LOCAL MULTICOLORING ALGORITHMS:*
COMPUTING A NEARLY-OPTIMAL TDMA SCHEDULE IN CONSTANT TIME

Fabian Kuhn
Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology
Cambridge, MA 02139
USA
fkuhn@csail.mit.edu

Abstract

We are given a set V of autonomous agents (e.g. the computers of a distributed system) that are connected to each other by a graph $G = (V, E)$ (e.g. by a communication network connecting the agents). Assume that all agents have a unique ID between 1 and N for a parameter $N \geq |V|$ and that each agent knows its ID as well as the IDs of its neighbors in G. Based on this limited information, every agent v must autonomously compute a set of colors $S_v \subseteq C$ such that the color sets S_u and S_v of adjacent agents u and v are disjoint. We prove that there is a deterministic algorithm that uses a total of $|C| = \mathcal{O}(\Delta^2 \log(N)/\varepsilon^2)$ colors such that for every node v of G (i.e., for every agent), we have $|S_v| \geq |C| \cdot (1 - \varepsilon)/(\delta_v + 1)$, where δ_v is the degree of v and where Δ is the maximum degree of G. For $N = \Omega(\Delta^2 \log \Delta)$, $\Omega(\Delta^2 + \log \log N)$ colors are necessary even to assign at least one color to every node (i.e., to compute a standard vertex coloring). Using randomization, it is possible to assign an $(1-\varepsilon)/(\delta+1)$-fraction of all colors to every node of degree δ using only $\mathcal{O}(\Delta \log |V|/\varepsilon^2)$ colors w.h.p. We show that this is asymptotically almost optimal. For graphs with maximum degree $\Delta = \Omega(\log |V|)$, $\Omega(\Delta \log |V| / \log \log |V|)$ colors are needed in expectation, even to compute a valid coloring.

The described multicoloring problem has direct applications in the context of wireless ad hoc and sensor networks. In order to coordinate the access to the shared wireless medium, the nodes of such a network need to employ some medium access control (MAC) protocol. Typical MAC protocols control the access to the shared channel by time (TDMA), frequency (FDMA), or code division multiple access (CDMA) schemes. Many channel access schemes assign a fixed set of time slots, frequencies, or (orthogonal) codes to the nodes of a network such that nodes that interfere with each other receive disjoint sets of time slots, frequencies, or code sets. Finding a valid assignment of time slots, frequencies, or codes hence directly corresponds to computing a multicoloring of a graph G. The scarcity of bandwidth, energy, and computing resources in ad hoc and sensor networks, as well as the often highly dynamic nature of these networks require that the multicoloring can be computed based on as little and as local information as possible.

*An extended abstract of this paper appeared in the proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS 2009)

1 Introduction

In this paper, we look at a variant of the standard vertex coloring problem that we name graph *multicoloring*. Given an n-node graph $G = (V, E)$, the goal is to assign a set S_v of colors to each node $v \in V$ such that the color sets S_u and S_v of two adjacent nodes $u \in V$ and $v \in V$ are disjoint while at the same time, the fraction of colors assigned to each node is as large as possible and the total number of colors used is as small as possible. In particular, we look at the following *distributed* variant of this multicoloring problem. Each node has a unique identifier (ID) between 1 and N for an integer parameter $N \geq n$. The nodes are *autonomous agents* and we assume that every agent has only very limited, *local* information about G. Specifically, we assume that every node $v \in V$ merely knows its own ID as well as the IDs of all its neighbors. Based on this local information, every node v needs to compute a color set S_v such that the color sets computed by adjacent nodes are disjoint. Since our locality condition implies that every node is allowed to communicate with each neighbor only once, we call such a a distributed algorithm a *one-shot algorithm*.

We prove nearly tight upper and lower bounds for deterministic and randomized algorithms solving the above distributed multicoloring problem. Let Δ be the largest degree of G. We show that for every $\varepsilon \in (0, 1)$, there is a deterministic multicoloring algorithm that uses $\mathcal{O}(\Delta^2 \log(N)/\varepsilon^2)$ colors and assigns a $(1-\varepsilon)/(\delta+1)$-fraction of all colors to each node of degree δ. Note that because a node v of degree δ does not know anything about the topology of G (except that itself has δ neighbors), no one-shot multicoloring algorithm can assign more than a $1/(\delta + 1)$-fraction of the colors to all nodes of degree δ (the nodes could be in a clique of size $\delta + 1$). The upper bound proof is based on the probabilistic method and thus only establishes the existence of an algorithm. We describe an algebraic construction yielding an explicit algorithm that achieves the same bounds up to polylogarithmic factors. Using $\mathcal{O}(\Delta^2 \log^2 N)$ colors, for a value $\varepsilon > 0$, the algorithm assigns a $\varepsilon/\mathcal{O}(\delta^{1+\varepsilon} \log N)$-fraction of all colors to nodes of degree δ. At the cost of using $\mathcal{O}(\Delta^{\log^* N} \log N)$ colors, it is even possible to improve the fraction of colors assigned to each node by a factor of $\log N$. The deterministic upper bound results are complemented by a lower bound showing that if $N = \Omega(\Delta^2 \log \Delta)$, even for the standard vertex coloring problem, every deterministic one-shot algorithm needs to use at least $\Omega(\Delta^2 + \log\log N)$ colors.

If we allow the nodes to use randomization (and only require that the claimed bounds are obtained with high probability), we can do significantly better. In a randomized one-shot algorithm, we assume that every node can compute a sequence of random bits at the beginning of an algorithm and that nodes also know their own random bits as well as the random bits of the neighbors when computing the color set. We show that for $\varepsilon \in (0, 1)$, with high probability, $\mathcal{O}(\Delta \log(n)/\varepsilon^2)$ colors suffice to assign a $(1-\varepsilon)/(\delta+1)$-fraction of all colors to every node of degree δ. If $\log n \leq \Delta \leq n^{1-\varepsilon}$ for a constant $\varepsilon > 0$, we show that every randomized one-shot algorithm needs at least $\Omega(\Delta \log n/ \log\log n)$ colors. Again, the lower bound even holds for standard vertex coloring algorithms where every node only needs to choose a single color.

Synchronizing the access to a common resource is a typical application of coloring in networks. If we have a c-coloring of the network graph, we can partition the resource (and/or time) into c parts and assign a part to each node v depending on v's color. In such a setting, it seems natural to use a multicoloring instead of a standard vertex coloring and assign more than one part of the resource to every node. This allows to use the resource more often and thus more efficiently.

The most prominent specific example of this basic approach occurs in the context of media access control (MAC) protocols for wireless ad hoc and sensor networks. These networks consist of autonomous wireless devices that communicate with each other by the use of radio signals. If two or more close-by nodes transmit radio signals at the same time, a receiving node only hears the superposition of all transmitted signals. Hence, simultaneous transmissions of close-by nodes interfere with each other and we thus have to control the access to the wireless channel. A standard way to avoid interference between close-by transmissions is to use a time (TDMA), frequency (FDMA), or code division multiple access (CDMA) scheme to divide the channel among the nodes. A TDMA protocol divides the time into time slots and assigns different time slots to conflicting nodes. When using FDMA, nodes that can interfere with each other are assigned different frequencies, whereas a CDMA scheme uses different (orthogonal) codes for interfering nodes. Classically, TDMA, FDMA, and CDMA protocols are implemented by a standard vertex coloring of the graph induced by the interference relations. In all three cases, it would be natural to use the more general multicoloring problem in order to achieve a more effective use of the wireless medium. Efficient TDMA schedules, FDMA frequency assignments, or CDMA code assignments are all directly obtained from a multicoloring of the interference graph where the fraction of colors assigned to each nodes is as large as possible. It is also natural to require that the total number of colors is small. This keeps the length of a TDMA schedule or the total number of frequencies or codes small and thus helps to improve the efficiency and reduce unnecessary overhead of the resulting MAC protocols.

In contrast to many wired networks, wireless ad hoc and sensor networks typically consist of small devices that have limited computing and storage capabilities. Because these devices operate on batteries, wireless nodes also have to keep the amount of computation and especially communication to a minimum in order to save energy and thus increase their lifetime. As the nodes of an ad hoc or sensor network need to operate without central control, everything that is computed, has to be computed by a distributed algorithm by the nodes themselves. Coordination between the nodes is achieved by exchanging messages. Because of the resource constraints, these distributed algorithms need to be as simple and efficient as possible. The messages transmitted and received by each node should be as few and as short as possible. Note that because of interference, the bandwidth of each local region is extremely limited. Typically, for a node v, the time needed to even receive a single message from all neighbors is proportional to the degree of v (see e.g. [19]). As long as the information provided to each node is symmetric, it is clear that every node needs to know the IDs of all adjacent nodes in G in order to compute a reasonably good multicoloring of G. Hence, the one-shot multicoloring algorithms considered in this paper base their computations on the minimum information needed to compute a non-trivial solution to the problem. Based on the above observations, even learning the IDs of all neighbors requires quite a bit of time and resources. Hence, acquiring significantly more information might already render an algorithm inapplicable in practice.[1]

As a result of the scarcity of resources, the size and simplicity of the wireless devices used in sensor networks, and the dependency of the characteristic of radio transmissions on environmental conditions, ad hoc and sensor networks are much less stable than usual wired networks. As a consequence, the topology of these networks (and of their interference graph) can be highly dynamic. This is especially true for ad hoc networks, where it is often even assumed that the nodes are mobile and thus can move in space. In order to adapt to such dynamic conditions, a multicoloring needs to be recomputed periodically. This makes the resource and time efficiency of the used algorithms even more important. This is particularly true for the locality of the algorithms. If the computation of every node only depends on the topology of a close-by neighborhood, dynamic changes also only affect near-by nodes.

The remainder of the paper is organized as follows. In Section 2, we discuss related work. The problem is formally defined in Section 3. We present the deterministic and randomized upper bounds in Section 4 and the lower bounds in Section 5.

[1]It seems that in order to achieve a significant improvement on the multicolorings computed by the algorithms presented in this paper, every node would need much more information. Even if every node knows its complete $O(\log \Delta)$-neighborhood, the best deterministic coloring algorithm that we are aware of needs $\Theta(\Delta^2)$ colors.

2 Related Work

There is a rich literature on distributed algorithms to compute classical vertex colorings (see e.g. [1, 4, 11, 15, 16, 21]). The paper most related to the present one is [15]. In [15], deterministic algorithms for the standard coloring problem in the same distributed setting are studied (i.e., every node has to compute its color based on its ID and the IDs of its neighbors). The main result is a $\Omega(\Delta^2/\log^2\Delta)$ lower bound on the number of colors. The first paper to study distributed coloring is a seminal paper by Linial [16]. The main result of [16] is an $\Omega(\log^* n)$-time lower bound for coloring a ring with a constant number of colors. As a corollary of this lower bound, one obtains an $\Omega(\log\log N)$ lower bound on the number of colors for deterministic one-shot coloring algorithms as studied in this paper. Linial also looks at distributed coloring algorithms for general graph and shows that one can compute an $\mathcal{O}(\Delta^2)$-coloring in time $\mathcal{O}(\log^* n)$. In order to color a general graph with less colors, the best known distributed algorithms are significantly slower.[2] Using randomization, an $\mathcal{O}(\Delta)$-coloring can be obtained in time $\mathcal{O}(\sqrt{\log n})$ [14]. Further, the fastest algorithm to obtain a $(\Delta+1)$-coloring is based on an algorithm to compute a maximal independent set by Luby [17] and on a reduction described in [16] and has time complexity $\mathcal{O}(\log n)$. The best known deterministic algorithms to compute a $(\Delta+1)$-coloring have time complexities $2^{\mathcal{O}(\sqrt{\log n})}$ and $\mathcal{O}(\Delta\log\Delta+\log^* n)$ and are described in [21] and [15], respectively. For special graph classes, there are more efficient deterministic algorithms. It has long been known that in rings [4] and bounded degree graphs [11, 16], a $(\Delta+1)$-coloring can be computed in time $\mathcal{O}(\log^* n)$. Very recently, it has been shown that this also holds for the much larger class of graphs with bounded local independent sets [26]. In particular, this graph class contains all graph classes that are typically used to model wireless ad hoc and sensor networks. Another recent result shows that graphs of bounded arboricity can be colored with a constant number of colors in time $\mathcal{O}(\log n)$ [3].

Closely related to vertex coloring algorithms are distributed algorithms to compute edge colorings [5, 12, 22]. In a seminal paper, Naor and Stockmeyer were the first to look at distributed algorithms where all nodes have to base their decisions on constant neighborhoods [20]. It is shown that a weak coloring with $f(\Delta)$ colors (every node needs to have a neighbor with a different color) can be computed in time 2 if every vertex has an odd degree. Another interesting approach is taken in [9] where the complexity of distributed coloring is studied in case there is an oracle that gives some nodes a few bits of extra information.

There are many papers that propose to use some graph coloring variant in order to compute TDMA schedules and FDMA frequency or CDMA code assignments (see e.g. [2, 10, 13, 18, 24, 25, 27]). Many of these papers compute a vertex coloring of the network graph such that nodes at distance at most 2 have different colors. This guarantees that no two neighbors of a node use the same time slot, frequency, or code. Some of the papers also propose to construct a TDMA schedule by computing an edge coloring and using different time slots for different edges. Clearly, it is straight-forward to use our algorithms for edge colorings, i.e., to compute a multicoloring of the line graph. With the exception of [13] all these papers compute a coloring and assign only one time slot, frequency, or code to every node or edge. In [13], first, a standard coloring is computed. Based on this coloring, an improved slot assignment is constructed such that in the end, the number of slots assigned to a node is inversely proportional to the number of colors in its neighborhood.

[2] In [6], it is claimed that an $\mathcal{O}(\Delta)$ coloring can be computed in time $\mathcal{O}(\log^*(n/\Delta))$. However, the argumentation in [6] has a fundamental flaw that cannot be fixed [23].

3 Formal Problem Description

3.1 Mathematical Preliminaries

Throughout the paper, we use $\log(\cdot)$ to denote logarithms to base 2 and $\ln(\cdot)$ to denote natural logarithms, respectively. By $\log^{(i)} x$ and by $\ln^{(i)} x$, we denote the i-fold applications of the logarithm functions log and ln to x, respectively[3]. The log star function is defined as $\log^* n := \min_i\{\log^{(i)} n \leq 1\}$. We also use the following standard notations. For an integer $n \geq 1$, $[n] = \{1, \dots, n\}$. For a finite set Ω and an integer $k \in \{0, \dots, |\Omega|\}$, $\binom{\Omega}{k} = \{S \in 2^\Omega : |S| = k\}$. The term with high probability (w.h.p.) means with probability at least $1 - 1/n^c$ for a constant $c \geq 1$.

3.2 Multicoloring

The multicoloring problem that was introduced in Section 1 can be formally defined as follows.

Definition 3.1 (Multicoloring). *An $(\rho(\delta), k)$-multicoloring γ of a graph $G = (V, E)$ is a mapping $\gamma : V \to 2^{[k]}$ that assigns a set $\gamma(v) \subset [k]$ of colors to each node v of G such that $\forall\{u, v\} \in E : \gamma(u) \cap \gamma(v) = \emptyset$ and such that for every node $v \in V$ of degree δ, $|\gamma(v)|/k \geq \rho(\delta)/(\delta + 1)$.*

We call $\rho(\delta)$ the *approximation ratio* of a $(\rho(\delta), k)$-multicoloring. Because in a one-shot algorithm (cf. the next section for a formal definition), a node of degree δ cannot distinguish G from $K_{\delta+1}$, the approximation ratio of every one-shot algorithm needs to be at most 1.

The multicoloring problem is related to the fractional coloring problem in the following way. Assume that every node is assigned the same number c of colors and that the total number of colors is k. Taking every color with fraction $1/c$ then leads to a fractional (k/c)-coloring of G. Hence, in this case, k/c is lower bounded by the fractional chromatic number $\chi_f(G)$ of G.

3.3 One-Shot Algorithms

As outlined in the introduction, we are interested in local algorithms to compute multicolorings of an n-node graph $G = (V, E)$. For a parameter $N \geq n$, we assume that every node v has a unique ID $x_v \in [N]$. In deterministic algorithms, every node has to compute a color set based on its own ID as well as the IDs of its neighbors. For randomized algorithms, we assume that nodes also know the random bits of their neighbors. Formally, a one-shot algorithm can be defined as follows.

Definition 3.2 (One-Shot Algorithm). *We call a distributed algorithm a one-shot algorithm if every node v performs (a subset of) the following three steps:*

1. *Generate sequence R_v of random bits (deterministic algorithms: $R_v = \emptyset$)*
2. *Send x_v, R_v to all neighbors*
3. *Compute solution based on x_v, R_v, and the received information*

Assume that G is a network graph such that two nodes u and v can directly communicate with each other iff they are connected by an edge in G. In the standard *synchronous message passing* model, time is divided into rounds and in every round, every node of G can send a message to each of its neighbors. One-shot algorithms then exactly correspond to computations that can be carried out in a single communication round.

[3]We have $\log^{(0)} x = \ln^{(0)} x = x$, $\log^{(i+1)} x = \log(\log^{(i)} x)$, and $\ln^{(i+1)} x = \ln(\ln^{(i)} x)$. Note that we also use $\log^i x = (\log x)^i$ and $\ln^i x = (\ln x)^i$

For deterministic one-shot algorithms, the output of every node v is a function of v's ID x_v and the IDs of v's neighbors. We call this information on which v bases its decisions, the *one-hop view* of v.

Definition 3.3 (One-Hop View)**.** *Consider a node v with ID x_v and let Γ_v be the set of IDs of the neighbors of v. We call the pair (x_v, Γ_v) the one-hop view of v.*

Let (x_u, Γ_u) and (x_v, Γ_v) be the one-hop views of two adjacent nodes. Because u and v are neighbors, we have $x_u \in \Gamma_v$ and that $x_v \in \Gamma_u$. It is also not hard to see that

$$\forall x_u, x_v \in [N] \text{ and } \forall \Gamma_u, \Gamma_v \in 2^{[N]} \text{ such that } x_u \neq x_v, x_u \in \Gamma_v \setminus \Gamma_u, x_v \in \Gamma_u \setminus \Gamma_v, \tag{1}$$

there is a labeled graph that has two adjacent nodes u and v with one-hop views (x_u, Γ_u) and (x_v, Γ_v), respectively. Assume that we are given a graph with maximum degree Δ (i.e., for all one-hop views (x_v, Γ_v), we have $|\Gamma_v| \leq \Delta$). A one-shot vertex coloring algorithm maps every possible one-hop view to a color. A correct coloring algorithm must assign different colors to two one-hop views (x_u, Γ_u) and (x_v, Γ_v) iff they satisfy Condition (1). This leads to the definition of the *neighborhood graph* $\mathcal{N}_1(N, \Delta)$ [15] (the general notion of neighborhood graphs has been introduced in [16]). The nodes of $\mathcal{N}_1(N, \Delta)$ are all one-hop views (x_v, Γ_v) with $|\Gamma_v| \leq \Delta$. There is an edge between (x_u, Γ_u) and (x_v, Γ_v) iff the one-hop views satisfy Condition (1). Hence, a one-shot coloring algorithm must assign different colors to two one-hop views iff they are neighbors in $\mathcal{N}_1(N, \Delta)$. The number of colors that are needed to properly color graphs with maximum degree Δ by a one-shot algorithm therefore exactly equals the chromatic number $\chi\big(\mathcal{N}_1(N, \Delta)\big)$ of the neighborhood graph (see [15, 16] for more details). Similarly, a one-shot $(\rho(\delta), k)$-multicoloring algorithm corresponds to a $(\rho(\delta), k)$-multicoloring of the neighborhood graph.

4 Upper Bounds

In this section, we prove all the upper bounds claimed in Section 1. We first prove that an efficient deterministic one-shot multicoloring algorithm exists in Section 4.1. Based on similar ideas, we derive an almost optimal randomized algorithm in Section 4.2. Finally, in Section 4.3, we introduce constructive methods to obtain one-shot multicoloring algorithms. For all algorithms, we assume that the nodes know the size of the ID space N as well as Δ, an upper bound on the largest degree in the network. It certainly makes sense that nodes are aware of the used ID space. Note that it is straight-forward to see that there cannot be a non-trivial solution to the one-shot multicoloring problem if the nodes do not have an upper bound on the maximum degree in the network.

4.1 Existence of an Efficient Deterministic Algorithm

The existence of an efficient, deterministic one-shot multicoloring algorithm is established by the following theorem.

Theorem 4.1. *Assume that we are given a graph with maximum degree Δ and node IDs in $[N]$. Then, for all $0 < \varepsilon \leq 1$, there is a deterministic, one-shot $\big(1 - \varepsilon, \mathcal{O}(\Delta^2 \log(N)/\varepsilon^2)\big)$-multicoloring algorithm.*

Proof. We use permutations to construct colors as described in [15]. For $i = 1, \dots, k$, let $\prec_i$ be a global order on the ID set $[N]$. A node v with 1-hop view (x_v, Γ_v) includes color i in its color set iff $\forall y \in \Gamma_v$: $x_v \prec_i y$. It is clear that with this approach the color sets of adjacent nodes are disjoint. In order to show that nodes of degree δ obtain a $\rho/(\delta+1)$-fraction of all colors, we need to show that for all $\delta \in [\Delta]$, all

$x \in [N]$, and all $\Gamma \in \binom{[N]\setminus\{x\}}{\delta}$, for all $y \in \Gamma$, $x \prec_i y$ for at least $k\rho/(\delta+1)$ global orders $\prec_i$. We use the probabilistic method to show that a set of size $k = 2(\Delta+1)^2\ln(N)/\varepsilon^2$ of global orders $\prec_i$ exists such that every node of degree $\delta \in [\Delta]$ gets at least an $(1-\varepsilon)/(\delta+1)$-fraction of the k colors. Such a set implies that there exists an algorithm that satisfies the claimed bounds for all graphs with maximum degree Δ and IDs in $[N]$.

Let $\prec_1, \ldots, \prec_k$ be k global orders chosen independently and uniformly at random. The probability that a node v with degree δ and 1-hop view (x_v, Γ_v) gets color i is $1/(\delta+1)$ (note that $|\Gamma_v| = \delta$). Let X_v be the number of colors that v gets. We have $\mathbb{E}[X_v] = k/(\delta+1) \geq k/(\Delta+1)$. Using a Chernoff bound, we then obtain

$$\mathbb{P}\left[X_v < (1-\varepsilon)\cdot\frac{k}{\delta+1}\right] = \mathbb{P}\left[X_v < (1-\varepsilon)\cdot\mathbb{E}[X_v]\right] < e^{-\varepsilon^2\mathbb{E}[X_v]/2} \leq e^{-\varepsilon^2 k/(2(\Delta+1))} = \frac{1}{N^{\Delta+1}}. \quad (2)$$

The total number of different possible one-hop views can be bounded as $|\mathcal{N}_1(N,\Delta)| = N\cdot\sum_{\delta=1}^{\Delta}\binom{N-1}{\delta} < N^{\Delta+1}$. By a union bound argument, we therefore get that with positive probability, for all $\delta \in [\Delta]$, all possible one-hop views (x_v, Γ_v) with $|\Gamma_v| = \delta$ get at least $(1-\varepsilon)\cdot k/(\delta+1)$ colors. Hence, there exists a set of k global orders on the ID set $[N]$ such that all one-hop views obtain at least the required number of colors. □

Remark: Note that if we increase the number of permutations (i.e., the number of colors) by a constant factor, all possible one-hop views (x, Γ) with $|\Gamma| = \delta$ get a $(1-\varepsilon)/(\delta+1)$-fraction of all colors w.h.p.

4.2 Randomized Algorithms

We will now show that with the use of randomization, the upper bound of Section 4.1 can be significantly improved if the algorithm only needs to be correct w.h.p. We will again use random permutations. The problem of the deterministic algorithm is that the algorithm needs to assign a large set of colors to all roughly N^Δ possible one-hop views. With the use of randomization, we essentially only have to assign colors to n randomly chosen one-hop views.

For simplicity, we assume that every node knows the number of nodes n (knowing an upper bound on n is sufficient). For an integer parameter $k > 0$, every $v \in V$ chooses k independent random numbers $x_{v,1}, \ldots, x_{v,k} \in [kn^4]$ and sends these random numbers to all neighbors. We use these random numbers to induce k random permutations on the nodes. Let $\Gamma(v)$ be the set of neighbors of a node v. A node v selects all colors i for which $x_{v,i} < x_{u,i}$ for all $u \in \Gamma(v)$.

Theorem 4.2. *Choosing $k = 6(\Delta+1)\ln(n)/\varepsilon^2$ leads to a randomized one-shot algorithm that computes a $(1-\varepsilon, k)$-multicoloring w.h.p.*

Proof. For every color i, the n random numbers $x_{v,i}$ for $v \in V$ are all different with probability at least $1 - 1/(kn^2)$. Hence, with probability at least $1 - 1/n^2$, the n random numbers are different for all color $1, \ldots, k$. In the following, we assume that we are in this case.

Consider a node v with neighbors $\Gamma(v)$ where $|\Gamma(v)| = \delta$. For all $i \in [k]$, the probability that v chooses color i is $1/(\delta+1)$. Hence by a Chernoff argument similar to Inequality (2), v chooses a $(1-\varepsilon)/(\delta+1)$-fraction of all colors with probability at least $1 - 1/n^3$. Hence, if the n random numbers $x_{v,i}$ for $v \in V$ are distinct for each color i, every degree δ node chooses a $(1-\varepsilon)/(\delta+1)$-fraction of all colors with probability at least $1 - 1/n^2$. Because the probability that the random numbers are distinct is at least $1 - 1/n^2$, the algorithm fails with probability at most $2/n^2$. □

Algorithm 1 Explicit Deterministic Multicoloring Algorithm: Basic Construction

Input: one-hop view (x, Γ), parameter $\ell \geq 0$
Output: set S of colors, initially $S = \emptyset$

```
1: for all (α_0, α_1, ..., α_ℓ) ∈ F_{q_0} × F_{q_1} × ··· × F_{q_ℓ} do
2:     β_{0,x} := φ_{0,x}(α_0); ∀y ∈ Γ : β_{0,y} := φ_{0,y}(α_0)
3:     for i := 1 to ℓ do
4:         β_{i,x} := φ_{i,β_{i-1,x}}(α_i); ∀y ∈ Γ : β_{i,y} := φ_{i,β_{i-1,y}}(α_i)
5: if ∀y ∈ Γ : β_{ℓ,x} ≠ β_{ℓ,y} then
6:     S := S ∪ (α_0, α_1, ..., α_ℓ, β_{ℓ,x})
```

Remark: In the above algorithm, every node has to generate $\mathcal{O}(\Delta \log^2(n)/\varepsilon^2)$ random bits and send these bits to the neighbors. Using a (non-trivial) probabilistic argument, it is possible to show that the same result can be achieved using only $\mathcal{O}(\log n)$ random bits per node.

4.3 Explicit Algorithms

We have shown in Section 4.1 that there is a one-shot algorithm that almost matches the lower bound that will be proven in Section 5.1. Unfortunately, the techniques of Section 4.1 do not yield an actual algorithm. In this section, we will present constructive methods to obtain a one-shot multicoloring algorithm.

We develop the algorithm in two steps. First, we construct a multicoloring where in the worst case, every node v obtains the same fraction of colors independent of v's degree. We then show how to increase the fraction of colors assigned to low-degree nodes. For an integer parameter $\ell \geq 0$, let $q_0, \ldots, q_\ell$ be prime powers and let $d_0, \ldots, d_\ell$ be positive integers such that $q_0^{d_0+1} \geq N$ and $q_i^{d_i+1} \geq q_{i-1}$ for $i \geq 1$. For a prime power q and a positive integer d, let $\mathcal{P}(q, d)$ be the set of all q^{d+1} polynomials of degree at most d in $\mathbb{F}_q[z]$, where $\mathbb{F}_q$ is the finite field of order q. We assume that that we are given an injection φ_0 from the ID set $[N]$ to the polynomials in $\mathcal{P}(q_0, d_0)$ and injections φ_i from $\mathbb{F}_{q_{i-1}}$ to $\mathcal{P}(q_i, d_i)$ for $i \geq 1$. For a value x in the respective domain, let $\varphi_{i,x}$ be the polynomial assigned to x by injection φ_i. The first part of the algorithm is an adaptation of a technique used in a coloring algorithm described in [16] that is based on an algebraic construction of [7]. There, a node v with one-hop view (x, Γ) selects a color $\big(\alpha, \varphi_{0,x}(\alpha)\big)$, where $\alpha \in \mathbb{F}_{q_0}$ is a value for which $\varphi_{0,x}(\alpha) \neq \varphi_{0,y}(\alpha)$ for all $y \in \Gamma$ (we have to set q_0 and d_0 such that this is always possible). We make two modifications to this basic algorithm. Instead of only selecting one value $\alpha \in \mathbb{F}_{q_0}$ such that $\forall y \in \Gamma : \varphi_{0,x}(\alpha) \neq \varphi_{0,y}(\alpha)$, we select all values α for which this is true. We then use these values recursively (as if $\varphi_{i,x}(\alpha_i)$ was the ID of v) ℓ times to reduce the dependence of the approximation ratio of the coloring on N. The details of the first step of the algorithm are given by Algorithm 1.

Lemma 4.3. *Assume that for $0 \leq i \leq \ell$, $q_i \geq f_i \Delta d_i$ where $f_i > 1$. Then, Algorithm 1 constructs a multicoloring with $q_\ell \cdot \prod_{i=0}^{\ell} q_i$ colors where every node at least receives a λ/q_ℓ-fraction of all colors where $\lambda = \prod_{i=0}^{\ell}(1 - 1/f_i)$.*

Proof. All colors that are added to the color set in line 6 are from $\mathbb{F}_{q_0} \times \mathbb{F}_{q_1} \times \cdots \times \mathbb{F}_{q_\ell} \times \mathbb{F}_{q_\ell}$. It is therefore clear that the number of different colors is $q_\ell \cdot \prod_{i=0}^{\ell} q_i$ as claimed. From the condition in line 5, it also follows that the color sets of adjacent nodes are disjoint.

To determine the approximation ratio, we count the number of colors, a node v with one-hop view (x, Γ) gets. First note that the condition in line 5 of the algorithm implies that (and is therefore equivalent to demand that) $\beta_{i,x} \neq \beta_{i,y}$ for all $y \in \Gamma$ and for all $i \in \{0, \ldots, \ell\}$ because $\beta_{i,x} = \beta_{i,y}$ implies $\beta_{j,x} = \beta_{j,y}$ for all $j \geq i$. We therefore need to count the number of $(\alpha_0, \ldots, \alpha_\ell) \in \mathbb{F}_{q_0} \times \cdots \times \mathbb{F}_{q_\ell}$ for which

$\beta_{i,x} \neq \beta_{i,y}$ for all $i \in \{0, \ldots, \ell\}$ and all $y \in \Gamma$. We prove by induction on i that for $i < \ell$, there are at least $\prod_{j=0}^{i} q_j \cdot (1 - 1/f_j)$ tuples $(\alpha_0, \ldots, \alpha_i) \in \mathbb{F}_{q_0} \times \cdots \mathbb{F}_{q_i}$ with $\beta_{j,x} \neq \beta_{j,y}$ for all $j \leq i$. Let us first prove the statement for $i = 0$. Because the IDs of adjacent nodes are different, we know that $\varphi_{0,x} \neq \varphi_{0,y}$ for all $y \in \Gamma$. Two different degree d_0 polynomials can be equal at at most d_0 values. Hence, for every $y \in \Gamma$, $\varphi_{0,x}(\alpha) = \varphi_{0,y}(\alpha)$ for at most d_0 values α. Thus, since $|\Gamma| \leq \Delta$, there are at least $q_0 - \Delta d_0 \geq q_0 \cdot (1 - 1/f_0)$ values α for which $\varphi_{0,x} \neq \varphi_{0,y}$ for all $y \in \Gamma$. This establishes the statement for $i = 0$. For $i > 0$, the argument is analogous. Let $(\alpha_0, \ldots, \alpha_{i-1}) \in \mathbb{F}_{q_0} \times \cdots \times \mathbb{F}_{q_{i-1}}$ be such that $\beta_{j,x} \neq \beta_{j,y}$ for all $y \in \Gamma$ and all $j < i$. Because $\beta_{i-1,x} \neq \beta_{i-1,y}$, we have $\varphi_{i,x} \neq \varphi_{i,y}$. Thus, with the same argument as for $i = 0$, there are at least $q_i \cdot (1 - 1/f_i)$ values α_i such that $\beta_{i,x} \neq \beta_{i,y}$ for all $y \in \Gamma$. Therefore, the number of colors in the color set of every node is at least $\prod_{i=0}^{\ell} q_i \cdot \big(1 - 1/f_i\big) = \lambda \cdot \prod_{i=0}^{\ell} q_i$. This is a (λ/q_ℓ)-fraction of all colors. □

The next lemma specifies how the values of q_i, d_i, and f_i can be chosen to obtain an efficient algorithm.

Lemma 4.4. *Let ℓ be such that $\ln^{(\ell)} N > \max\{e, \Delta\}$. For $0 \leq i \leq \ell$, we can then choose q_i, d_i, and f_i such that Algorithm 1 computes a multicoloring with $\mathcal{O}(\ell\Delta)^{\ell+2} \cdot \log_\Delta N \cdot \log_\Delta \ln^{(\ell)} N$ colors and such that every node gets at least a $1/\big(4e^{9/4}\Delta\lceil \log_\Delta \ln^{(\ell)} N\rceil\big)$-fraction of all colors.*

Proof. We choose the degrees d_i as

$$d_i = \left\lceil \frac{e \ln^{(i+1)} N}{(e-1)\ln^{(i+2)} N} \right\rceil \text{ for } i < \ell \quad \text{and} \quad d_\ell = \left\lceil \ln_\Delta \ln^{(\ell)} N \right\rceil .$$

For $0 \leq i \leq \ell$, the parameters q_i, and f_i are then chosen as follows:

$$f_i = \min\left\{\ell + 1, 2^{\ell+1-i}\right\} \quad \text{and} \quad f_i \Delta d_i \leq q_i \leq 2 f_i \Delta d_i.$$

Note that for $0 \leq i \leq \ell$, we have $d_i \geq 2$ and $f_i \geq 2$. W.l.o.g., we can certainly also assume that $\Delta \geq 2$. Further note that it is always possible to choose a prime power q_i in the given range. Choosing the values in this way also guarantees that the conditions $q_i \geq f_i \Delta d_i$ and $f_i > 1$ required in Lemma 4.3 are satisfied. Because there must be injective functions from $[N]$ to $\mathcal{P}(q_0, d_0)$ and from $\mathbb{F}_{q_{i-1}}$ to $\mathcal{P}(q_i, d_i)$ for $i \geq 1$, we need to check that $q_0^{d_0+1} \geq N$ and that $q_i^{d_i+1} \geq q_{i-1}$ for $i \geq 1$. We have

$$\begin{aligned}
(d_0 + 1)\ln q_0 &\geq \left(\frac{e \ln N}{(e-1)\ln\ln N} + 1\right) \cdot (\ln\ln N - \ln\ln\ln N) > \ln N \quad (\text{if } \ell > 1) \quad \text{and} \\
(d_\ell + 1)\ln q_\ell &\geq \left(\left\lceil \frac{\ln^{(\ell+1)} N}{\ln \Delta} \right\rceil + 1\right) \cdot (\ln \Delta + \ln f_\ell + \ln d_\ell) \geq \ln^{(\ell+1)} N + \ln\Delta + 2\ln f_\ell + 2 d_\ell \\
&\geq \ln^{(\ell+1)} N + \ln \Delta + \ln f_{\ell-1} + 2\ln 2 \ > \ \ln(2\Delta f_{\ell-1} d_{\ell-1}) \ \geq \ \ln q_{\ell-1}.
\end{aligned}$$

Note that for $1 \leq i \leq \ell$ we have

$$\begin{aligned}
\ln d_{i-1} &< \ln\left(\frac{e\ln^{(i)} N}{(e-1)\ln^{(i+1)} N} + 1\right) \ < \ \ln^{(i+1)} N - \ln^{(i+2)} N + \ln\frac{e}{e-1} + \frac{(e-1)\ln^{(i+1)} N}{e \ln^{(i)} N} \\
&< \ln^{(i+1)} N + \ln\frac{e}{e-1} + \frac{e-1}{e^2} \ < \ \ln^{(i+1)} N + \ln 2.
\end{aligned}$$

because we assume that $\ln^{(\ell)} N > \max\{e, \Delta\}$. We thus have $q_0^{d_0+1} \geq N$ and $q_\ell^{d_\ell+1} \geq q_{\ell-1}$. For the other cases ($0 < i < \ell$), we obtain

$$\begin{aligned}
(d_i + 1)\ln q_i &\geq \left(\left\lceil \frac{e\ln^{(i+1)} N}{(e-1)\ln^{(i+2)} N} \right\rceil + 1\right) \cdot (\ln\Delta + \ln f_i + \ln d_i) \\
&> \left\lceil \frac{e^2}{e-1} + 1 \right\rceil \cdot (\ln\Delta + \ln f_i) + \frac{e\ln^{(i+1)} N}{e-1} \cdot \left(1 - \frac{\ln^{(i+3)} N}{\ln^{(i+2)} N}\right) + \ln\frac{\ln^{(i+1)} N}{\ln^{(i+2)} N} \\
&> 6(\ln\Delta + \ln f_i) + \left(\frac{e}{e-1} \cdot \frac{e-1}{e}\right) \cdot \ln^{(i+1)} N \\
&> \ln\Delta + \ln f_{i-1} + \ln^{(i+1)} N + 9\ln 2 \;>\; \ln q_{i-1}.
\end{aligned}$$

We again used that $\ln^{(\ell)} N > e$ and $\ln d_{i-1} < \ln^{(i+1)} N + \ln 2$. By Lemma 4.3, the number of colors of the computed coloring is at most

$$q_\ell \prod_{i=0}^{\ell} q_i \;\leq\; (c\ell\Delta)^{\ell+2} \cdot \log_\Delta N \cdot \log_\Delta \ln^{(\ell)} N.$$

To show that every node obtains the right number of colors, we first prove the following inequality:

$$\prod_{i=1}^{\infty} \left(1 - \frac{1}{2^i}\right) \geq e^{-5/4}. \tag{3}$$

The inequality follows because

$$\begin{aligned}
\ln\left(\prod_{i=1}^{\infty}\left(1 - \frac{1}{2^i}\right)\right) &= \sum_{i=1}^{\infty} \ln\left(1 - \frac{1}{2^i}\right) = -\sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \frac{1}{j \cdot (2^i)^j} = -\sum_{j=1}^{\infty}\sum_{i=1}^{\infty} \frac{1}{j \cdot (2^j)^i} \\
&= -\sum_{j=1}^{\infty} \frac{1}{j \cdot (2^j - 1)} \;\geq\; -1 - \frac{1}{6} - \frac{1}{21} - \sum_{j=4}^{\infty} \frac{4}{15 \cdot 2^j} \;=\; -\frac{131}{105} \;>\; -\frac{5}{4}.
\end{aligned}$$

Using Inequality (3) and $(1 - 1/n)^{n-1} > 1/e$ for $n \geq 2$, we can now bound the factor λ of Lemma 4.3 as

$$\lambda = \prod_{i=0}^{\ell}\left(1 - \frac{1}{f_i}\right) = \prod_{i=0}^{\ell}\left(1 - \frac{1}{\min\left\{\ell+2, 2^{\ell+1-i}\right\}}\right) > \left(1 - \frac{1}{\ell+1}\right)^{\ell} \cdot \prod_{i=1}^{\infty}\left(1 - \frac{1}{2^i}\right) > e^{-9/4}.$$

The lemma now follows by Lemma 4.3 because $q_\ell \leq 4\Delta\left\lceil 1 + \log_\Delta \ln^{(\ell)} N \right\rceil$. □

The number of colors that Algorithm 1 assigns to nodes with degree almost Δ is close to optimal even for small values of ℓ. If we choose $\ell = \Theta(\log^* N - \log^* \Delta)$, nodes of degree $\Theta(\Delta)$ even receive at least a (d/Δ)-fraction of all colors for some constant d. Because the number of colors assigned to a node v is independent of v's degree, however, the coloring of Algorithm 1 is far from optimal for low-degree nodes. In the following, we show how to improve the algorithm in this respect.

Let $\mathcal{A}_{\Delta,N}$ be an instance of Algorithm 1 for nodes with degree at most Δ and let $\mathcal{C}_{\Delta,N}$ be the color set of $\mathcal{A}_{\Delta,N}$. Further, for a one-hop view (x, Γ), let $\mathcal{C}_{\Delta,N}[x, \Gamma]$ be the colors assigned to (x, Γ) by Algorithm $\mathcal{A}_{\Delta,N}$. We run instances $\mathcal{A}_{2^i,N}$ for all $i \in \left[\lceil \log\Delta \rceil\right]$. A node v with degree δ chooses the colors of all

Algorithm 2 Explicit Deterministic Multicoloring Algorithm: Small Number of Colors

Input: one-hop view (x, Γ), instances $\mathcal{A}_{2^i,N}$ for $i \in \big[\lceil \log \Delta \rceil\big]$ of Algorithm 1, parameter $\varepsilon \in [0,1]$
Output: set S of colors, initially $S = \emptyset$

1: **for all** $i \in \big[\lceil \log \Delta \rceil\big]$ **do**
2: $\quad \omega_i := \Big\lceil \big(\Delta/2^{i-1}\big)^{\varepsilon} \cdot \big|\mathcal{C}_{2^{\lceil \log \Delta \rceil},N}\big| / \big|\mathcal{C}_{2^i,N}\big| \Big\rceil$
3: **for all** $i \in \big\{\lceil \log |\Gamma| \rceil, \dots, \lceil \log \Delta \rceil\big\}$ **do**
4: $\quad$ **for all** $c \in \mathcal{C}_{2^i,N}[x, \Gamma]$ **do**
5: $\quad\quad$ **for all** $j \in [\omega_i]$ **do** $S := S \cup (c, i, j)$

instances for which $2^i \geq \delta$. In order to achieve the desired trade-offs, we introduce an integer weight ω for each color c, i.e., instead of adding color c, we add colors $(1, c), \dots, (\omega, c)$. The details are given by Algorithm 2. The properties of Algorithm 2 are summarized by the next theorem. The straight-forward proof is omitted.

Theorem 4.5. *Assume that in the instances of Algorithm 1, the parameter ℓ is chosen such that for all Δ, $\mathcal{A}_{\Delta,N}$ assigns at least a $f(N)/\Delta$-fraction of the colors to every node. Then, for a parameter $\varepsilon \in [0,1]$, Algorithm 2 computes a $\big(\Omega(f(N)\varepsilon/\delta^{\varepsilon}), \mathcal{O}(|\mathcal{C}_{2\Delta,N}| \cdot \Delta^{\varepsilon}/\varepsilon)\big)$-multicoloring.*

Proof. Let $C_i = \omega_i \cdot \big|\mathcal{C}_{2^i,N}\big|$ be the number of colors arising from colors computed by Algorithm $\mathcal{A}_{2^i,N}$ and let $C = \sum_i C_i$ be the total number of colors used. We have $C_i = \mathcal{O}(C_1 \cdot 2^{-\varepsilon i})$ and thus get $C = \mathcal{O}(C_1 \cdot 2^{\varepsilon}/(2^{\varepsilon} - 1)) = \mathcal{O}(|\mathcal{C}_{2\Delta,N}| \cdot \Delta^{\varepsilon}/\varepsilon)$. Consider a node v of degree δ and let $k = \lceil \log \delta \rceil$. The number of colors assigned to v is at least

$$\frac{f(N) C_k}{2\delta} \;=\; \Omega\left(\frac{f(N)|\mathcal{C}_{2\Delta,N}|\Delta^{\varepsilon}}{\delta^{1+\varepsilon}}\right) \;=\; \Omega\left(\frac{f(N)\varepsilon}{\delta^{1+\varepsilon}}\right) \cdot C.$$

This concludes the proof. □

Corollary 4.6. *Let $\varepsilon \in [0,1]$ and let $\ell \geq 0$ be a fixed constant in all used instances of Algorithm 1. Then, Algorithm 2 computes an $\big(\varepsilon/\mathcal{O}(\delta^{\varepsilon} \log_\Delta \ln^{(\ell)} N), \mathcal{O}(\Delta^{\ell+2} \cdot \log_\Delta N \cdot \log_\Delta \ln^{(\ell)} N)\big)$-multicoloring. In particular, choosing $\ell = 0$ leads to an $\big(\varepsilon/\mathcal{O}(\delta^{\varepsilon} \log_\Delta N), \mathcal{O}(\Delta^2 \log_\Delta^2 N)\big)$-multicoloring. Taking the maximum possible value for ℓ in all used instances of Algorithm 1 yields an $\big(\varepsilon/\mathcal{O}(\delta^{\varepsilon}), \Delta^{\mathcal{O}(\log^* N - \log^* \Delta)} \cdot \log_\Delta N\big)$-multicoloring.*

Remark: Essentially the same results can also be achieved by a different method. Instead of interpreting an ID x as a polynomial φ_x and choose colors $(\alpha, \varphi_x(\alpha))$ for values of α for which $\varphi_x(\alpha) \neq \varphi_y(\alpha)$ for all neighboring IDs, we can choose colors as follows. Let p_i be the i^{th} prime number. We choose colors $(i, x \bmod p_i)$ for prime numbers p_i for which $x \not\equiv y \pmod{p_i}$. Based on this basic coloring method, one can do the same recursive construction as in Algorithm 1. The prime number theorem and the Chinese remainder theorem guarantee that all nodes can choose sufficiently many small colors.

5 Lower Bounds

In this section, we give lower bounds on the number of colors required for one-shot multicoloring algorithms. In fact, we even derive the lower bounds for algorithms that need to assign only one color to every node, i.e., the results even hold for standard coloring algorithms.

It has been shown in [15] that every deterministic one-shot c-coloring algorithm $\mathcal{A}$ can be interpreted as a set of c antisymmetric relations on the ID set $[N]$. Assume that $\mathcal{A}$ assigns a color from a set C with $|C| = c$ to every one-hop view (x, Γ). For every color $\alpha \in C$, there is a relation $\lhd_\alpha$ such that for all $x, y \in [N]$ $x \ntriangleleft_\alpha y \vee y \ntriangleleft_\alpha x$. Algorithm $\mathcal{A}$ can assign color $\alpha \in C$ to a one-hop view (x, Γ) iff $\forall y \in \Gamma : x \lhd_\alpha y$.

For $\alpha \in C$, let $\mathrm{Bad}_\alpha(x) := \{y \in [N] : x \ntriangleleft_\alpha y\}$ be the set of IDs that must not be adjacent to an α-colored node with ID x. To show that there is no deterministic, one-shot c-coloring algorithm, we need to show that for every c antisymmetric relations $\lhd_{\alpha_1}, \dots, \lhd_{\alpha_c}$ on $[N]$, there is a one-hop view (x, Γ) such that $\forall i \in [c] : \Gamma \cap \mathrm{Bad}_{\alpha_i}(x) \neq \emptyset$. Lemma 5.1 is proven in [15].

Lemma 5.1. [15] *For all ID sets $X \subseteq [N]$ and all colors $\alpha \in C$, we have $\sum_{x \in X} \left|\mathrm{Bad}_\alpha(x) \cap X\right| \geq \binom{|X|}{2}$.*

The following lemma is a generalization of Lemma 4.5 in [15] and key for the deterministic and the randomized lower bounds of this paper.

Lemma 5.2. *Let $X \subseteq [N]$ be a set of IDs and let $t_1, \dots, t_\ell$ and $k_1, \dots, k_\ell$ be positive integers such that*

$$t_i \cdot \big(\lambda(|X| - c)t_i - c\big) > 2c(k_i - 1) \quad \textit{for } 1 \leq i \leq \ell \textit{ and a parameter } \lambda \in [0, 1].$$

Then there exists an ID set $X' \subseteq X$ with $|X'| > (1 - \ell \cdot \lambda) \cdot (|X| - c)$ such that for all $i \in [\ell]$,

$$\forall x \in X', \forall \alpha_1, \dots, \alpha_{t_i} \in C : \sum_{j=1}^{t_i} \left|\mathrm{Bad}_{\alpha_j}(x) \cap X\right| \geq k_i \quad \textit{and} \quad \forall x \in X', \forall \alpha \in C : \mathrm{Bad}_\alpha(x) \cap X \neq \emptyset.$$

Proof. Let $Y = \{x \in X : \forall \alpha \in C : \mathrm{Bad}_\alpha(x) \cap X \neq \emptyset\}$. Lemma 5.1 implies that for any two colors $x, y \in X$ and every color $\alpha \in C$, $\left|(\mathrm{Bad}_\alpha(x) \cup \mathrm{Bad}_\alpha(y)) \cap X\right| \geq 1$. Hence, for every color $\alpha \in C$, there is at most one ID $x \in X$ for which $\mathrm{Bad}_\alpha(x) \cap X = \emptyset$. We thus have $|Y| \geq |X| - c$ and we will show that we can choose $X' \subseteq Y$ with $|X'| > (1 - \ell \cdot \lambda) \cdot |Y|$. Let t and k be two positive integers such that $t\big(\lambda(|X| - c) - c\big) > 2c(k - 1)$. We show that then there is a subset $X'' \subseteq Y$ with $|X''| > (1 - \lambda) \cdot |Y|$ such that

$$\forall x \in X'', \forall \alpha_1, \dots \alpha_t \in C : \sum_{j=1}^{t} \left|\mathrm{Bad}_{\alpha_j}(x) \cap X\right| \geq k. \tag{4}$$

This implies that for every $i \in [\ell]$, there is a set $X''_i \subseteq Y$ with $|X''_i| > (1 - \lambda) \cdot |Y|$ such that Inequality (4) holds for $t = t_i$ and $k = k_i$. Because $|X''_1 \cap \ldots \cap X''_\ell| > (1 - \ell \cdot \lambda) \cdot |Y|$, we can therefore choose $X' = X''_1 \cap \ldots \cap X''_\ell$.

For the sake of contradiction, assume that there is no set X'' for which Inequality (4) holds. This implies that there exists a set $Z \subseteq Y$ of size $|Z| \geq \lambda|Y|$ such that

$$\forall x \in Z, \exists \alpha_1, \dots, \alpha_t : \sum_{i=1}^{t} \left|\mathrm{Bad}_{\alpha_i}(x) \cap X\right| \leq k - 1. \tag{5}$$

For $x \in Z$, let $\alpha_{x,1}, \dots, \alpha_{x,t}$ be the t colors that minimize the sum $\sum_{i=1}^{t} \left|\mathrm{Bad}_{\alpha_{x,i}}(x) \cap X\right|$. Condition (5) then implies that

$$\sum_{x \in Z} \sum_{i=1}^{t} \left|\mathrm{Bad}_{\alpha_{x,i}}(x) \cap X\right| \leq |Z| \cdot (k - 1). \tag{6}$$

For a color $\alpha \in C$, let $\#_\alpha = \big|\{(x,i) \in Z \times [t] : \alpha_{x,i} = \alpha\}\big|$. We have $\sum_{\alpha \in C} \#_\alpha = |Z|t$. Applying Lemma 5.1. we thus get

$$\sum_{x\in Z}\sum_{i=1}^{t} \big|\mathrm{Bad}_{\alpha_{x,i}}(x) \cap X\big| \geq \sum_{\alpha\in C}\binom{\#_\alpha}{2} \geq c \cdot \frac{\frac{|Z|t}{c}\cdot\left(\frac{|Z|t}{c}-1\right)}{2} = \frac{|Z|t\cdot(|Z|t-c)}{2c}.$$

Together with Inequality (6) and since we have $|Z| \geq \lambda|Y| \geq \lambda\cdot(|X|-c)$, we therefore get $2c(k-1) \geq t(|Z|t-c) \geq t\cdot\big(\lambda(|X|-c)t-c\big)$. This is a contradiction to the assumption that $t\cdot\big(\lambda(|X|-c)t-c\big) > 2c(k-1)$ and thus concludes the proof. $\square$

5.1 Deterministic Lower Bound

Assume that we are given a deterministic one-shot coloring algorithm $\mathcal{A}$ that assigns colors from a color set C to one-hop views (x,Γ) with $x \in [N]$ and $\Gamma \subset [N]$. We assume that $N = 4\delta^2 \cdot h$ and that $c = \delta^2$ for a parameter $\delta = 2^h$ where h is a positive integer. We will show that for these values of N and c, there is a one-hop view (x,Γ) with $|\Gamma| = \Theta(\delta)$ to which none of the colors in C can be assigned. This then implies the desired $\Omega(\Delta^2)$ lower bound.

Recall that we have to show that there is a one-hop view (x,Γ) such that $\forall\alpha \in C : \Gamma \cap \mathrm{Bad}_\alpha(x) \neq \emptyset$. The next lemma shows that for almost $N/2$ values $x \in [N]$, there is a one-hop view (x,Γ) such that $\Gamma \cap \mathrm{Bad}_\alpha(x) \neq \emptyset$ for most colors $\alpha \in C$.

Lemma 5.3. *There is a set $X \subseteq [N]$ of size $|X| \geq \delta^2(2h-1)$ such that for all $x \in X$, there is a set $\Gamma \subseteq [N] \setminus \{x\}$ of size $|\Gamma| = O(\delta)$ such that $\Gamma \cap \mathrm{Bad}_\alpha(x) \neq \emptyset$ for all but at most $h \cdot \delta$ colors $\alpha \in C$.*

Proof. Recall that $N = 4h\delta^2$ and $c = \delta^2$. Let k be an integer that satisfies $h\delta((N-c)h\delta/2-c) > 2c(k-1)$. By Lemma 5.2, there is a set $X \subseteq [N]$ of size $|X| \geq \delta^2(2h-1)$ such that

$$\forall x \in X, \forall \alpha_1,\dots,\alpha_{h\delta} \in C : \sum_{i=1}^{h\delta} \big|\mathrm{Bad}_{\alpha_i}(x)\big| \geq k.$$

Hence, for all $x \in X$, for all but $h\delta$ colors $\alpha \in C$, we have $\big|\mathrm{Bad}_\alpha(x)\big| \geq k/(h\delta)$. We can choose

$$k \;\geq\; \frac{h\delta\big((N-c)h\delta/2-c\big)}{2c} \;=\; h\delta\cdot\left(\frac{(4h-1)h\delta}{4}-\frac{1}{2}\right) \;\geq\; h\delta\cdot\frac{h^2\delta}{2}.$$

For the last inequality, we use $h \geq 1$ and $\delta = 2^h \geq 2$. For an ID $x \in X$, let $C'_x \subseteq C$ be the set of colors α for which $|\mathrm{Bad}_\alpha(x)| \geq h^2\delta/2$. By the above observation, we have $|[N] \setminus C'_x| \leq h\delta$. To prove the lemma, we show that for every $x \in X$, there is a set $\Gamma \subseteq [N]\setminus\{x\}$ satisfying $|\Gamma| = O(\delta)$ such that for all $\alpha \in C'_x$, $\Gamma \cap \mathrm{Bad}_\alpha(x) \neq \emptyset$. Consider a specific ID $x \in X$. Finding a smallest possible set Γ that satisfies the requirements can be seen as an instance of a minimum set cover problem. We need to find IDs $y_1, y_2, \dots, y_{|\Gamma|} \in [N]\setminus\{x\}$ such that every color $\alpha \in C'_x$ is 'covered.' A color α is 'covered' if there is an $y_i \in \mathrm{Bad}_\alpha(x)$. Because every color α can be covered by $h^2\delta/2$ different IDs, choosing every ID $y \in [N]\setminus\{x\}$ with weight $2/(h^2\delta)$ gives a fractional solution to this set cover problem. The size of this fractional solution is $2(N-1)/(h^2\delta) \leq 8\delta/h$. Since the integrality gap of set cover is at most logarithmic in the number of elements, our set cover problem has an integer solution of size at most $O(\log(c)8\delta/h) = O(\delta\log\delta/h) = O(\delta)$. This implies that there is a set Γ of size $O(\delta)$ that satisfies our requirements and thus concludes the proof. $\square$

Lemma 5.4. *Let $X \subseteq [N]$ be an ID set of size $f \cdot \delta^2$ for $f > 2$. There is a set $X' \subseteq X$ of size $X' \geq \frac{f-1}{2}\delta^2$ such that for every $\delta \cdot 2^i$ colors $\alpha_1, \dots, \alpha_{\delta 2^i}$ and every $x \in X'$, there is an ID $y \in X \setminus \{x\}$ that occurs in at least $2^{2i}/16$ different sets $\mathrm{Bad}_{\alpha_j}(x)$ for $j \in [\delta 2^i]$.*

Proof. Let k be an integer that satisfies $\delta 2^i((N-c)\delta 2^i/2 - c) > 2c(k-1)$ for $i \geq 1$. By Lemma 5.2, there is a set $X \subseteq [N]$ of size $|X'| \geq (|X| - c)/2 = \frac{f-1}{2}\delta^2$ such that

$$\forall x \in X', \forall \alpha_1, \dots, \alpha_{\delta 2^i} \in C : \sum_{j=1}^{\delta 2^i} \left|\mathrm{Bad}_{\alpha_j}(x) \cap X\right| \geq k. \tag{7}$$

We can choose

$$k \;\geq\; \frac{\delta 2^i\big((N-c)\delta 2^i/2 - c\big)}{2c} \;=\; \frac{\delta^2 2^{2i}(f-1)\delta^2}{4\delta^2} - \frac{\delta 2^i}{2} \;\geq\; 2^{2i} f \delta^2 \cdot \left(\frac{1}{4} - \frac{1}{8} - \frac{1}{16}\right) \;=\; \frac{2^{2i} f \delta^2}{16}.$$

For the second inequality, we use that $f > 2$, $i \geq 1$, and $\delta \geq 2$. Hence, on average, all $y \in X$ occur $k/|X|$ times in the sets $\mathrm{Bad}_{\alpha_j}(x) \cap X$ for all $x \in X'$ and $j \in [\delta 2^i]$. Hence, for every $x \in X'$ and all $\alpha_1, \dots, \alpha_{\delta 2^i}$, there is an ID $y \in X$ that occurs $k/|X| \geq 2^{2i}/16$ times. □

Combining Lemmas 5.3 and 5.4 allows to find a one-hop view (x, Γ) to which none of the c colors can be assigned.

Lemma 5.5. *There is an ID $x \in [N]$ and a set $\Gamma \in [N] \setminus \{x\}$ of size $|\Gamma| = O(\delta)$ such that $\Gamma \cap \mathrm{Bad}_\alpha(x) \neq \emptyset$ for all colors $\alpha \in C$.*

Proof. In Lemma 5.3 it is shown that there is a set $X \subseteq [N]$ of size $|X| \geq \delta^2(2h-1)$ such that for all $x \in X$, there is a set Γ of size $|\Gamma| = O(\delta)$ such that $\Gamma \cap \mathrm{Bad}_\alpha(x) \neq \emptyset$ for all but $h\delta$ colors $\alpha \in C$. It therefore remains to show that there is an $x \in X$ for which we can find $O(\delta)$ additional IDs $y \in [N]$ such that the remaining $h\delta$ colors are also 'covered.'

We construct sets $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_{\lfloor \log h \rfloor} \subseteq X$ of size $|X_i| \geq (h/2^{\lfloor \log h \rfloor - i} - 1)\delta^2$ as follows. By Lemma 5.4, we can choose a set $X_{\lfloor \log h \rfloor} \subseteq X$ of size $\frac{2h-2}{2}\delta^2 = (h-1)\delta^2$ such that for all $\delta \cdot 2^{\lfloor \log h \rfloor}$ colors $\alpha_1, \dots, \alpha_{\delta 2^{\lfloor \log h \rfloor}} \in C$ and every $x \in X_{\lfloor \log h \rfloor}$, there is an ID $y \in X$ that occurs in at least $2^{2\lfloor \log h \rfloor}/16$ different sets $\mathrm{Bad}_{\alpha_j}(x)$. Similarly, for $i < \lfloor \log h \rfloor$, we can choose a set $X_i \subseteq X_{i+1}$ of size $(2^{\lfloor \log h \rfloor - i} - 1)\delta^2$ such that for all $\delta 2^i$ colors $\alpha_1, \dots, \alpha_{\delta 2^i} \in C$ and every $x \in X_i$, there is an ID $y \in X_{i+1}$ that occurs in at least $2^{2i}/16$ different sets $\mathrm{Bad}_{\alpha_j}(x)$. For simplicity, we define $X_{\lfloor \log h \rfloor + 1} = X$. By the above argumentation, as long as we still have to 'cover' at least $\delta 2^i$ colors, we can choose an ID $y \in X_{i+1}$ that 'covers' at least $2^{2i}/16$ colors. The total number of IDs we need until all but 2δ colors are 'covered' therefore is at most

$$\sum_{i=1}^{\lfloor \log h \rfloor} \frac{16\delta^2(h/2^{\lfloor \log h \rfloor - i + 1} - h/2^{\lfloor \log h \rfloor - i})}{\delta 2^{2i}} = \sum_{i=1}^{\lfloor \log h \rfloor} \frac{16\delta h/2^{\lfloor \log h \rfloor}}{2^i} \leq \sum_{i=1}^{\infty} \frac{32\delta}{2^i} \leq 64\delta.$$

Clearly, the remaining 2δ colors can be covered by at most 2δ additional IDs and the claim of the lemma therefore follows. □

Theorem 5.6. *If $N = \Omega(\Delta^2 \log \Delta)$, every deterministic one-shot coloring algorithm needs at least $\Omega(\Delta^2 + \log\log N)$ colors.*

Proof. Choosing δ appropriately, the $\Omega(\Delta^2)$ bound directly follows from Lemma 5.5 and from the fact that the number of colors needed can only grow if N becomes larger. The $\Omega(\log\log N)$ lower bound is proven in [16]. □

5.2 Randomized Lower Bound

To obtain a lower bound for randomized multicoloring algorithms, we can again use the tools derived for the deterministic lower bound by applying Yao's principle. On a worst-case input, the best randomized algorithm cannot perform better than the best deterministic algorithm for a given random input distribution. Choosing the node labeling at random allows to again only consider deterministic algorithms.

We assume that the n nodes are assigned a random permutation of the labels $1, \dots, n$ (i.e., every label occurs exactly once). Note that because we want to prove a lower bound, assuming the most restricted possible ID space makes the bound stronger. For an ID $x \in [n]$, we sort all colors $\alpha \in C$ by increasing values of $|\mathrm{Bad}_\alpha(x)|$ and let $\alpha_{x,i}$ be the i^{th} color in this sorted order. Further, for $x \in [n]$, we define $b_{x,i} := \left|\mathrm{Bad}_{\alpha_{x,i}}(x)\right|$. In the following, we assume that

$$c = \kappa \cdot \frac{\Delta \lfloor \ln n \rfloor}{\lceil \ln \ln n \rceil + 2} \quad \text{and} \quad n \geq 12 \quad \text{and} \quad n \geq \Delta \cdot \ln n \tag{8}$$

for a constant $0 < \kappa \leq 1$ that will be determined later. By applying Lemma 5.2 in different ways, the next lemma gives lower bounds on the values of $b_{x,i}$ for $n/2$ IDs $x \in [n]$.

Lemma 5.7. *Assume that c and n are as given by Equation* (8) *and let $0 < \rho < 1/3$ be a positive constant. Further, let $\tilde{t} = \lceil \rho \ln n / \ln \ln n \rceil$ and $t_i = 2^{i-1} \cdot \lfloor \ln n \rfloor$ for $1 \leq i \leq \ell$ where $\ell = \lceil \ln \ln n \rceil + 2$. Then, for at least $n/2$ of all IDs $x \in [n]$, we have*

$$b_{x,1} \geq \frac{\ln \ln n}{44\kappa \cdot \ln n} \cdot \frac{n}{\Delta} - 1, \quad b_{x,\tilde{t}} \geq \frac{\rho}{48\kappa} \cdot \frac{n}{\Delta} - \frac{1}{2}, \quad \textit{and} \quad b_{x,t_i} \geq 2^{i-1} \cdot \left(\frac{1}{8\kappa} \cdot \frac{n}{\Delta} - \frac{1}{2} \right) \quad \textit{for} \quad 1 \leq i \leq \ell.$$

Proof. We first prove that the bounds on $b_{x,1}$ and on $b_{x,\tilde{t}}$ hold for at least $5/6 \cdot n$ IDs $x \in [n]$. For $t \in [c]$, we use $B_{x,1} = \sum_{i=1}^{t} b_{x,i}$, i.e., $B_{x,i}$ is the total number of IDs in the smallest t sets $\mathrm{Bad}_\alpha(x)$ for $\alpha \in C$. Using $\lambda = 1/22$, $\ell = 2$, and $X = [n]$ in Lemma 5.2 yields that there is a set X' of size $|X'| \geq (1-1/11)(n-c) \geq (1-1/11) \cdot 11/12 \cdot n = 5/6 \cdot n$ such that for all $x \in X'$,

$$b_{x,1} = B_{x,1} \geq \frac{n - 23c}{44c} \quad \text{and} \quad b_{x,\tilde{t}} \geq \frac{B_{x,\tilde{t}}}{\tilde{t}} \geq \frac{(n-c)\tilde{t} - 22c}{44c}.$$

Using the value for c from (8) then gives

$$\begin{aligned} b_{x,1} &\geq \frac{n \cdot \lceil 2 + \ln \ln n \rceil}{44\kappa\Delta \lfloor \ln n \rfloor} - \frac{23}{44} \geq \frac{\ln \ln n}{44\kappa \cdot \ln n} \cdot \frac{n}{\Delta} - 1, \\ b_{x,\tilde{t}} &\geq \frac{11/12 \cdot n \cdot \tilde{t}}{44c} - \frac{1}{2} \geq \frac{n \cdot \lceil 2 + \ln \ln n \rceil \cdot \lceil \rho \ln n / \ln \ln n \rceil}{48\kappa \cdot \Delta \cdot \lfloor \ln n \rfloor} - \frac{1}{2} \geq \frac{\rho}{48\kappa} \cdot \frac{n}{\Delta} - \frac{1}{2}. \end{aligned}$$

To determine the bounds for the values $b_{x,t_1}, \dots, b_{x,t_\ell}$, we again apply Lemma 5.2 and use $X = X'$ and $\lambda = 1/(3\ell)$. This implies that there is an ID set $X'' \subseteq X'$ of size $|X''| \geq (1-1/3)(|X'|-c) \geq 2/3 \cdot 3/4 n = n/2$ such that for all $x \in X''$ and for all $i \in [\ell]$, we have

$$\begin{aligned} b_{x,t_i} &\geq \frac{B_{x,t_i}}{t_i} \geq \frac{2^{i-1} \cdot \lfloor \ln n \rfloor \cdot 5/6 \cdot n - c}{3\ell \cdot 2c} - \frac{1}{2} \geq \frac{2^{i-1} \cdot \lfloor \ln n \rfloor \cdot 3/4 \cdot n}{6\ell c} - \frac{1}{2} \\ &= \frac{2^{i-1} \cdot \lfloor \ln n \rfloor \cdot n \cdot \lceil 2 + \ln \ln n \rceil}{8\kappa \cdot \lceil 2 + \ln \ln n \rceil \cdot \Delta \cdot \lfloor \ln n \rfloor} - \frac{1}{2} \geq 2^{i-1} \cdot \left(\frac{1}{8\kappa} \cdot \frac{n}{\Delta} - \frac{1}{2} \right). \end{aligned}$$

□

In order to prove the lower bound, we want to show that for a randomly chosen one-hop view (x, Γ) with $|\Gamma| = \Delta$, the probability that there is a color $\alpha \in C$ for which $\Gamma \cap \mathrm{Bad}_\alpha(x) = \emptyset$ is sufficiently small. Instead of directly looking at random one-hop views (x, Γ) with $|\Gamma| = \Delta$, we first look at one-hop views with $|\Gamma| \approx \Delta/e$ that are constructed as follows. Let $X \subseteq [n]$ be the set of IDs x of size $|X| \geq n/2$ for which the bounds of Lemma 5.7 hold. We choose x_R uniformly at random from X. The remaining $n-1$ IDs are independently added to a set Γ_R with probability $p = \frac{\Delta}{en}$. For a color $\alpha \in C$, let $\mathcal{E}_\alpha$ be the event that $\Gamma_R \cap \mathrm{Bad}_\alpha(x_R) \neq \emptyset$, i.e., $\mathcal{E}_\alpha$ is the event that color α cannot be assigend to the randomly chosen one-hop view (x_R, Γ_R).

Lemma 5.8. *The probability that the randomly chosen one-hop view cannot be assigned one of the c colors in C is bounded by*

$$\mathbb{P}\left[\bigcap_{\alpha\in C}\mathcal{E}_\alpha\right] \;\geq\; \prod_{\alpha\in C}\mathbb{P}\big[\mathcal{E}_\alpha\big] \;\geq\; \prod_{\alpha\in C}\left(1 - e^{-\frac{\Delta}{en}\cdot|\mathrm{Bad}_\alpha(x_R)|}\right) \;=\; \prod_{i=1}^{c}\left(1 - e^{-\frac{\Delta\cdot b_{x_R,i}}{en}}\right).$$

Proof. Note first that for $\alpha \in C$, we have

$$\mathbb{P}\big[\overline{\mathcal{E}_\alpha}\big] \;=\; \mathbb{P}\big[\Gamma_R \cap \mathrm{Bad}_\alpha(x_R) = \emptyset\big] \;=\; (1-p)^{|\mathrm{Bad}_\alpha(x_R)|} \;\leq\; e^{-p|\mathrm{Bad}_\alpha(x_R)|} \;=\; e^{-\frac{\Delta}{en}\cdot|\mathrm{Bad}_\alpha(x_R)|}.$$

It therefore remains to prove that the probability that all events $\mathcal{E}_\alpha$ occur can be lower bounded by the probability that would result for independent events. Let us denote the colors in C by $\alpha_1, \ldots, \alpha_c$. We then have

$$\mathbb{P}\left[\bigcap_{\alpha\in C}\mathcal{E}_\alpha\right] \;=\; \prod_{i=1}^{c}\mathbb{P}\left[\mathcal{E}_{\alpha_i}\,\middle|\,\bigcap_{j=1}^{i-1}\mathcal{E}_{\alpha_j}\right] \;\geq\; \prod_{i=1}^{c}\mathbb{P}\big[\mathcal{E}_{\alpha_i}\big]. \tag{9}$$

The inequality holds because the events $\mathcal{E}_\alpha$ are positively correlated. Knowing that an element from a set $\mathrm{Bad}_\alpha(x_R)$ is in Γ_R cannot decrease the probability that an element from a set $\mathrm{Bad}_{\alpha'}(x_R)$ is in Γ_R. Note that this is only true because the IDs are independently added to Γ_R. More formally, Inequality (9) can also directly be followed from the FKG inequality [8]. $\square$

For space reasons, the following technical lemma is given without proof.

Lemma 5.9. *Assume that c and n are given as in* (8) *where the constant κ is chosen sufficiently small and let $\rho > 0$ be a constant as in Lemma 5.7. There is a constant $n_0 > 0$ such that for $n \geq n_0$, $\mathbb{P}\left[\bigcap_{\alpha\in C}\mathcal{E}_\alpha\right] > \frac{1}{2n^{3\rho}}$.*

Proof. Let $\hat{b}_{x,1} = \ln\ln n/(44\kappa\ln n)\cdot n/\Delta - 1$, $\hat{b}_{x,\tilde{t}} = \rho/(48\kappa)\cdot n/\Delta - 1/2$, and $\hat{b}_{x,t_i} = 2^{i-1}\cdot\big(n/(8\kappa\Delta) - 1/2\big)$ for $i \in [\ell]$ be the bounds from Lemma 5.7 on $b_{x,1}$, b_x, $\tilde{t}$, and b_{x,t_i}, respectively. Further, let

$$q(b) := 1 - e^{-\frac{\Delta\cdot b}{en}}.$$

We then have

$$\prod_{i=1}^{c}\left(1 - e^{-\frac{\Delta\cdot b_{x_R,i}}{en}}\right) = \prod_{i=1}^{c} q(b_{x_R,i}) \geq q(\hat{b}_{x,1})^{\tilde{t}-1}\cdot q(\hat{b}_{x,\tilde{t}})^{t_1-\tilde{t}}\cdot\prod_{i=1}^{\ell-1} q(\hat{b}_{x,t_i})^{t_{i+1}-t_i}\cdot q(\hat{b}_{x,t_\ell})^{n-t_\ell}. \tag{10}$$

To obtain useful bounds on $q(b)$, we need the following two inequalities for a parameter $s > 0$.

$$1 - e^{-s} \geq s - \frac{s^2}{2} \geq \frac{s}{2} \;\text{ if } s \leq 1 \quad\text{and}\quad 1 - e^{-s} \geq 1 - 2e^{-s} + 2e^{-2s} \geq e^{-2e^{-s}} \;\text{ if } s \geq \ln 2. \tag{11}$$

We bound the 4 factors in Inequality (10) individually. If $n \geq n_0$ for a sufficiently large constant n_0, we clearly have

$$\frac{\Delta \cdot \hat{b}_{x,1}}{en} \;=\; \frac{\ln\ln n}{44e\kappa \ln n} - \frac{\Delta}{en} \;\leq\; 1.$$

By Inequality (11), we therefore get

$$q(\hat{b}_{x,1})^{\tilde{t}-1} \geq \left(\frac{\ln\ln n}{88e\kappa\ln n} - \frac{\Delta}{en}\right)^{\frac{\rho\ln n}{\ln\ln n}} = \left(\frac{f \cdot \ln\ln n}{\ln n}\right)^{\frac{\rho\ln n}{\ln\ln n}} = e^{-\rho\ln n(1-o(1))} \geq \frac{1}{n^\rho}. \tag{12}$$

for a positive constant f and $n \geq n_0$ for a sufficiently large constant n_0. Recall that we assume that $n \geq \Delta \ln n$. If we choose κ small enough, we have

$$\frac{\Delta \cdot \hat{b}_{x,\tilde{t}}}{en} = \frac{\rho}{48e\kappa} - \frac{\Delta}{2en} \geq \ln 2.$$

For the second factor in Inequality (10), for a sufficiently small constant κ, we therefore obtain

$$q(\hat{b}_{x,\tilde{t}})^{t_1-\tilde{t}} \geq e^{-2(\ln n - \tilde{t})\; e^{-\frac{\rho}{48e\kappa}-\frac{\Delta}{2en}}} \geq e^{-\rho\ln n} = \frac{1}{n^\rho}. \tag{13}$$

Note that we have to choose κ to be at most $\kappa \approx \rho/(48e\log(2/\rho))$. To bound the third factor in (10), we use that $\sum_{i=0}^{\infty} 2^i \cdot e^{-2^i} < 1/e + 2\cdot 2/e^2 < 1$. Applying Inequality (11), we then get

$$\prod_{i=1}^{\ell-1} q(\hat{b}_{x,t_i})^{t_{i+1}-t_i} \geq \prod_{i=1}^{\ell-1} e^{-2(t_{i+1}-t_i)e^{-2^{i-1}\cdot\left(\frac{1}{8e\kappa}-\frac{\Delta}{2en}\right)}} \geq e^{-\rho\sum_{i=0}^{\ell-2} 2^i\cdot e^{-2^i}} > \frac{1}{n^\rho} \tag{14}$$

if the constant κ is chosen sufficiently small. For the last part, we finally have (again for a sufficiently small constant κ)

$$q(\hat{b}_{x,t_\ell})^{n-t_\ell} \geq e^{-2n\cdot e^{-2\ln n\cdot\left(\frac{1}{8e\kappa}-\frac{\Delta}{2en}\right)}} \geq \frac{1}{2}. \tag{15}$$

The lemma now follows by combining Lemma 5.8, Inequality (10), and Inequalities (12)–(15). ☐

Lemma 5.10. *Let (x, Γ) be a one-hop view chosen uniformly at random from all one-hop views with $|\Gamma| = \Delta$. If $\Delta \geq e(\ln n + 2)$ and n, c, and ρ are as before, the probability that none of the c colors can be assigned to (x, Γ) is at least $1/(8n^{3\rho})$.*

Proof. We can choose the random one-hop view (x, Γ) as follows. We choose x at random from $[n]$. We add every other ID independently with probability $p = \Delta/(en)$ to a set Γ'. If $|\Gamma'| = \Delta$, we set $\Gamma = \Gamma'$. If $|\Gamma| < \Delta$, we add $\Delta - |\Gamma'|$ additional random IDs to Γ' to obtain Γ, whereas if $|\Gamma'| > \Delta$, we remove $|\Gamma'| - \Delta$ random IDs from Γ' and get Γ. Let $\mathcal{E}_x$ be the event that we choose x from the at least $n/2$ IDs for which the bounds of Lemma 5.7 hold. Further, let $\mathcal{E}_\Gamma$ be the event that $|\Gamma'| \leq \Delta$ and let $\mathcal{E}$ be the event that $\Gamma \cap \mathrm{Bad}_\alpha(x) \neq \emptyset$ for all $\alpha \in C$. As before, $\mathcal{E}_\alpha$ is the event that $\Gamma' \cap \mathrm{Bad}_\alpha(x) \neq \emptyset$. In order to prove the lemma, we need to have a lower bound on the probability $\mathbb{P}[\mathcal{E}]$. We have

$$\begin{aligned}
\mathbb{P}[\mathcal{E}] \;&\geq\; \mathbb{P}\left[\mathcal{E}_x \cap \mathcal{E}_\Gamma \cap \bigcap_{\alpha\in C}\mathcal{E}_\alpha\right] = \mathbb{P}[\mathcal{E}_x]\cdot\mathbb{P}\left[\mathcal{E}_\Gamma \cap \bigcap_{\alpha\in C}\mathcal{E}_\alpha \,\middle|\, \mathcal{E}_x\right] = \mathbb{P}[\mathcal{E}_x]\cdot\left(1 - \mathbb{P}\left[\overline{\mathcal{E}_\Gamma} \cup \overline{\bigcap_{\alpha\in C}\mathcal{E}_\alpha} \,\middle|\, \mathcal{E}_x\right]\right)\\
&\geq\; \mathbb{P}[\mathcal{E}_x]\cdot\left(1 - \mathbb{P}[\overline{\mathcal{E}_\Gamma}|\mathcal{E}_x] - \mathbb{P}\left[\overline{\bigcap_{\alpha\in C}\mathcal{E}_\alpha}\,\middle|\,\mathcal{E}_x\right]\right) = \mathbb{P}[\mathcal{E}_x]\cdot\left(\mathbb{P}\left[\bigcap_{\alpha\in C}\mathcal{E}_\alpha\,\middle|\,\mathcal{E}_x\right] - \mathbb{P}[\overline{\mathcal{E}_\Gamma}]\right)\\
&\geq\; \frac{1}{2}\cdot\left(\frac{1}{2n^{3\rho}} - \frac{1}{4n^{3\rho}}\right) \;\geq\; \frac{1}{8n^{3\rho}}.
\end{aligned}$$

The first inequality on the last line follows from Lemma 5.9 and because by applying a Chernoff bound and using that $\Delta \geq e(\ln n + 2)$, we have

$$\mathbb{P}\big[\overline{\mathcal{E}_\Gamma}\big] = \mathbb{P}\big[\Gamma' > \Delta\big] < e^{-\Delta/e} \leq e^{-\ln n - 2} = \frac{1}{e^2 n} < \frac{1}{4n^{3\rho}}.$$

□

In the following, we call a node u together with Δ neighbors $v_1, \ldots, v_\Delta$, a Δ-star.

Theorem 5.11. *Let G be a graph with n nodes and $2n^\varepsilon$ disjoint Δ-stars for a constant $\varepsilon > 0$. On G, every randomized one-shot coloring algorithm needs at least $\Omega(\Delta \log n / \log\log n)$ colors in expectation and with high probability.*

Proof. W.l.o.g., we can certainly assume that $n \geq n_0$ for a sufficiently large constant n_0. We choose $\rho \leq \varepsilon/4$ and consider n^ε of the $2n^\varepsilon$ disjoint Δ-stars. Let us call these n^ε Δ-stars $S_1, \ldots, S_{n^\varepsilon}$. Assume that the ID assignment of the n nodes of G is chosen uniformly at random from all ID assignments with IDs $1, \ldots, n$. The IDs of the star S_1 are perfectly random. We can therefore directly apply Lemma 5.10 and obtain that the probability that the center node of S_1 gets no color is at least $1/(8n^{3\rho})$. Consider star S_2. The IDs of the nodes of S_2 are chosen at random among the $n - \Delta - 1$ IDs that are not assigned to the nodes of S_1. Applying Lemma 5.10 we get that the probability that S_2 does not get a color is at least $1/(8(n-\Delta-1)^{3\rho}) \geq 1/(8n^{3\rho})$ independently of whether S_1 does get a color. The probability that the starts $S_1, \ldots, S_{n^\varepsilon}$ all get a color therefore is at most

$$\prod_{i=0}^{n^\varepsilon - 1} \left(1 - \frac{1}{8(n - i(\Delta+1))^{3\rho}}\right) \leq \left(1 - \frac{1}{8n^{3\rho}}\right)^{n^\varepsilon} \leq e^{-\frac{n^\varepsilon}{8n^{3\rho}}} \leq e^{-n^\rho/8}.$$

Hence, there is a constant $\eta > 0$ such that $\eta\Delta \ln n / \ln\ln n$ colors do not suffice with probability at least $1 - e^{-n^\rho/8}$ for a positive constant ρ. The lemma thus follows. □

References

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

[2] B. Balasundaram and S. Butenko. Graph domination, coloring and cliques in telecommunications. In M. Resende and P. Pardalos, editors, *Handbook of Optimization in Telecommunications*, pages 865–890. Springer, 2006.

[3] L. Barenboim and M. Elkin. Sublogarithmic distributed mis algorithm for sparse graphs using nash-williams decomposition. In *Proc. of 27th ACM Symposium on Principles of Distributed Computing (PODC)*, 2008.

[4] R. Cole and U. Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986.

[5] A. Czygrinow, M. Hańćkowiak, and M. Karoński. Distributed $O(\delta \log n)$-edge-coloring algorithm. In *Proc. of 9th Annual European Symposium on Algorithms (ESA)*, volume 2161 of *LNCS*, pages 345–355, 2001.

[6] G. De Marco and A. Pelc. Fast distributed graph coloring with $O(\Delta)$ colors. In *Proc. of 12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 630–635, 2001.

[7] P. Erdős, P. Frankl, and Z. Füredi. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics*, 51:79–89, 1985.

[8] C. Fortuin, J. Ginibre, and P. Kasteleyn. Correlation inequalities on some partially ordered sets. *Comm. in Mathematical Physics*, 22:89–103, 1971.

[9] P. Fraigniaud, C. Gavoille, D. Ilcinkas, and A. Pelc. Distributed computing with advice: Information sensitivity of graph coloring. In *Proc. of 34th Int. Coll. on Automata, Languages and Programming*, 2007.

[10] S. Gandham, M. Dawande, and R. Prakash. Link scheduling in sensor networks: Distributed edge coloring revisited. In *Proc. of 24th IEEE Conference on Computer Communications (INFOCOM)*, pages 2492–2501, 2005.

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

[12] D. A. Grable and A. Panconesi. Nearly optimal distributed edge coloring in $\mathrm{O}(\log \log n)$ rounds. *Random Structures and Algorithms*, 10(3):385–405, 1997.

[13] T. Herman and S. Tixeuil. A distributed TDMA slot assignment algorithm for wireless sensor networks. In *Proc. of 1st Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks*, pages 45–58, 2004.

[14] K. Kothapalli, M. Onus, C. Scheideler, and C. Schindelhauer. Distributed coloring in $O(\sqrt{\log n})$ bit rounds. In *Proc. of 20th IEEE Int. Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

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

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

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

[18] S. Mecke. MAC layer and coloring. In D. Wagner and R. Wattenhofer, editors, *Algorithms for Sensor and Ad Hoc Networks*, pages 63–80, 2007.

[19] T. Moscibroda and R. Wattenhofer. Coloring unstructured radio networks. In *Proc. of 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 39–48, 2005.

[20] M. Naor and L. Stockmeyer. What can be computed locally? In *Proc. of 25th ACM Symposium on Theory of Computing (STOC)*, pages 184–193, 1993.

[21] A. Panconesi and A. Srinivasan. On the complexity of distributed network decomposition. *Journal of Algorithms*, 20(2):581–592, 1995.

[22] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.

[23] A. Pelc. Personal communication.

[24] S. Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wireless Networks*, 5:81–94, 1999.

[25] I. Rhee, A. Warrier, J. Min, and L. Xu. DRAND: Distributed randomized TDMA scheduling for wireless ad-hoc networks. In *7th ACM Symp. on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 190–201, 2006.

[26] J. Schneider and R. Wattenhofer. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In *Proc. of 27th ACM Symposium on Principles of Distributed Computing (PODC)*, 2008.

[27] X. Zhang, J. Hong, L. Zhang, X. Shan, and V. Li. CP-TDMA: Coloring- and probability-based TDMA scheduling for wireless ad hoc networks. *IEICE Transactions on Communication*, E91-B(1):322–326, 2008.