lie in some regime $R_j$, where $j \neq \phi(v)$ and $j \neq \phi(w)$. However, $F(v)$ and $F(w)$ each contain at most one color in each such $R_j$. These colors are $f_j(\phi(v))$ and $f_j(\phi(w))$, respectively. We obtain $f_j(\phi(v)) \neq f_j(\phi(w))$ because $f_j$ is injective and $\phi(v) \neq \phi(w)$ holds. □

As $v$ executes the third case, we have $\phi(v) \geq \ell$, and no neighbor of $v$ can execute the second case. Hence, all neighbors of $v$ either stick to their color that is already $< \ell$ or also execute the third case. Let $d(v)$ be the number of neighbors of $v$ that execute the first case, i.e., do not recolor

themselves. As $v$ does not execute the second case, we obtain $d(v) < \Delta$. In order to show that $v$ can always select a color, i.e., that $F(v)\backslash\phi(N(v)) \neq \emptyset$ holds, we show that $|F(v)| \geq d(v) + 1$ holds. We lower bound the size of $F(v)$ in two cases.

*Case* $\Delta - d(v) \leq k - 1$*:* Let $Y = \{\phi(w) \mid w \in N(v), w \text{ executes the third case}\}$ be the set of input colors of neighbors of $v$ that execute the third case. We obtain $|Y| \leq \min\{k-1, \Delta - d(v)\}$ (the $k-1$ term appears as the input color of node $v$ cannot appear as an input color in its neighborhood). Let $X = [m]\backslash([\ell] \cup Y \cup \{\phi(v)\})$ be the set of input colors $\geq \ell, \neq \phi(v)$ that do not appear in the neighborhood of $v$. We have $|X| = (k-1) - |Y| \geq (k-1) - \min\{k-1, \Delta - d(v)\} = d(v) + k - \Delta - 1$. For each color in $X$ there is a corresponding regime from which $F(v)$ contains one color. We obtain

$$|F(v)| \geq |R_{\phi(v)-\ell}| + |X| \geq \Delta - k + 2 + d(v) + k - \Delta - 1 = d(v) + 1.$$

*Case* $\Delta - d(v) > k - 1$*:* The condition implies that $d(v) < \Delta - k + 1$ holds and we obtain

$$|F(v)| \geq |R_\phi(v) - \ell| = \Delta - k + 2 > d(v) + 1.$$

In both cases we obtain $|F(v)| \geq d(v) + 1$. Thus, in the last line of the algorithm, $v$ can pick a color in $F(v)$ not used by the $d(v)$ neighbors that do not recolor themselves, and due to Claim 4.2 there is no conflict with any neighbor that executes the third case. To finish the proof, recall that no neighbor executes the second case. □

Next, we show that the result of Lemma 4.1 is tight. The next lemma can be seen as a generalization of a result in [44], which proved a result of a similar flavor but only for $m \geq \frac{\Delta^2}{4} + \frac{\Delta}{2} + 1$.

Lemma 4.3 (Lower Bound for One-Round Algorithms). *Let $k, \Delta$, and $m$ be arbitrary integers satisfying $1 \leq k \leq \Delta - 1$ and $m \leq k(\Delta - k + 3) - 1$. Then, there is no one-round* LOCAL *algorithm that computes a $q = m - k$ coloring on every input $m$-colored graph with maximum degree $\Delta$.*

Proof. We may assume that $m = k(\Delta - k + 3) - 1$, as an impossibility result for this choice of $m$ implies the same result for any $m' \leq m$. Assume for contradiction that there is a one-round algorithm $\mathcal{A}$ that colors a graph with an input coloring with $m$ colors with $q = m - k$ output colors. Call an input color $\phi$ *sensitive* if for any output color $c \in [q]$ there is an input color $\phi' \in [m]$ such that if a node $v$ is input colored with $\phi$ and has a neighbor with input color $\phi'$, then node $v$ does not output $c$ (regardless of the colors of other neighbors). In the rest of the proof, our notation identifies nodes with their input color.

Claim 4.4. *There are at least $k$ sensitive input colors.*

Proof. We introduce the following definition where $c \in [m]$ is an output color. An input color $\phi$ is called *$c$-robust* if for all input colors $\phi' \neq \phi$ there is a set $A$ of size at most $\Delta$ satisfying $\phi' \in A$ and $\mathcal{A}((\phi, A)) = c$. Let $S = \bigcup_{c \in [q]} \{\phi \in [m] \mid \phi \text{ is } c\text{-robust}\}$ be the set of input colors that are $c$-robust for some $c$. Next, we upper bound the size of $S$ by $q$. In particular, we show that each of the sets in the union contains at most a single element. Assume for contradiction that $\phi$ and $\phi'$ are two distinct input colors that are $c$-robust. By definition, there are sets $A_\phi$ and $A_{\phi'}$ of size at most $\Delta$ satisfying $\phi' \in A_\phi$ and $\phi \in A_{\phi'}$ such that $c = \mathcal{A}((\phi, A_\phi)) = \mathcal{A}((\phi', A_{\phi'}))$, a contradiction to the correctness of $\mathcal{A}$ as $(\phi, A_\phi)$ and $(\phi', A_{\phi'})$ can be neighborhoods of neighboring nodes in some graph.

Let $T = [m]\backslash S$ and observe that $|T| \geq m - |S| \geq k$. For each $\phi \in T$ and for all $c \in [q]$, $\phi$ is not $c$-robust; that is, for all $c$ there exists an input color $\phi'$ such that for all sets $\phi' \in A$ of size at most $\Delta$, $\mathcal{A}(\phi, A) \neq c$. Hence, each color in $T$ is sensitive and the claim follows. □

Next, we construct a one-hop neighborhood that cannot be colored with one of the $q$ output colors and thus leads to a contradiction. Let $T$ be a set of sensitive input colors of size $k$ that exists

due to Claim 4.4. Consider the partial neighborhoods $N_x = (x, T\backslash\{x\})$ with one node $x \in T$ in the center and the $k-1$ other nodes in $T\backslash\{x\}$ as one-hop neighbors of $x$. This is a valid partial neighborhood as $|T\backslash\{x\}| \leq k-2 \leq \Delta$. We call a color $c \in [q]$ a *candidate color* for $N_x$ if there exists a set of input colors $B$ (of size $\leq \Delta$) satisfying $T\backslash\{x\} \subseteq B \subseteq [m]$ such that $\mathcal{A}((x,B)) = c$; i.e., algorithm $\mathcal{A}$ outputs color $c$ on the neighborhood $(x,B)$.

Claim 4.5. *For $x \neq x' \in T$ the sets of candidate colors of $N_x$ and $N_{x'}$ are disjoint.*

Proof. Assume for contradiction that $c \in [q]$ is a candidate color for $N_x$ and $N_{x'}$, and let $B \supseteq T\backslash\{x\}$ and $B' \supseteq T\backslash\{x'\}$ be the respective sets such that $c = \mathcal{A}((x,B)) = \mathcal{A}((x',B))$. As $x \in B'$ and $x' \in B$, the neighborhoods $(x,B)$ and $(x',B')$ can occur next to each other in a graph, a contradiction to the correctness of $\mathcal{A}$. □

By Claim 4.5 and by the pigeonhole principle, there exists one $x_* \in T$ for which $N_x$ has at most $\alpha$ candidate colors where

$$\alpha = \left\lfloor \frac{q}{|T|} \right\rfloor = \left\lfloor \frac{q}{k} \right\rfloor = \left\lfloor \frac{m-k}{k} \right\rfloor = \left\lfloor \frac{k\Delta - k^2 + 2k - 1}{k} \right\rfloor$$
$$= \left\lfloor \frac{k(\Delta - k + 2) - 1}{k} \right\rfloor = (\Delta - k + 2) - 1 = \Delta - (k-1).$$

Now, let $C_* \subseteq [q]$ be the set of candidate colors of $N_{x_*}$. As $x_*$ is in $T$ and all colors in $T$ are sensitive, for each $c \in C_*$ there exists some $\phi_c \in [m]$ such that $\mathcal{A}$ does not output $c$ for $v$ whenever $\phi_c$ is the input color of one of $v$'s neighbors. We conclude that the one-hop neighborhood $\tilde{N}_{x_*} = (x_*, \{\phi_c \mid c \in C_*\} \cup (T\backslash\{x\}))$ cannot be colored by $\mathcal{A}$, a contradiction. The choice of parameters is important. The constructed neighborhood $\tilde{N}_{x_*}$ is a feasible neighborhood as $|\{\phi_c \mid c \in C_*\} \cup (T\backslash\{x\})| \leq \alpha + k - 1 = \Delta$. □

Lemma 4.3 implies a heuristic lower bound of $\Omega(\Delta)$ to reduce a $\Delta^2/2$-coloring to a $\Delta^2/5$-coloring (if you have $\leq \Delta^2/4$ input colors you can remove at most $\Delta/2$ colors per iteration). In contrast, the algorithm from Corollary 1.2 (for a suitable choice of $k$) can reduce a $\Delta^4$-coloring to a $\Delta^2/5$ coloring in $O(1)$ rounds. Thus, the iterative application of *tight* bounds for one-round algorithms can be beaten significantly by a simple $O(1)$-round algorithm. This suggests that it is important to understand constant-time algorithms to settle the complexity of distributed graph coloring problems.

Proof of Theorem 1.6. The upper bound follows with Lemma 4.1. The lower bound states that we cannot reduce $k+1$ colors. It follows from Lemma 4.3 applied with $k+1$ (instead of $k$). □

## 5 CONCLUSION

In the current article we have seen a simple algorithm for distributed graph coloring in which each vertex locally computes a permutation of the output colors and then *tries* them in batches. A trial is *successful* if there is no *conflict*; that is, no neighbor tries the same color in the same round and no neighbor is already permanently colored with that color. Depending on the size of the batches, this algorithm scales between Linial's famous color reduction [50] and the locally iterative algorithm by Barenboim et al. [14]. If nodes tolerate conflicts up to a certain threshold, the same algorithm can be used to obtain the defective coloring algorithms of [16, 45] and [10, 16], as well as obtaining a simpler algorithm (as compared to [9, 14]) to compute low outdegree colorings aka *arbdefective colorings*. The latter are one of the two crucial ingredients in the state-of-the-art $(\Delta+1)$-coloring algorithm in [51]. The second ingredient is a two-round *list version* of Linial's color reduction, together with the observation that the degree bound can be replaced with a bound on the outdegree. One can also see our algorithm as an extension of Linial's algorithm, or the other way around: in the setting where nodes can only try one color per round ($k = 1$), one wants to get

colored with one out of $O(\Delta)$ output colors; this process is guaranteed to be successful in $O(\Delta)$ rounds if vertices try colors in a suitable order. Now, if you want to try more than one color per iteration, that is, you want to compress several rounds of the original algorithm into one iteration, you need to also mark each trial with the round number in which you would have tried it in the original algorithm, yielding an $O(\Delta^2)$-coloring if you want to execute all $O(\Delta)$ rounds in one iteration. We find it astonishing that in hindsight many crucial results in this area can be related to the algorithm that was presented in Linial's seminal paper [50] roughly 30 years ago.

On the lower bound side his initial $\Omega(\log^* n)$ bound is still the state of the art. The only progress is in terms of understanding one-round algorithms or weak variants of the LOCAL model [44]. Our article showed that there is a large discrepancy between iterating the best one-round algorithm for $O(1)$ times and what can be achieved by a *"smart"* algorithm that uses $O(1)$ rounds. This suggests that we first need to understand constant-time algorithms, from both the upper and lower bound side, before we can settle the complexity of the $(\Delta+1)$-coloring problem. Another approach would be to attack the coloring problem through lower bounds for ruling sets, as there is recent progress for the latter. A large lower bound for a $(2, r)$-ruling set would imply a lower bound for graph coloring via Lemma 3.2; however, it is unclear whether large enough lower bounds for ruling sets exist.

We end with an additional observation. We purposely keep the observation informal as we merely include it as an additional intuitive guide for the search of the *right* lower bound questions. We believe that a formal statement would actually hinder the creativity in this process.

Observation 5.1 (Informal). *Modulo a $\log\Delta$-factor the difficult part of the $(\Delta+1)$-coloring problem is to reduce a $(1+\varepsilon)\Delta$ coloring to a $(\Delta+1)$-coloring.*

Proof sketch. Assume an algorithm $\mathcal{A}$ that reduces the number of colors from $(1+\varepsilon)\Delta$ to $\Delta+1$. Now, assume an input coloring with $m \gg (1+\varepsilon\Delta)$ colors is given. Then, one can chop $[m]$ into $x \approx m/((1+\varepsilon)(\Delta+1))$ disjoint color spaces, each of size $(1+\varepsilon)\Delta$, and run $\mathcal{A}$ on each of them in parallel, using a disjoint output color space for each application. This uses $x \cdot (\Delta+1) \approx m/(1+\varepsilon)$ output colors; i.e., we have reduced the number of colors by a constant factor (if $\varepsilon$ is constant). Thus, if we begin with $m = O(\Delta^2)$ colors, we obtain a $(\Delta+1)$-coloring with a $O(\log_{1+\varepsilon}\Delta)$ multiplicative overhead. □

If $\Delta$ is a large enough $\operatorname{poly}\log n$, then a randomized algorithm can very efficiently compute $(1+\varepsilon)\Delta$-colorings [23, 41, 58].

## ACKNOWLEDGMENTS

We thank the several people with whom we have discussed the presented algorithm, in particular, Fabian Kuhn and Janosch Deurer. We also thank the anonymous reviewers who made a great effort to improve the presentation of this work.

## REFERENCES

[1] Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. 1989. Network decomposition and locality in distributed computation. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS'89)*. 364–369. https://doi.org/10.1109/SFCS.1989.63504

[2] Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. 2020. Classification of distributed binary labeling problems. In *Proceedings of the International Symposium on Distributed Computing (DISC'20)*. https://doi.org/10.4230/LIPIcs.DISC.2020.17

[3] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. 2019. Lower bounds for maximal matchings and maximal independent sets. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS'19)*. 481–497. https://doi.org/10.1145/3461458

[4] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. 2022. Distributed Δ-coloring plays hide-and-seek. In *54th Annual ACM SIGACT Symposium on Theory of Computing (STOC'22)*. ACM, 464–477. https://doi.org/10.1145/3519935.3520027

[5] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. 2022. Distributed edge coloring in time polylogarithmic in Δ. In *ACM Symposium on Principles of Distributed Computing (PODC'22)*. ACM, 15–25. https://doi.org/10.1145/3519270.3538440

[6] Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. 2020. Distributed lower bounds for ruling sets. In *61st IEEE Annual Symposium on Foundations of Computer Science (FOCS'20)*. 365–376. https://doi.org/10.1109/FOCS46700.2020.00042

[7] Alkida Balliu, Juho Hirvonen, Christoph Lenzen, Dennis Olivetti, and Jukka Suomela. 2019. Locality of not-so-weak coloring. In *Proceedings of the International Colloquium on Structural Information and Communication Complexity (SIROCCO'19)*. 37–51. https://doi.org/10.1007/978-3-030-24922-9_3

[8] Philipp Bamberger, Fabian Kuhn, and Yannic Maus. 2020. Efficient deterministic distributed coloring with small bandwidth. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC'20)*. https://doi.org/10.1145/3382734.3404504

[9] Leonid Barenboim. 2016. Deterministic (Δ + 1)-coloring in sublinear (in Δ) time in static, dynamic, and faulty networks. *Journal of the ACM* 63, 5 (2016), 47:1–47:22. https://doi.org/10.1145/2979675

[10] Leonid Barenboim and Michael Elkin. 2009. Distributed (Delta+1)-Coloring in linear (in Delta) time. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC'09)*. 111–120. https://doi.org/10.1145/1536414.1536432

[11] Leonid Barenboim and Michael Elkin. 2010. Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. *Distributed Computing* 22, 5-6 (2010), 363–379. https://doi.org/10.1007/s00446-009-0088-2

[12] Leonid Barenboim and Michael Elkin. 2011. Deterministic distributed vertex coloring in polylogarithmic time. *Journal of the ACM* 58, 5 (2011), 23:1–23:25. https://doi.org/10.1145/2027216.2027221

[13] Leonid Barenboim and Michael Elkin. 2013. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Morgan & Claypool Publishers.

[14] Leonid Barenboim, Michael Elkin, and Uri Goldenberg. 2018. Locally-iterative distributed (Δ+ 1)-coloring below Szegedy-Vishwanathan barrier, and applications to self-stabilization and to restricted-bandwidth models. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC'18)*. 437–446. https://doi.org/10.1145/3212734.3212769

[15] Leonid Barenboim, Michael Elkin, and Uri Goldenberg. 2021. Locally-iterative distributed (Î$\dot{T}$ + 1)-coloring and applications. *Journal of the ACM* 69, 1, Article 5 (Dec. 2021), 26 pages. https://doi.org/10.1145/3486625

[16] Leonid Barenboim, Michael Elkin, and Fabian Kuhn. 2014. Distributed (Delta+1)-coloring in linear (in Delta) time. *SIAM Journal on Computing* 43, 1 (2014), 72–95. https://doi.org/10.1137/12088848X

[17] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. 2016. The locality of distributed symmetry breaking. *Journal of the ACM* 63, 3 (2016), 20:1–20:45. https://doi.org/10.1145/2903137

[18] Anton Bernshteyn. 2022. A fast distributed algorithm for Δ + 1-edge-coloring. *Journal of Combinatorial Theory, Series B* 152 (2022), 319–352. https://doi.org/10.1016/j.jctb.2021.10.004

[19] Sebastian Brandt. 2019. An automatic speedup theorem for distributed problems. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC'19)*. 379–388. https://doi.org/10.1145/3293611.3331611

[20] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. 2016. A lower bound for the distributed Lovász local lemma. In *Proceedings of the ACM Symposium on Theory of Computing (STOC'16)*. 479–488. https://doi.org/10.1145/2897518.2897570

[21] Sebastian Brandt and Dennis Olivetti. 2020. Truly tight-in-Δ bounds for bipartite maximal matching and variants. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC'20)*. https://doi.org/10.1145/3382734.3405745

[22] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. 2019. An exponential separation between randomized and deterministic complexity in the LOCAL Model. *SIAM Journal on Computing* 48, 1 (2019), 122–143. https://doi.org/10.1137/17M1117537

[23] Yi-Jun Chang, Wenzheng Li, and Seth Pettie. 2020. Distributed (Δ + 1)-coloring via ultrafast graph shattering. *SIAM Journal on Computing* 49, 3 (2020), 497–539. https://doi.org/10.1137/19M1249527

[24] Michael Elkin and Shaked Matar. 2019. Near-additive spanners in low polynomial deterministic CONGEST time. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC'19)*. 531–540. https://doi.org/10.1145/3293611.3331635

[25] Paul Erdös, Peter Frankl, and Zóltan Füredi. 1985. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics* 51, 1 (1985), 79–89.

[26] Salwa Faour, Mohsen Ghaffari, Christoph Grunau, Fabian Kuhn, and Václav Rozhon. 2023. Local distributed rounding: Generalized to MIS, matching, set cover, and beyond. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'23)*. SIAM, 4409–4447. https://doi.org/10.1137/1.9781611977554.ch168

[27] Manuela Fischer. 2020. Improved deterministic distributed matching via rounding. *Distributed Computing* 33, 3–4 (2020), 279–291. https://doi.org/10.1007/s00446-018-0344-4

[28] Manuela Fischer, Magnús M. Halldórsson, and Yannic Maus. 2023. Fast distributed brooks' theorem. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'23)*. SIAM. https://doi.org/10.1137/1.9781611977554.ch98

[29] Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. 2016. Local conflict coloring. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS'16)*. 625–634. https://doi.org/10.1109/FOCS.2016.73

[30] Pierre Fraigniaud and Ami Paz. 2020. The topology of local computing in networks. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP'20)*. 128:1–128:18. https://doi.org/10.4230/LIPIcs.ICALP.2020.128

[31] Mohsen Ghaffari, Christoph Grunau, Bernhard Haeupler, Saeed Ilchi, and Václav Rozhon. 2023. Improved distributed network decomposition, hitting sets, and spanners, via derandomization. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'23)*. SIAM, 2532–2566. https://doi.org/10.1137/1.9781611977554.ch97

[32] Mohsen Ghaffari, Christoph Grunau, and Václav Rozhon. 2021. Improved deterministic network decomposition. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'21)*. SIAM, 2904–2923. https://doi.org/10.1137/1.9781611976465.173

[33] Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. 2018. On derandomizing local distributed algorithms. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS'18)*. 662–673. https://doi.org/10.1109/FOCS.2018.00069

[34] Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, and Yannic Maus. 2018. Improved distributed delta-coloring. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC'18)*. 427–436. https://doi.org/10.1145/3212734.3212764

[35] Mohsen Ghaffari and Fabian Kuhn. 2021. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. In *62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS'21)*. IEEE, 1009–1020. https://doi.org/10.1109/FOCS52979.2021.00101

[36] Mohsen Ghaffari, Fabian Kuhn, Yannic Maus, and Jara Uitto. 2018. Deterministic distributed edge-coloring with fewer colors. In *Proceedings of the ACM Symposium on Theory of Computing (STOC'18)*. 418–430. https://doi.org/10.1145/3188745.3188906

[37] Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Tigran Tonoyan. 2021. Efficient randomized distributed coloring in CONGEST. In *Proceedings of the ACM Symposium on Theory of Computing (STOC'21)*. https://doi.org/10.1145/3406325.3451089 arXiv:2012.14169

[38] Magnús M. Halldórsson, Fabian Kuhn, Alexandre Nolin, and Tigran Tonoyan. 2022. Near-optimal distributed degree+1 coloring. In *54th Annual ACM SIGACT Symposium on Theory of Computing (STOC'22)*. ACM, 450–463. https://doi.org/10.1145/3519935.3520023

[39] Magnús M. Halldórsson, Yannic Maus, and Alexandre Nolin. 2022. Fast distributed vertex splitting with applications. In *36th International Symposium on Distributed Computing (DISC'22)*. https://doi.org/10.4230/LIPIcs.DISC.2022.26

[40] Magnús M. Halldórsson, Alexandre Nolin, and Tigran Tonoyan. 2022. Overcoming congestion in distributed coloring. In *ACM Symposium on Principles of Distributed Computing (PODC '22)*. ACM, 26–36. https://doi.org/10.1145/3519270.3538438

[41] MagnÃžs M. HalldÃşrsson and Alexandre Nolin. 2021. Superfast coloring in CONGEST via efficient color sampling. In *28th International Colloquium on Structural Information and Communication Complexity*. https://doi.org/10.1007/978-3-030-79527-6_5

[42] David G. Harris. 2019. Distributed local approximation algorithms for maximum matching in graphs and hypergraphs. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS'19)*. 700–724. https://doi.org/10.1137/19M1279241

[43] David G. Harris, Johannes Schneider, and Hsin-Hao Su. 2018. Distributed ($\Delta$ +1)-coloring in sublogarithmic rounds. *Journal of the ACM* 65, 4 (2018), 19:1–19:21. https://doi.org/10.1145/3178120

[44] Dan Hefetz, Fabian Kuhn, Yannic Maus, and Angelika Steger. 2016. Polynomial lower bound for distributed graph coloring in a weak LOCAL model. In *Proceedings of the International Symposium on Distributed Computing (DISC'16)*. 99–113. https://doi.org/10.1007/978-3-662-53426-7_8

[45] Fabian Kuhn. 2009. Weak graph colorings: Distributed algorithms and applications. In *Proceedings of the ACM Symposium on Parallelism in Algorithms and Architecture (SPAA'09)*. 138–144. https://doi.org/10.1145/1583991.1584032

[46] Fabian Kuhn. 2020. Faster deterministic distributed coloring through recursive list coloring. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*. 1244–1259. https://doi.org/10.1137/1.9781611975994.76

[47] Fabian Kuhn, Yannic Maus, and Simon Weidner. 2018. Deterministic distributed ruling sets of line graphs. In *Structural Information and Communication Complexity - 25th International Colloquium (SIROCCO'18), Revised Selected Papers*. 193–208. https://doi.org/10.1007/978-3-030-01325-7_19

[48] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 2016. Local computation: Lower and upper bounds. *Journal of the ACM* 63, 2 (2016). https://doi.org/10.1145/2742012

[49] Fabian Kuhn and Roger Wattenhofer. 2006. On the complexity of distributed graph coloring. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC'06)*. 7–15. https://doi.org/10.1145/1146381.1146387

[50] Nathan Linial. 1992. Locality in distributed graph algorithms. *SIAM Journal on Computing* 21, 1 (1992), 193–201. https://doi.org/10.1137/0221015

[51] Yannic Maus and Tigran Tonoyan. 2020. Local conflict coloring revisited: Linial for lists. In *34th International Symposium on Distributed Computing (DISC'20)*. 16:1–16:18. https://doi.org/10.4230/LIPIcs.DISC.2020.16

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

[53] Dennis Olivetti. 2020. Brief announcement: Round eliminator: A tool for automatic speedup simulation. In *ACM Symposium on Principles of Distributed Computing (PODC'20)*. 352–354. https://doi.org/10.1145/3382734.3405694

[54] Alessandro Panconesi and Aravind Srinivasan. 1992. Improved distributed algorithms for coloring and network decomposition problems. In *Proceedings of the ACM Symposium on Theory of Computing (STOC'92)*. 581–592. https://doi.org/10.1145/129712.129769

[55] David Peleg. 2000. *Distributed Computing: A Locality-sensitive Approach*. SIAM.

[56] Václav Rozhon and Mohsen Ghaffari. 2020. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proceedings of the ACM Symposium on Theory of Computing (STOC'20)*. 350–363. https://doi.org/10.1145/3357713.3384298

[57] Johannes Schneider, Michael Elkin, and Roger Wattenhofer. 2013. Symmetry breaking depending on the chromatic number or the neighborhood growth. *Theoretical Computer Science* 509 (2013), 40–50. https://doi.org/10.1016/j.tcs.2012.09.004

[58] Johannes Schneider and Roger Wattenhofer. 2010. A new technique for distributed symmetry breaking. In *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing (PODC'10)*. ACM, 257–266. https://doi.org/10.1145/1835698.1835760

[59] Hsin-Hao Su and Hoa T. Vu. 2019. Towards the locality of Vizing's theorem. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'19)*. 355–364. https://doi.org/10.1145/3313276.3316393

[60] Mario Szegedy and Sundar Vishwanathan. 1993. Locality based graph coloring. In *Proceedings of the ACM Symposium on Theory of Computing (STOC'93)*. 201–207. https://doi.org/10.1145/167088.167156

[61] V. G. Vizing. 1964. On an estimate of the chromatic class of a p-graph. *Diskret analiz* 3 (1964), 25–30.

Received 16 December 2021; accepted 21 June 2023