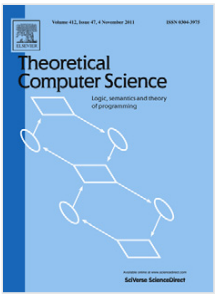

Symmetry breaking depending on the chromatic number or the neighborhood growth

Johannes Schneider, Michael Elkin, Roger Wattenhofer

PII: S0304-3975(12)00824-9
DOI: 10.1016/j.tcs.2012.09.004
Reference: TCS 9035

To appear in: *Theoretical Computer Science*

Please cite this article as: J. Schneider, M. Elkin, R. Wattenhofer, Symmetry breaking depending on the chromatic number or the neighborhood growth, *Theoretical Computer Science* (2012), doi:10.1016/j.tcs.2012.09.004

Symmetry Breaking Depending on the Chromatic Number or the Neighborhood Growth*

Johannes Schneider
Computer Eng. and Networks Lab.
8092 Zurich, Switzerland
schneider@tik.ee.ethz.ch

Michael Elkin
Department of Computer Science
Ben-Gurion Univ., Israel
elkinm@cs.bgu.ac.il

Roger Wattenhofer
Computer Eng. and Networks Lab.
8092 Zurich, Switzerland
wattenhofer@tik.ee.ethz.ch

Abstract

We deterministically compute a $\Delta + 1$ coloring and a maximal independent set(MIS) in time $O(\Delta^{1/2+\Theta(1/\sqrt{h})} + \log^* n)$ for $\Delta_{1+i} \leq \Delta^{1+i/h}$, where Δ_j is defined as the maximal number of nodes within distance j for a node and $\Delta := \Delta_1$. Our greedy coloring and MIS algorithms improve the state-of-the-art algorithms running in $O(\Delta + \log^* n)$ for a large class of graphs, i.e., graphs of (moderate) neighborhood growth with $h \geq 36$. We also state and analyze a randomized coloring algorithm in terms of the chromatic number, the run time and the used colors. Our algorithm runs in time $O(\log \chi + \log^* n)$ for $\Delta \in \Omega(\log^{1+1/\log^* n} n)$ and $\chi \in O(\Delta/\log^{1+1/\log^* n} n)$. For graphs of polylogarithmic chromatic number the analysis reveals an exponential gap compared to the fastest $\Delta + 1$ coloring algorithm running in time $O(\log \Delta + \sqrt{\log n})$. The algorithm works without knowledge of χ and uses less than Δ colors, i.e., $(1 - 1/O(\chi))\Delta$ with high probability. To the best of our knowledge this is the first distributed algorithm for (such) general graphs taking the chromatic number χ into account. We also improve on the state of the art deterministic computation of $(2, c)$-ruling sets.

*This is the journal version of [25].

1 Introduction

The coloring and maximal independent set problems are typical symmetry breaking tasks. Coloring is a fundamental problem with many applications. Unfortunately, even in a centralized setting, where the whole graph is known, approximating the chromatic number (the minimal number of needed colors), is currently computationally infeasible for general graphs and believed to take exponential running time. Thus, basically any reduction of the used colors below $\Delta + 1$ – even just to Δ – is non-trivial in general. Looking at the problem in a distributed setting, i.e., without global knowledge of the graph,

makes the problem harder, since coloring is not a purely "local" problem, i.e., nodes that are far from each other have an impact on each other (and the chromatic number). Therefore, it is not surprising that all previous work has targeted to compute a $\Delta+1$ coloring in general graphs as fast as possible (or resorted to very restricted graph classes). However, this somehow overlooks the original goal of the coloring problem, i.e., use as little colors as possible. Though in distributed computing the focus is often on communication, in many cases keeping the number of colors low outweighs the importance of minimizing communication. For example, a TDMA schedule can be derived from a (2-hop) coloring. The length of the schedule (and thus the throughput of the network) is determined by the number of employed colors.

In this paper, we also consider fast distributed computation of $\Delta + 1$ colorings and maximal independent sets in the first part. The algorithms for both problems are similar. In the second part we are interested in both using less than $\Delta+1$ colors and efficient computation. For sparse graphs, such as trees and planar graphs, as well as for dense graphs, e.g., cliques and unit disk graphs (UDG), efficient distributed algorithms are known that have both "good" time complexity and "good" approximation ratio of the chromatic number. Sparse graphs typically restrict the number of edges to be linear in the number of nodes. Unit disk graphs restrict the number of independent nodes within distance i to be bounded by a polynomial function $f(i)$. Our requirements on the graph are much less stringent than for UDGs, i.e., we do not restrict the number of independent nodes to grow dependent on the distance only. We allow for growth of the neighborhood dependent on the distance and also on Δ, i.e., n. For illustration, if the number of nodes within distance $i+1$ is bounded by $\Delta^{1+i/10}$ our deterministic MIS algorithm improves on the state-of-the-art algorithms running in linear time in Δ by more than a factor of $\Delta^{1/10}$. Note, for any graph the size of the neighborhood within distance $1 + i$ is bounded by Δ^{1+i}. Additionally, if the size of the neighborhood within distance i of a graph is lower bounded by $\Delta^{h \cdot i}$ for an aribtrary constant h then the graph can have only small diameter, i.e., $O(\log \Delta)$. In such a case a trivial algorithm collecting the whole graph would allow for a coloring exponentially faster than the current state of the art deterministic algorithms running in time $O(\Delta + \log^* n)$ for small Δ already. Therefore, we believe that for many graphs that are considered "difficult" to color we significantly improve on the best known algorithms. The guarantee on the number of used colors is the same as in previous work, i.e., $\Delta + 1$. Despite the hardness of the coloring problem, intuitively, it should be possible to color a graph with small chromatic number with fewer colors and also a lot faster than a graph with large chromatic number. Our (randomized) algorithm in the second part of the paper shows that this in indeed the case. The algorithm works without knowledge of the chromatic number χ.

2 Model and Definitions

Communication among nodes is done in synchronous rounds without collisions. Each node can exchange one distinct message with each neighbor. Nodes start the algorithm concurrently. The communication network is modeled with a graph $G = (V, E)$. The distance between two nodes u, v is given by the length of the shortest path between nodes u and v. For a node v its

neighborhood $N^r(v)$ represents all nodes within distance r of v (not including v itself). By $N(v)$ we denote $N^1(v)$ and by $N_+(v) := N(v) \cup v$. The degree $d(v)$ of a node v is defined as $|N(v)|$, $d_+(v) := |N_+(v)|$, $\Delta := \max_{u \in V} d(u)$ and $\Delta_i := \max_{u \in V} |N^i(u)|$. By $G^i = (V, E^i)$ of $G = (V, E)$ we denote the graph where for each node $v \in V$ there is an edge to each node $u \in N^i(v)$. In a (vertex) coloring any two adjacent nodes u, v have a different color. A set $T \subseteq V$ is said to be independent in G if no two nodes $u, v \in T$ are neighbors. We consider a slight generalization of ruling sets: A set $R \subseteq V$ is (α, β)-ruling for W if every two nodes in the set R have distance at least α in G and any node $w \in W$ not in the set R has a node in the set R within distance β. An (α, β)-ruling set is short for an (α, β)-ruling for V (the set of all nodes). A (2,1)-ruling set is called a maximal independent set (MIS). The function $\log^* n$ states how often one has to take the (iterated) logarithm to get at most 1, i.e., $\log^{(\log^* n)} n \leq 1$. The term "with high probability" abbreviated by w.h.p. denotes a number $1 - 1/n^c$ for an arbitrary constant $c > 1$.

Every node knows an upper bound on the total number of nodes n and the maximal degree Δ. This assumption is probably not necessary using techniques from [16]. We also use the following Chernoff bound:

Theorem 1. *The probability $Pr(X < (1 - \delta)a)$ that the number X of occurred independent events $X_i \in \{0, 1\}$, i.e., $X := \sum X_i$, is less than $(1 - \delta)$ times a with $a \leq \mathbb{E}[X]$ can be upper bounded by $e^{-a\delta^2/2}$. The probability $Pr(X > (1 + \delta)b)$ that the sum X is more than $(1+\delta)b$ with $b \geq \mathbb{E}[X]$ with $\delta \in [0, 1]$ can be upper bounded by $e^{-b\delta^2/3}$.*

Corollary 2. *The probability that the number X of occurred independent events $X_i \in \{0, 1\}$, i.e., $X := \sum X_i$, is less than $\mathbb{E}[X]/2$ is at most $e^{-\mathbb{E}[X]/8}$ and the probability that is more than $3\mathbb{E}[X]/2$ is bounded by $e^{-\mathbb{E}[X]/12}$.*

3 Related Work

Distributed coloring is a well studied problem in general graphs in the message passing model, e.g., [6, 7, 24, 22, 19, 18]. There is a trade-off between the number of used colors and the running time of an algorithm. Even allowing a constant factor more colors can have a dramatic influence on the running time of a coloring algorithm, i.e., in [24] the gap between the running time of an $O(\Delta)$ and an $\Delta + 1$ coloring algorithm can be more than exponential for randomized algorithms. More precisely, a $\Delta+1$ coloring is computed in time $O(\log \Delta + \sqrt{\log n})$ and an $\mathrm{O}(\Delta + \log^{1+1/\log^* n} n)$ coloring in time $\mathrm{O}(\log^* n)$. When using $O(\Delta^2)$ colors, a coloring can be computed in time $\mathrm{O}(\log^* n)$ [19], which is asymptotically optimal for constant degree graphs due to a lower bound of time $\Omega(\log^* n)$ for three coloring of an n-cycle. Using $\mathrm{O}(\Delta^{1+o(1)})$ colors [7] gives a deterministic algorithm running in time $\mathrm{O}(f(\Delta) \log \Delta \log n)$ where $f(\Delta) = \omega(1)$ is an arbitrarily slow growing function in Δ. To this date, the fastest deterministic algorithm to compute a $\Delta + 1$ coloring in general graphs requires $\mathrm{O}(\Delta + \log^* n)$ [6, 17] or $n^{O(1)/\sqrt{\log n}}$ time [22]. For a (d, c) cluster decomposition, each cluster has diameter d and if all nodes in a cluster are assigned the same color then the graph can be colored with c colors such that adjacent nodes from distinct clusters have distinct colors. Then all clusters of the same color compute a solution in parallel, e.g., the computation is carried out by a leader. The computation of the decomposition is quite involved merging clusters and re-

Colors	Type	Time
$(1-1/O(\chi))(\Delta+1)$	ra.	$O(\log\chi+\log^* n)$ [This paper] for $\Delta\in\Omega(\log^{1+1/\log^* n} n)$ and $\chi\in O(\Delta/\log^{1+1/\log^* n} n)$
$\Delta+1$	ra.	$O(\log\Delta+\sqrt{\log n})$ [24]
	ra.	$O(\log n)$ [20, 1, 14]
	det.	$O(\Delta^{1/2+\Theta(1/\sqrt{h})}+\log^* n)$ [This paper] for $\lvert N^{i+1}(v)\rvert\leq\Delta^{1+i/h}$
	det.	$O(\Delta+\log^* n)$ [17, 6]
	det.	$O(2^{\sqrt{\log n}})$ [22]
$O(\Delta+\log^{1+1/\log^* n})$	ra.	$O(\log^* n)$[24]
$O(\Delta)$	det.	$O(\Delta^c\log n)$[7]
$O(\Delta\log^{(c)} n+\log^{1+1/c} n)$	ra.	$O(1)$ [24]
$O(\Delta^{1+o(1)})$	det.	$O(\log\Delta\log n)$[7]
$O(\Delta^2)$	det.	$O(\log^* n)$ [19, 21]

Table 1: Comparison of coloring algorithms, where c is an arbitrary constant

cursively computing a decomposition on the resulting graph of merged clusters until the maximum degree in the graph is sufficiently small. In contrast we repeatedly (but non-recursively) compute network decompositions differently, i.e., using [24]. The algorithms [6, 17] improved on [18] by a factor of $\log\Delta$ through employing defective colorings, i.e., several nodes initially choose the same color. However, through multiple iterations the number of adjacent nodes with the same color is reduced until a proper coloring is achieved. In [7] defective colorings were combined with tree decompositions [5]. In comparison, our deterministic algorithm improves the linear running time in Δ by a factor Δ^d for a constant d for a large class of graphs by iteratively computing ruling sets, such that a node in the ruling set can color its two hop neighborhood.

Overall $\Delta+1$ coloring has probably attracted more attention than employing $O(\Delta)$ or more colors. Using less than $\Delta+1$ colors is not possible for complete graphs – not even in a centralized setting, where the entire graph is known. An algorithm in [15] parallelizes Brooks' sequential algorithm to obtain a Δ coloring from a $\Delta+1$ coloring. In a centralized setting the authors of [2] showed how to approximate a three-colorable graph using $O(n^{0.2111})$ colors. Some centralized algorithms iteratively compute large independent sets, e.g., [8]. It seems tempting to apply the same ideas in a distributed setting, e.g., a parallel minimum greedy algorithm for computing large independent sets is given in [13]. It has approximation ratio $(\Delta+2)/3$. However, the algorithm runs in time polynomial in Δ and logarithmic in n and thus is far from efficient. For some restricted graph classes, there are algorithms that allow for better approximations in a distributed setting. A Δ/k coloring for $\Delta\in O(\log^{1+c} n)$ for a constant c with $k\leq c_1(c)\log\Delta$ where constant c_1 depends on c is given in [12]. It works for quite restricted graphs (only), i.e., graphs that are Δ-regular, triangle free and $\Delta\in O(\log^{1+c} n)$. Throughout the algorithm a node increases its proba-

bility to be active. An active node chooses a color uniformly at random. The algorithm runs in $O(k + \log n/\log\Delta)$ rounds.

Constant approximations of the chromatic number are achieved for growth bounded graphs (e.g. unit disk graphs) [23] and for many types of sparse graphs [5]. In [9] the existence of graphs of arbitrarily high girth was shown such that $\chi \in \Omega(\Delta/\log\Delta)$. Since graphs of high girth locally look like trees and trees can be colored with two colors only, this implies that coloring is a non-local phenomenon. Thus, a distributed algorithm that only knows parts of the graph and is unaware of global parameters such as χ, has a clear disadvantage compared to a centralized algorithm. In [10] a lower bound is given that any distributed algorithm choosing computing a greedy coloring requires time at least $\Omega(\log n/\log\log n)$ for some large Δ. Our Algorithm *RulingColoring* assigns colors greedily.

We give a randomized coloring algorithm in terms of the chromatic number of a graph which uses ideas from [24]. Given a set of colors $\{0, 1, \ldots, f(\Delta)\}$ for an arbitrary function f with $f(x) \geq x$ [24] computes an $f(x) + 1$ coloring. The run time depends on f, i.e., for $f(\Delta) := \Delta$ Algorithm *DeltaPlus1Coloring*[24] takes time $O(\log\Delta + \sqrt{\log n})$. For $f(\Delta) := O(\Delta + \log^{1+1/\log^* n} n)$ Algorithm *ConstDeltaColoring*[24] takes only $O(\log^* n)$ time. Both Algorithms from [24] operate analogously: In each communication round a node chooses a subset of all available colors and keeps one of the colors, if no neighbor has chosen the same color. In [26] the message size of [24] is improved while maintaining the time complexity.

The fastest deterministic algorithms for obtaining a MIS for graphs with $\Delta \in O(2^{\sqrt{\log n}})$ set actually start out by computing a coloring [6, 17]. Once the coloring is obtained a node with color i joins the MIS in round i, if none of its neighbors has joined already. Thus their running time is the time to compute the coloring plus the number of used colors, i.e., $O(\Delta + \log^* n)$. In contrast our MIS algorithm for graphs of moderate expansion computes a MIS directly. [22] gives the fastest deterministic algorithm for $\Delta+1$ coloring and MIS for graphs with $\Delta \in \Omega(2^{\sqrt{n}})$. It recursively computes (d, c)-cluster decompostions.

A (α, β)-ruling set [4] defines a network decomposition, such that any component has diameter at least α and at most β. In [4] it is shown how to compute a $(k, k\log n)$-ruling set in time $\mathrm{O}(k\log n)$. In [11] a $(1, \log\log\Delta)$-ruling set is computed in time $\mathrm{O}(\log\log\Delta)$ such that each node in the ruling set has at most $\mathrm{O}(\log^5 n)$ neighbors also in the ruling set. In [24] $(2, c)$-ruling sets are computed in time $2^c d^{1/c}$ given a d-coloring of the graph or, more generally, a $(k+1, ck)$-ruling set in G in time $k\cdot 2^c d^{1/c}$ given a d-coloring of the graph G^{k+1}. Such ruling sets allow to obtain network decompositions (and covers) of equal diameter (up to a constant factor). Such a cover is of interest if the main concern is communication time among all nodes within a cluster. We improve on this result by computing a $(k+1, ck)$-ruling set in G in time $k\cdot cd^{1/c}$ given a d-coloring of the graph G^k.

4 Ruling sets

Given a d-coloring for a set W we compute a $(2, c)$-ruling set for W in time $c \cdot d^{1/c}$ using Algorithm *RulingSet* with node set W having colors from $[0, d-1]$. In short the algorithm operates as follows: Partition the vertex set W into $W_0 \cup W_1 \cup \cdots \cup W_{t-1}$ vertices for $t := d^{1/c}$. Set W_0 contains all vertices of colors $\{0, \ldots, d/t-1\}$, W_1

contains all vertices of colors $\{d/t, \ldots, 2d/t-1\}$, etc. Recurse in each W_i to compute (in parallel) ruling set U_i for each W_i. Set $U'_0 := U_0$. For each vertex $u_1 \in U_1$ that has a neighbor in U'_0, eliminate u_1. Denote by U'_1 the subset of vertices that remained in U_1 after this process. For each $u_2 \in U_2$ that has a neighbor in $(U'_0 \cup U'_1)$ do the same. Afterwards, for each $u_3 \in U_3$ that has a neighbor in $(U'_0 \cup U'_1 \cup U'_2)$ do the same. Continue in this way for $t-2$ iterations. All nodes $V \setminus W$ stay inactive, i.e., do not take part in the computation.

Algorithm $(2,c)$***-RulingSet***
Given nodes W with colors from $[a,b]$ compute ruling set U

1: $t := (b-a)^{1/c}$
2: **if** $t < 2$ **then**
3: $\quad U := W$
4: **else**
5: $\quad r_i := [i \cdot t, (i+1) \cdot t - 1]$ {Colors in W_i}
6: $\quad W_i := \{v \in W | color(v) \in r_i\}$
7: $\quad U_i :=$ Result of RulingSet(W_i, r_i) {Computed in parallel $\forall i \in [0, (b-a)/t - 1]$}
8: $\quad U := U_0$
9: $\quad$ **for** $i = 1$ to $t-1$ **do**
10: $\quad\quad$ **if** $(v \in U_i) \wedge (\nexists u \in (N(v) \cap U))$ **then** $U := U \cup v$ **end if**
11: $\quad$ **end for**
12: **end if**

Theorem 3. *Given a d-coloring of a graph Algorithm* RulingSet *computes a* $(2,c)$*-ruling set for* W *in time* $c \cdot d^{1/c}$.

Let W_i^j be a set of nodes with colors in r_i^j performing the jth recursive call to Algorithm *RulingSet* with parameters (W_i^j, r_i^j). Analogously, let U_i^j be the returned ruling set of that call. Let t^j be the t used in the jth recursion, i.e. $t^j := (((b-a)^{1/c})^{1/c})^{\cdots}$, where the exponent $1/c$ is applied j times.

Proof. Given a d-coloring for the first call, we have for the range of colors $b - a \leq d$. The time complexity is given by the following recursion: $T(d) = T(d/t) + (t-1)$. This equation is due to the following observations: Every iteration reduces the problem size by a factor t, i.e., the nodes are partitioned according to their colors into t sets. Computing a ruling set for recursion j requires iterating through $t-1$ sets. On the bottom level of the recursion, we just take the entire color class. We obtain a ruling set in time $T(d) = T(d/t) + (t-1) \leq \log_t d \cdot t \leq c \cdot d^{1/c}$. Thus the total number of recursions is $c \leq \log_t d$. Next, we prove correctness, i.e. that the distance between two nodes in the ruling sets is at most c and at least 2. In the last (cth) recursion we have $t^c < 2$ and $U_i^c := W$ with $W := \{v\}$ for some node v. (The case $W = \{\}$ is trivial.) In the second to last iteration $c-1$ we have $t^{c-1} \geq 2$. The result U_i^{c-1} is initialized to U_0^c. When iterating through all sets U_k^c with $k \in [1, t^{c-1} - 1]$, a node $v \in U_k^c$ is either joined U_i^{c-1} or a neighbor $u \in N(v)$ has been joined U_i^{c-1}. In the same manner in the jth ($j \leq c-2$) recursion a node v belonging to set U_i^{j+1} is either joined set U_i^j or a neighbor $u \in N(v)$ is joined the set. Since any two adjacent nodes $u, v \in W_i^{j+1}$ have distinct colors, they will not be considered in the same iteration and thus they will not both be joined U_i^{j+1}. Thus, after c recursions the distance to a node in the ruling set can be at most c and it is also at least 2. □

Next, we extend Algorithm $(2,c)$-*RulingSet* to compute a $(k+1, ck)$-ruling set for $W \subseteq V$ in G for an aribtrary integer $k \geq 1$. When com-

puting a $(2,c)$-ruling set for W it is sufficient if all nodes $V \setminus W$ remain quiet and all nodes in W simply communicate with their neighbors being in W. When computing a $(k+1,ck)$-ruling set for W with $k \geq 1$ in every communication round in Algorithm $(2,c)$-*RulingSet* turns into k communication rounds. Let a step be one communication round in Algorithm $(2,c)$-*RulingSet* and k communication rounds, when computing a $(k+1,ck)$-ruling set for W. The first round is the same as in Algorithm $(2,c)$-*RulingSet* followed by $k-1$ forwarding rounds, i.e., in round $l \in [2,k-1]$ a node simply retransmits some values it has received in a prior round m with $m < l$. More precisely, in the lth iteration any node $v \in V$ must forward the smallest color for every set W_i^j it has received during any of the communication rounds $m < l$ in this step. Thus, a node only joins the ruling set, if its color is smallest among any color received of a node W_i^j within distance k. Furthermore, if a node u is joined the ruling set during this step then its neighbors $w \in N(u)$ forward that some node is joined the ruling set (It does not matter which node joined). Next, the neighbors of w forward this information and so on for $k-2$ rounds, i.e., until the step is over. Any node u becoming aware that some node $w \in N^k(v)$ being joined the ruling set stops attempting to join the MIS and only forwards messages.

Theorem 4. *Given a d-coloring of nodes W in a graph G^k the extended Algorithm* RulingSet *computes a $(k+1,ck)$-ruling set for W in G in time $(k+1)c \cdot d^{1/c}$ for an aribtrary integer $k \geq 1$.*

The proof is analogous to Theorem 3. We restate the parts that require adapation.

Proof. The time complexity is given by the following recursion: $T(d) = T(d/t) + k \cdot (t-1)$. This equation is due to the following observations: Every iteration reduces the problem size by a factor t, i.e., the nodes are partitioned according to their colors into t sets. Computing a ruling set for recursion j requires iterating through $(t-1)$ sets. The multiplication with k follows since the exchange of a message for two adjacent nodes u, v in G^k requires not one round but k rounds of communication (in G). On the bottom level of the recursion, i.e., the cth recursion, we just take the entire color class. A ruling set is computed in time $T(d) = T(d/t) + k(t-1) \leq k \log_t d \cdot t \leq kc \cdot d^{1/c}$.

By definition of the algorithm a node $w \in W$ is joined the ruling set if and only if its color is smallest for all nodes in W within distance k. Since we are given a coloring of the graph G^k all nodes within distance k must have distinct colors and, therefore, if a node $w \in W$ has smallest color within distance k from it then there can be no other node $u \in N^k(w)$ having the same color. If node w having smallest color is joined the ruling set then this information is forwarded up to distance k from u and thus no node within distance k in G from node w is joined the ruling set after w is joined. Therefore, the distance between two nodes in the ruling set must be at least $k+1$. For graph G for a recursion j either a node $v \in W_i^j$ is joined the ruling set U_i^j or it gets a node $u \in N^k(v)$ at distance at most k in the ruling set $u \in W_i^j$. □

In fact, it is possible to decrease the running time by a factor of a by increasing the bound on the maximal distance to a node in the ruling set by the same factor a. For simplicity we only look at $(2,ac)$-ruling sets for V running in time $c/a \cdot d^{1/c}$. The algorithm *RulingSetTradeOff* given next is similar to the original one but contains one additional step to take into account

the reduction in the running time by a factor of a. Instead of dividing the nodes into t sets and going through all t sets sequentially, we combine a of the t sets into a $(2,a)$-ruling set W'_l for $W_{la}, W_{la+1}, \ldots, W_{(l+1)a-1}$ with $l \in [0, t/a-1]$ in one round. Then we continue for W'_l in the same manner as for a set W_i in the original algorithm to get the ruling set U_l.

Given a d-coloring we partition the vertex set $V(=:W)$ into $W_0 \cup W_1 \cup \cdots \cup W_{t-1}$ vertices for $t := d^{1/c}$. Set W_0 contains all vertices of colors $\{0, \ldots, d/t-1\}$, W_1 contains all vertices of colors $\{d/t, \ldots, 2d/t+1\}$, etc. Set $W'_l := W_{a \cdot l}$ with $0 \leq l \leq t/a-1$. For each vertex $w \in W_{al+i}$ with $1 \leq i < a$ add w to W'_l, if there is no neighbor $u \in (W_{al+j} \cap N(w))$ with $j \in [l \cdot a, i \leq (l+1)a-1]$. Recurse in each W'_l to compute (in parallel) ruling set U_l. Set $U'_0 := U_0$. For each vertex $u_1 \in U_1$ that has a neighbor in U'_0, eliminate u_1. Denote by U'_1 the subset of vertices that remained in U_1 after this process. For each $u_2 \in U_2$ that has a neighbor in $(U'_0 \cup U'_1)$ do the same. Afterwards, for each $u_3 \in U_3$ that has a neighbor in $(U'_0 \cup U'_1 \cup U'_2)$ do the same. Continue in this way for $t/a-1$ iterations.

Theorem 5. *Given a d-coloring of a graph Algorithm* RulingSetTradeOff *computes a $(2, ac)$-ruling set in time $c/a \cdot d^{1/c}$.*

The proof is analogous to Theorem 3. We restate the parts that require adapation.

Proof. For a recursion j every node $v \in W^j_{al+i}$ (for some l, i) gets a node $u \in N^a_+(v)$ at distance at most a being joined some W'^l_k. This follows, since nodes W^j_{al} are joined W'_l and a node $w \in W^j_{al+i}$ with $i \in [1, a-1]$ is not joined only if there is a neighbor $u \in N(w)$ being also in W^j_{al+i-1}. Thus, the longest path $P = (x_0, x_1, ..., x_k)$ from a node $x_0 \in W^j_{al+i}$ not in W'_l to a node $x_k \in W'_l$ is given by $x_i \in W^j_{al+o}$ for $i \in [0, a]$, i.e. P is of length $k := a$.

The time complexity is given by the following recursion: $T(d) = T(d/t) + (t/a-1)$. This equation is due to the following observations: Every iteration reduces the problem size by a factor t, i.e., the nodes are partitioned according to their colors into t/a sets W_l. Computing sets W'_l from sets W_{al+i} takes $O(1)$ time. Computing a ruling set for recursion j requires iterating through $(t/a-1)$ sets. On the bottom level of the recursion, i.e., the cth recursion, just take the entire color class. We obtain a ruling set in time $T(d) = T(d/t) + (t/a-1) \leq \log_t d \cdot t/a = c/a \cdot d^{1/c}$. □

5 Deterministic Coloring

For coloring one can either let each node decide itself on a color or decompose the graph into (disjoint) clusters and elect a leader to coordinate the coloring in a cluster. Our deterministic algorithm follows the later strategy by iteratively computing ruling sets. Each node in the set can color itself and its neighbors in a greedy manner. To make fast progress, only nodes can join the ruling set that color many nodes. Once no node has sufficiently many neighbors to color, i.e., less than Δ^ϵ (for a parameter ϵ of the algorithm), the nodes switch to another algorithm [6, 17].

When a node v is in the ruling set, it gets to know all nodes $N^2(v)$ and assigns colors to at least Δ^ϵ uncolored nodes by taking into account previously assigned colors. Node v can assign colors to vertices of $N_+(v)$ locally. It does so in a greedy manner, i.e., it looks at a node $u \in N_+(v)$ and chooses the smallest color that is not already given to a node $w \in N(u)$. Potentially, two nearby nodes u, v in the ruling set might concurrently assign the same colors to

adjacent nodes, e.g., node u assigns color 1 to node x, node v assigns 1 to y and x, y are neighbors. To prevent this problem any two nodes u, v in the ruling-set must have distance at least 4. Thus, the algorithm computes a $(4, 3c)$- ruling set, where c is a parameter of the Algorithm *RulingSet* in the previous section. Computing a $(4, 3c)$-ruling set in turn demands that two nodes u, v within distance 3 have distinct labels, therefore we start out by computing an $O(|N^3(v)|^2)$ coloring in the graph G^3 using [19] (or [7] to compute an O($|N^3(v)|$) coloring). [19] takes a graph H and computes a $O(\Delta^2)$ coloring in time $O(\log^* n)$.

We call a node u *active* if color $col(u) > \Delta$. Nodes that are not *active* are removed from the graph, i.e. do not communicate any more. After the initial coloring, a node participates in iteratively computing $(4, 3c)$-ruling sets. As soon as a node v has less than Δ^ϵ active neighbors for some parameter ϵ it does not join the ruling set itself but still forwards messages and still might get colored by a neighbor. As soon as all nodes $w \in N^3(u)$ within distance 3 have less than Δ^ϵ active neighbors node w switches to another algorithm, i.e. it executes [6] or [17]. [6, 17] compute a $\Delta + 1$ coloring for a graph H in time $O(\Delta + \log^* n)$. Nodes might end the while loop at different times. However, [6, 17] assume a synchronous start of all nodes. To deal with synchronization issues we can either execute the algorithm in "lock step", i.e. each loop is executed for a fixed number of rounds, or, alternatively, use a well-known synchronizer, e.g. an α synchronizer [3]. We decided on the later option, i.e., a node executing algorithm [6] only transmits a message (say the transmission is the tth step of the algorithm), if it has received all messages by its active neighbors that is supposed to have received due to [6] (or [17]) up to step $t-1$. Using [6, 17] the remaining active nodes U compute a $\Delta^\epsilon+1$ coloring among themselves, i.e., disregarding non-active nodes $u \in V \backslash U$, which have already obtained a color $col(u) \leq \Delta$. Thus, each node $v \in U$ obtains a color $col'(v) \in [0, \Delta^\epsilon]$. As a last step we sequentially go through all $\Delta^\epsilon + 1$ colors and let node $v \in U$ with color $col'(v) = i$ choose greedily a final color $col(v) \in [0, \Delta]$ in round i, taking into account all previously assigned colors $col(u)$ with $u \in V$.

Let $N_+^{act}(v)$ be all active nodes $u \in N_+(v)$. Set $d_+^{act}(v) := |N_+^{act}(v)|$.

Before elaborating on the running time for Algorithm *RulingColoring* we address correctness.

Theorem 6. *Algorithm* RulingColoring *computes a proper* $\Delta + 1$ *coloring.*

Proof. By definition of a $(4, 3c)$-ruling set R any two nodes $u, v \in R$ have distance 4. Therefore, if all nodes in the ruling set color their neighbors, there will not be any color conflicts. The remaining nodes $u \in U$, i.e., those with less than Δ^ϵ neighbors of color larger than Δ, are assigned a color $col'(u) \in [0, \Delta^\epsilon]$ (see Theorem 4.6 in [6]). If a node $u \in U$ with $col'(u) = i$ chooses a final color $col(u)$ in round i, no neighbor $w \in N(u)$ makes a choice at the same time (since all neighbors have distinct colors $col'(w) \neq col'(u)$) and thus the resulting coloring is proper. □

Theorem 7. *Algorithm* RulingColoring *computes a* $\Delta+1$ *coloring in time* $O(\Delta^{1/2+\Theta(1/\sqrt{h})} + \log^* n)$ *for* $\Delta_{i+1} \leq \Delta^{1+i/h}$. *The running time is sublinear in* Δ *for* $h \geq 36$.

Proof. Correctness follows by Theorem 6. Since we compute every ruling set for all active nodes v with $d_+^{act}(u) \geq \Delta^\epsilon$, any node joining the ruling set colors $\Delta^\epsilon + 1$ nodes, i.e., renders them inactive. Every active node (participating in the

Algorithm *RulingColoring* for graph G with $\Delta_{i+1} \leq \Delta^{1+i/h}$

1: $\epsilon := 1/2 + \frac{3h+2\sqrt{h}+3}{h^{3/2}}$
2: $col(v) :=$ Compute an $O((\Delta_3)^2)$ coloring in the graph G^3 using [19]
3: **while** $\exists u \in N_+^3(v)$ s.t. $d_+^{act}(u) \geq \Delta^\epsilon$ **do**
4: Compute a $(4, 3\left\lfloor\sqrt{h}\right\rfloor)$-ruling set R for active nodes
5: **if** $v \in R$ **then**
6: (Greedily) color all nodes $u \in N_+^{act}(v)$
7: **end if**
8: **end while**
9: **if** v is active **then**
10: $col'(v) :=$ color from $[0, \Delta^\epsilon]$ using [6]
11: **for** $i = 0..\Delta^\epsilon$ **do**
12: **if** $col'(v) = i$ **then**
13: Greedily pick color $col(v) \in [0, \Delta]$, s.t. $\nexists u \in N(v)$ with $col(u) = col(v)$
14: **end if**
15: **end for**
16: **end if**

computation of a ruling set) gets a node in the ruling set within distance $3c$. Therefore, any node that has more than Δ^ϵ active neighbors gets at least Δ^ϵ inactive nodes within distance $3c+1$ by computing one ruling set. The total time until node v gets colored with a color of at most Δ or has less than Δ^ϵ neighbors with color larger Δ is given by the following terms: The time it takes to compute a $O(|N^3(v)|^2)$ coloring using [19], i.e., $O(\log^* n)$. In addition, the total number of nodes $|N^{3c+1}(v)| \leq \Delta_{3c+1}$ within distance $3c+1$ divided by the number of nodes that get their final color for a computation of a ruling set, i.e., at least Δ^ϵ, multiplied by the time it takes to color the nodes and to compute one ruling set, i.e., $O(c \cdot (\Delta_3)^{2/c})$ (see Theorem 4). In total this results in time $O(c\Delta_{3c+1}/\Delta^\epsilon \cdot (\Delta_3)^{2/c})$. To compute color $col'(u)$ for the remaining nodes $u \in U$, i.e., those with at most Δ^ϵ nodes $w \in N_+(u)$ with color larger Δ, using [6] (or [17]) takes time $O(\Delta^\epsilon + \log^* n)$. Considering each $col'(u) \in [0, \Delta^\epsilon]$ in a distinct round requires time $O(\Delta^\epsilon)$. This yields $O(c\Delta_{3c+1} \cdot (\Delta_3)^{2/c}/\Delta^\epsilon + \Delta^\epsilon + \log^* n)$. This term is minimized for $c\Delta_{3c+1} \cdot (\Delta_3)^{2/c}/\Delta^\epsilon = \Delta^\epsilon$. Neglecting the factor c, using $\Delta_{i+1} = \Delta^{1+i/h}$ and taking the logarithm, we get $1 + 3c/h + 2(1+2/h)/c - \epsilon = \epsilon$, i.e., $\epsilon = 1/2 + \frac{3c^2+2h+4}{2ch}$. Clearly, the larger h the smaller the expansion and thus the faster our coloring algorithm runs. Thus, in case h is large, i.e., $h \notin O(1)$, the time complexity of our algorithm only improves compared to h being constant. Therefore, assume $h \in \Theta(1)$. Setting parameter $c := \sqrt{h}$ yields $\epsilon := 1/2 + \frac{3h+2\sqrt{h}+3}{h^{3/2}}$. We get running time linear in Δ, i.e., $O(\Delta + \log^* n)$, for $\epsilon = 1 = 1/2 + \frac{5h+4}{h^{3/2}}$, i.e., $h = 26.53...$. Since we have $c := \sqrt{h}$ and c must be integer, we get $c := 6$, i.e., sublinear running time for $h \geq c^2 = 36$.

Substituting $c := \sqrt{h}$ and $\epsilon := 1/2 + \frac{5h+4}{h^{3/2}} \leq 1/2 + \Theta(1/\sqrt{h})$ in $O(c\Delta_{3c+1} \cdot (\Delta_3)^{2/c}/\Delta^\epsilon + \Delta^\epsilon + \log^* n)$ yields $O(\sqrt{h} \cdot \Delta^{1/2+\Theta(1/\sqrt{h})} + \log^* n) = O(\Delta^{1/2+\Theta(1/\sqrt{h})} + \log^* n)$.

□

One might start out by computing an $\mathrm{O}((\Delta_3)^{2-a})$ coloring in the graph G^3 using [6] in time $O((\Delta_3)^a + \log^* n)$ instead of an $\mathrm{O}((\Delta_3)^2 + \log^* n)$ coloring using [19]. Depending on the maximal degree Δ, it might be better to compute an (initial) $O(|N^3(v)|)$ coloring in time $O((\Delta_3)^{c_0} \log \Delta_3 \log n)$ for an arbitrary small constant c_0 using [7]. However, this only influences the constants.

Algorithm *RulingColoring* as well as the fastest algorithm for large Δ [22] use large messages, i.e., potentially more than polylogarithmic in n. In particular the initial coloring in G^3 using [19] or [7] and the assignment of colors by nodes in the ruling set requires collecting all connectivity information up to distance 3 of a node.

6 Deterministic MIS

The MIS algorithm *RulingMIS* operates in a similar manner as the previously described coloring algorithm. We start out with a coloring. A node v iteratively participates in computing $(2, c)$-ruling sets, which are added to the MIS, as long as there are at least Δ^ϵ *active* nodes $u \in N_+(v)$. We call a node u *active* if the node u is not in the MIS and also not adjacent to a node $w \in N(u)$ in the MIS. The remaining nodes are dealt with using [6] or [17] with a synchronizer [3] (as described in the previous section).

Let $N_+^{act}(v)$ be all active nodes $u \in N_+(v)$. Set $d_+^{act}(v) := |N_+^{act}(v)|$.

Algorithm *RulingMIS* for graph G with $\Delta_{i+1} \leq \Delta^{1+i/h}$

1: $\epsilon := 1/2 + \frac{2h+\sqrt{h}+2}{h^{3/2}}$
2: $col(v) :=$ Compute an $\mathrm{O}(\Delta^2)$ coloring in the graph G using [19]
3: **while** (v is active) $\wedge$ ($d_+^{act}(v) \geq \Delta^\epsilon$) **do**
4: Compute a $(2, \lfloor\sqrt{h}\rfloor)$-ruling set R
5: **if** $v \in R$ **then** Join MIS **end if**
6: **end while**
7: Compute MIS using Algorithm [6] if v is active

Correctness and time complexity for Algorithm *RulingMIS* are proven analogously to Algorithm *RulingColoring*.

Theorem 8. *Algorithm* RulingMIS *computes a MIS.*

Proof. By definition of a $(2, c)$-ruling set any two nodes have distance 2. Since nodes in the ruling set join the MIS and their neighbors become inactive, all nodes in the MIS must have distance at least 2, i.e., are independent. Nodes only become inactive if they join the MIS or have a neighbor in the MIS. Thus, once a node becomes inactive, it is either in the MIS or has a neighbor in the MIS. For the remaining nodes a MIS is computed correctly due to Theorem 4.6 in [6]. □

Theorem 9. *Algorithm* RulingMIS *has time complexity* $O(\Delta^{1/2+\Theta(1/\sqrt{h})} + \log^* n)$ *for* $\Delta_{i+1} \leq \Delta^{1+i/h}$. *It is sublinear for* $h \geq 16$.

The proof is analogous to the proof of Theorem 7.

Proof. Correctness follows by Theorem 8. Since we compute ruling sets for all active nodes v

with $d_+^{act}(u) \geq \Delta^\epsilon$, any node joining the ruling set (and thus the MIS) renders $\Delta^\epsilon + 1$ nodes inactive. Any active node (participating in the computation of a ruling set) either joins the ruling set or gets a neighbor within distance c that is in the ruling set. The total time until node v becomes inactive or has less than Δ^ϵ active neighbors is given by the following terms: The time it takes to compute a $O(|N(v)|^2)$ coloring using [19], i.e., $O(\log^* n)$. In addition, the total number of nodes $|N^{c+1}(v)| \leq \Delta_{c+1}$ within distance $c+1$ divided by the number of nodes that get colored for a computation of a ruling set, i.e., at least $\Delta^\epsilon + 1$, multiplied by the time it takes to color the nodes and to compute one ruling set, i.e., $O(c \cdot \Delta^{2/c})$ (see Theorem 4). To compute a MIS for the remaining nodes, i.e., at most Δ^ϵ neighbors of each node, using [6] (or [17]) takes time $O(\Delta^\epsilon + \log^* n)$. Thus, overall we get time $O(c\Delta_{c+1}/\Delta^\epsilon \cdot \Delta^{2/c} + \Delta^\epsilon + \log^* n)$. This term is minimized for $c\Delta_{c+1} \cdot \Delta^{2/c}/\Delta^\epsilon = \Delta^\epsilon$. Neglecting the factor c, using $\Delta_{i+1} = \Delta^{1+i/h}$ and taking the logarithm, we get $\epsilon = 1/2 + \frac{c^2+2h}{2ch}$. Clearly, the larger h the smaller the expansion and thus the faster our MIS algorithm runs. Thererfore, in case h is large, i.e., $h \notin O(1)$, the time complexity of our algorithm only improves compared to h being constant. Therefore, assume $h \in O(1)$. Setting parameter $c := \sqrt{h}$ yields $\epsilon = 1/2 + \frac{3h}{2h^{3/2}}$. The time complexity becomes $O(\Delta^{1/2+\Theta(1/\sqrt{h})} + \log^* n)$, which is sublinear in Δ for $h > 9$. But since $c := \sqrt{h}$ must be an integer we get sublinearity for $c \geq 4$, i.e. $h \geq 16$. □

To improve upon the constants one might compute a different initial coloring (in the same manner as explained for Algorithm *RulingColoring*. Algorithm *RulingMIS* only requires messages of size $O(\log n)$.

7 Coloring Depending on the Chromatic Number

The algorithm (but not the analysis) itself is straightforward without many novel ideas. In the first two rounds a node attempts to get a color from a set with less than Δ colors. Then, (essentially) coloring algorithms from [24] are used to color the remaining nodes.

Let $\Delta_0 := \Delta$ be the maximal size of a neighborhood in the graph, where all nodes are uncolored, and let $N_0(v)$ be the neighbors of node v upon the start of the algorithm. Let $N(v)$ be all uncolored neighbors of node v in the current iteration. The algorithm lets an uncolored node v be active twice with a fixed constant probability $1/c_1$. An active node chooses a random color from all available colors in the interval $[0, \Delta_0/2 - 1]$. Node v obtains its chosen color and exits the algorithm, if none of its neighbors $N(v)$ has chosen the same color. After the initial two attempts to get colored each node v computes the set of all colors $C^1_{N(v)}$ that have been colored by neighbors $N(v)$ in iterations 0 and 1 and how many neighbors $d(v)$ are left to color. The number of "conserved" (or saved) colors $s(v)$ (compared to a $\Delta + 1$ coloring) is given by the difference $s(v) := \Delta_0 - d(v) + |C^1_{N(v)}|$. The algorithm can use the conserved colors to either speed up the running time, since more available colors render the problem simpler, e.g., allow for easier symmetry breaking, or to reduce the number of used colors as much as possible. In Algorithm *FastRandColoring* we spend half of the conserved colors for fast execution and preserve the other half to compute a coloring using $\Delta_0 + 1 - s(v)/2$ colors. A node v repeatedly chooses uniformly at random an available color from $[0, \Delta_0 + 1 - s(v)/2]$ using Algorithm *Delta-*

Plus1Coloring[24] until the number of available colors is at least a factor two larger than the number of uncolored neighbors. Afterwards it executes Algorithm *ConstantDeltaColoring* [24] using 2Δ colors.

Algorithm *FastRandColoring*

1: $col(v) := none$
2: $J(v) := [0, \Delta_0/2 - 1]$
3: **for** $i = 0..1$ **do**
4: $choice(v)$:= With probability $1/c_1$ random color from $J(v)$ else *none*
5: **if** $choice(v) \neq none \wedge \nexists u \in N(v), s.t. choice(u) = choice(v) \vee col(u) = choice(v)$ **then** $col(v) := choice(v)$ and exit **end if**
6: **end for**
7: $C^1_{N(v)} := \{col(u) | u \in N_0(v)\} \setminus none$
8: $s(v) := \Delta_0 - d(v) - |C^1_{N(v)}|$
9: $H(v) := [0, \Delta_0 + 1 - s(v)/2] \setminus C^1_{N(v)}$ {available colors}
10: Execute Algorithm DeltaPlus1Coloring [24] using colors $H(v)$ until $|H(v)| \geq 2d(v)$
11: Execute Algorithm ConstDeltaColoring [24] using colors $H(v)$

We start by giving an outline of the proof. We consider an optimal coloring using χ colors. Let S_c be all nodes with color $c \in [0, \chi - 1]$. Note that any two nodes $u, v \in S_c$ are independent since they obtain the same color c for an optimal coloring. Using Algorithm *FastRandColoring* we prove that for a node w many of its neighbors, i.e. pairs in $u, v \in (S_c \cap N(w))$ get the same color. First, we show (Theorem 11) that after the first possibility of obtaining a color, the number of colored neighbors of a node v is within certain bounds. In the following Theorem 12 we consider an uncolored neighbor $u \in S_c$ of v. We show that for the second possibility of obtaining a color a constant fraction of all colors taken by nodes $N(v) \cap S_c$ at the first possibility are not chosen by any neighbor $y \in N(u)$ or have been obtained by a node $y \in N(u)$ before.. This implies that a neighbor u has a "good" (quantified in Theorem 14) chance to get a color that has been chosen by a node $x \in (N(v) \cap S_c)$. Every such pair of nodes $u, x \in (N(v) \cap S_c)$ with the same color increases the number of "conserved" colors by 1.

For the analysis we use the following observation: If an event occurs with high probability then conditioning on the fact that the event actually took place does not alter the probability of other likely events much as shown in the next theorem. It follows directly from the union-bound.

Theorem 10. *For n^{k_0} (dependent) events E_i with $i \in \{0, ..., n^{k_0} - 1\}$ and constant k_0, such that each event E_i occurs with probability $Pr(E_i) \geq 1 - 1/n^k$ for $k > k_0$, the probability that all events occur is at least $1 - 1/n^{k-k_0}$.*

Proof. For each E_i, $Pr(\text{not } E_i) \leq 1/n^k$. Hence $Pr(\exists i \text{ s.t. not } E_i) \leq (1/n^k) \cdot n^{k_0} = 1/n^{k-k_0}$. Therefore, $Pr(\text{all } E_i \text{ occur}) \geq 1 - 1/n^{k-k_0}$. □

Consider any coloring of the graph G using the minimal number of colors χ. Let S_c be a set of nodes having color $c \in [0, \chi - 1]$ for this coloring. For a node v with color c for this optimal coloring, we have $v \in S_c$. Let choice $i \geq 0$ (of colors) be the $(i + 1)$-st possibility where a node could have chosen a color, i.e., iteration i of the for-loop of Algorithm *FastRandColoring*. Let the set C^i_S be all distinct colors that have been obtained by a set of nodes S for any choice $j \leq i$, i.e., $C^i_S := \{c | \exists u \in S, \text{ s.t. } col(u) =$

c after iteration $i\}$. We do not use multisets here, i.e., a color c can only occur once in C_S^i. Let P_S^i be all nodes in S that make a choice in iteration i, i.e., $P_S^i := \{c|\exists u \in S$, s.t. $choice(u) = c$ in iteration $i\}$. Let CP_S^i be all colors that have been chosen (but not yet obtained) by a set of nodes S in iteration i, i.e., $CP_S^i := \{c|\exists u \in P_S^i$, s.t. $choice(u) = c$ in iteration $i\}$. By definition, $|CP_S^i| \leq |P_S^i|$.

To deal with the interdependence of nodes we follow the idea of *stochastic domination*. If X is a sum of random binary variables $X_i \in \{0,1\}$, i.e., $X := \sum_i X_i$, with probability distributions A, B and $Pr_A(X_i = 1|X_0 = x_0, X_1 = x_1, ..., X_{i-1} = x_{i-1}) \geq Pr_B(X_i = 1|X_0 = x_0, X_1 = x_1, ..., X_{i-1} = x_{i-1}) = p$ for any values of $x_0, x_1, ..., x_{i-1}$, we can apply a Chernoff bound to lower bound $Pr_A(X \geq x)$ by a sum of independent random variables X_i, where $X_i = 1$ with probability p.

Theorem 11. *After choice $i \in [0,1]$ for every node v holds w.h.p.: The colored nodes C_S^1 of any set $S \in \{S_c \cap N_0(v)|c \in [0, \chi - 1]\}$ or $S \in \{N(v), N_0(v)\}$ with $|S| \geq c_2 \log n$ fulfill $|C_S^1| \in [|S|/(16c_1), 3|S|/c_1]$ with $c_1 > 32$. The number of nodes $|P_S^1|$ making a choice is at least $|S|/(4c_1)$ and at most $3|S|/(2c_1)$.*

Proof. Consider such a set S of nodes for some node v. For i possibilities to make a choice we expect (up to) $i|S|/c_1$ nodes to actually make a choice. Using the Chernoff bound from Corollary 2 the number of nodes that choose a color deviates by no more than one half of the expectation with probability $1-2^{i/8|S|/c_1} \geq 1-2^{i/8c_2 \log n/c_1} = 1 - 1/n^{c_3}$ for a constant $c_3 := ic_2/(8c_1)$. Thus, at most $3i|S|/(2c_1)$ neighbors of v make a choice and potentially get colored with probability $1 - 1/n^{c_3}$. Using Theorem 10 this holds for all nodes with probability $1 - 1/n^{c_3-3}$, which yields the bounds $|P_S^1| \leq 3|S|/(2c_1)$ and $|C_S^1| \leq 3|S|/c_1$.

For choice i w.h.p. the number of nodes that make a choice is therefore in $[a, b] := [1/2 \cdot (1 - 3i/(2c_1)) \cdot |S|/c_1, 3|S|/(2c_1)]$ w.h.p.. The lower bound, i.e., $a \leq |P_S^i|$, follows if we assume that for each choice $j < i$ at most $3|S|/(2c_1)$ nodes get colored, which happens w.h.p.. Thus, after $i - 1$ choices at least $(1 - 3i/(2c_1)) \cdot |S|$ nodes can make a choice, i.e., are still uncolored. We expect a fraction of $1/c_1$ of them to choose a color. Using Corollary 2 the number of nodes that make a choice is at least half the expected number w.h.p.. Thus, for choice 1 we have for $c_1 > 32$ and $a := (1 - 3/(2c_1))/(2c_1) \cdot |S|$ the following: $|S|/(4c_1) \leq a \leq |P_S^1|$.

Consider an arbitrary order $w_0, w_1, ..., w_{|S|-1}$ of nodes S. We compute the probability that node $w_k \in S$ obtains a distinct color for choice i from all previous nodes $w_0, w_1, ..., w_{k-1} \in S$. The probability is minimized, if all $k - 1$ nodes have distinct colors and k is large. Since $k \leq b = 3|S|/(2c_1)$ we have $p(col(w_k) \in [0, \Delta_0/2] \setminus S_{w_0,w_1,...,w_{k-1}}) \geq p(col(w_k) \in [0, \Delta_0/2] \setminus S_{w_0,w_1,...,w_{b-1}}) \geq 1/c_1 \cdot (1 - b/(\Delta_0/2)) \geq (1 - 3/\Delta_0/(2c_1)/(\Delta_0/2))/c_1 = 1/c_4$ with constant $c_4 := 1/c_1 \cdot (1 - 3/c_1)$. The lower bound for the probability of $1/c_4$ holds for any $k \in [0, b-1]$ and any outcome for nodes $S_{w_0,w_1,...,w_{k-1}}$. Thus, to lower bound the number of distinct colors $|C_S|$ that are obtained by nodes in S we assume that the number of nodes that make a choice is only a and that each node that makes a choice gets a color with probability $1/c_4$ (independent of the choices of all other nodes). Using the Chernoff bound from Corollary 2 gives the desired result for a set S. In total there are n nodes and we have to consider at most $1 + \chi \leq 1 + n$ sets per node. Using Theorem 10 for $n \cdot (n + 1)$ events each occurring w.h.p. completes the proof. □

Next we consider a node v and prove that for the second attempt of all uncolored nodes $u \in (S_c \cap N(v))$ a constant fraction of colors taken by independent nodes $w \in (S_c \cap N(v) \setminus \{u\})$ from u are not taken (or chosen) by its neighbors $y \in N(u)$.

Theorem 12. *For the second choice let $E(c)$ be the event that for a node v for each uncolored node $u \in (N_0(v) \cap S_c)$ holds $|(CP^1_{N_0(u)} \cup C^1_{N_0(u)}) \cap C^1_{N_0(v)\cap S_c}| \leq 3/4|C_{N_0(v)\cap S_c}|$ for $|N(v) \cap S_c| \geq c_2 \log n$. Event $E(c)$ occurs w.h.p.*

Proof. Consider a colored node $w \in S_c \cap N_0(v)$ for some node v. We compute an upper bound on the probability that an uncolored node $y \in N(u)$ gets (or chooses) color $col(w)$, i.e., $p(\exists y \in N(u), col(y) = col(w) \vee choice(y) = col(w)) = p(\vee_{y\in N(u)} col(y) = col(w) \vee choice(y) = col(w)) \leq \sum_{y\in N(u)} p(col(y) = col(w) \vee choice(y) = col(w))$. The latter inequality follows from the inclusion-exclusion principle: For two events A, B we have $p(A \cup B) = p(A) + p(B) - p(A \cap B) \leq p(A) + p(B)$. We consider the worst case topology and worst case order in which nodes make their choices to maximize $\sum_{y\in N(u)} p(col(y) = col(w) \vee choice(y) = col(w))$. Due to Theorem 11 for every uncolored node $y \in N(u)$ at most $|P^0_{N_0(y)}| + |P^1_{N_0(y)}| \leq 3d(y)/c_1 \leq 3\Delta_0/c_1$ neighbors $z \in N_0(y)$ make a choice during the first two attempts $i \in [0, 1]$. To maximize the chance that some node y obtains (or chooses) color $col(w)$, we can minimize the number of available colors for y and the probability that some neighbor $z \in N_0(y)$ chooses color $col(w)$, since when making choice i we have $p(choice(y) = col(w)) \leq 1/(c_1|J(y)|)$ because each available color in $J(y)$ is chosen with the same probability. To minimize $|J(y)|$ the number of colored nodes $z \in N_0(y)$ should be maximized and at the same time each node $z \in N_0(y)$ should have a neighbor itself with color $col(w)$. The latter holds, if $z \in N_0(y)$ is adjacent to node w. Thus, to upper bound $p(col(y) = col(w))$ we assume that node w and each node $y \in N(u)$ share the same neighborhood (except u), i.e., $N_0(y) \setminus \{u\} = N_0(w)$, and the maximal number of nodes in $N_0(y)$ given our initial assumption are colored or make a choice, i.e., $3d_0(y)/c_1 \leq 3\Delta_0/c_1$. This, yields $p(col(y) = col(w)) \leq 1/(c_1|J(y)|) \leq 1/(c_1(\Delta_0/2 - 3\Delta_0/c_1)) \leq 8/(c_1\Delta_0)$ (for $c_1 > 32$) and therefore $p(\exists y \in N(u), col(y) = col(w)) = p(\vee_{y\in N(u)} col(y) = col(w)) \leq \sum_{y\in N(u)} p(col(y) = col(w)) \leq 3\Delta_0/c_1 \cdot 8/(c_1\Delta_0) \leq 1/c_1$ (for $c_1 > 32$). In other words, the probability that some uncolored node $y \in N(u)$ has obtained color $col(w)$ or chooses $col(w)$ is bounded by $1/c_1$.

Let us estimate the probability that some neighbor $y \in N_0(u)$ gets the same color as a node $w_1 \in N_0(v) \cap S_c$ given that some nodes $z \in N_0(u)$ have chosen or obtained $col(w_0)$ for some node $w_0 \in C_{N_0(v)\cap S_c} \setminus \{w_1\}$. To minimize $|J(y)|$ we assume that $|J(y)|$ is reduced by 1 for every colored node $w_0 \in C_{N_0(v)\cap S_c} \setminus \{w_1\}$. Since at most $3/2d_0(y)/c_1 \leq 3/2\Delta_0/c_1$ neighbors make a choice concurrently, the event reduces the size of $|J(y)|$ by at most $3/2\Delta_0/c_1$. Using the same calculations as above with $|J(y)| \leq \Delta_0/2 - 9/2\Delta_0/c_1$, the probability that some node $y \in N_0(u)$ has obtained color $col(w)$ or chooses $col(w)$ given the outcome for any set of colored nodes $W \subseteq N_0(v) \cap S_c$ is at most $1/2$. Thus, we expect at most $|C_{N_0(v)\cap S_c}|/2$ colors from $C_{N_0(v)\cap S_c}$ to occur in node u's neighborhood. Using the Chernoff bound from Corollary 2, we get that the deviation is at most $1/2$ the expectation with probability $1 - 2^{-|C_{N_0(v)\cap S_c}|/8}$ for node u, i.e., the probability $p(E(u, c))$ of the event $E(u, c)$ that for a node

$u \in N(v)$ at most $3|C_{N_0(v)\cap S_c}|/4$ colors from $N_0(v) \cap S_c$ are also taken or chosen by its neighbors $y \in N_0(u)$ is at least $1 - 2^{-|C_{N(v)\cap S_c}|/8}$. Using Theorem 11 for $S = N_0(v) \cap S_c$ we have $|C_{N_0(v)\cap S_c}| \geq |S|/(16c_1) = |N_0(v) \cap S_c|/(16c_1) \geq c_2 \log n/(16c_1)$. Therefore, $p(E(u,c)) \geq 1 - 1/n^{c_2/(16c_1)}$. Due to Theorem 10 the event $E(c) := \bigwedge_{u\in N_0(v)} E(u,c)$ occurs w.h.p.. □

Using Theorem 12 together with Theorem 10 yields the following corollary.

Corollary 13. *An event $E(c)$ (as defined in Theorem 12) occurs w.h.p. given $\bigwedge_{c_1\in X\subseteq[0,\chi],|N(v)\cap S_{c_1}|\geq c_2 \log n} E(c_1)$ for an arbitrary set $X \subseteq [0,\chi]$*

Theorem 14. *After the first two choices for a node v with initial degree $d_0(v) \geq \Delta_0/2$ there exists a subset $N_c \subseteq N_0(v)$ with $|N_c| \geq (\Delta+1)/(c_5\chi)$ that has been colored with $(\Delta+1)/(2c_5\chi)$ colors for a constant c_5 w.h.p. for $\Delta \in \Omega(\log^{1+1/\log^* n} n)$ and $\chi \in O(\Delta/\log n)$.*

Proof. By assumption $\chi \in O(\Delta/\log n)$, i.e., $\chi < 1/(4c_3)\Delta/\log n$. At least half of all neighbors $u \in N_0(v)$ with $u \in S_c \cap N_0(v)$ must be in sets $|S_c \cap N_0(v)| \geq c_3 \log n$. This follows, since the maximum number of nodes in sets $|S_c \cap N_0(v)| < c_3 \log n$ is bounded by $\chi \cdot c_3 \log n \leq \Delta_0/4$. Assume that all statements of Theorem 11 that happen w.h.p. have actually taken place. Consider a node v and a set $N_0(v)\cap S_c$ with $|S_c \cap N_0(v)| \geq c_3 \log n$ given there are at most $3/4 d_0(v) \leq 3/4\Delta_0$ colored neighbors $u \in N_0(v)$. For a node $u \in N_0(v) \cap S_c$ the probability that it obtains the same color of another node $N_0(v) \setminus \{u\} \cap S_c$ is given by the probability that it chooses a color $col(w)$ taken by node $w \in N_0(v) \setminus \{u\} \cap S_c$ that is not chosen by any of u's neighbors $x \in N_0(u)$. Due to Corollary 13 $|C_{N_0(v)\cap S_c}|/4$ colors exist that are taken by some node $w \in N_0(v) \cap S_c$ but not taken (or chosen for the second choice) by a neighbor $x \in N_0(u)$. Due to Theorem 11 we have $|C_{N_0(v)\cap S_c}|/4 \geq |N_0(v) \cap S_c|/(64c_1)$. Additionally, the theorem yields $|P^1_{N_0(v)\cap S_c}| \geq |N_0(v) \cap S_c|/(4c_1)$.

The probability for a node $u \in P^1_{N_0(v)\cap S_c}$ to obtain (not only choose) a color in $C_{N_0(v)\cap S_c}$ becomes the number of "good" colors, i.e., $|N_0(v)\cap S_c|/(64c_1)$, divided by the total number of available colors, i.e., $1/(\Delta_0/2)$, yielding $|N_0(v) \cap S_c|/(32c_1 \cdot \Delta_0)$. This holds irrespectively of the behavior of other nodes $w \in P^1_{N_0(v)\cap S_c}$ and $w \in N_0(v) \cap S_d$ with $d \in [0,\chi-1] \setminus \{c\}$. The reason is that a node u makes its decision what color to choose independently of its neighbors $y \in N_0(u)$ and Theorem 11 and Corollary 13 already account for the worst case behavior of neighbors $y \in N_0(u)$ to bound the probability that node u gets a chosen color.

Thus, for a set of $|P^1_{N_0(v)\cap S_c}| \geq |N_0(v) \cap S_c|/(4c_1)$ nodes we expect that for at least $|N_0(v)\cap S_c|^2/(128c_1^2\cdot\Delta_0)$ nodes u there exists another node $w \in (N_0(v)\cap S_c) \setminus \{u\}$ with the same color. The expectation $|N_0(v) \cap S_c|^2/(128c_1^2 \cdot \Delta_0)$ is minimized if all sets $|N_0(v) \cap S_c| \geq c_3 \log n$ are of equal size and as small as possible, i.e., $\Delta_0/(4\chi)$ since at least $\Delta_0/4$ nodes are in sets $|N_0(v) \cap S_c| \geq c_3 \log n$ for some c. This gives $\sum_{c\in[0,\chi-1]} |N_0(v)\cap S_c|^2/(128c_1^2\cdot\Delta_0) \geq \sum_{c\in[0,\chi-1]} (\Delta_0)^2/(2048c_1^2 \cdot \Delta_0 \cdot \chi^2) = \Delta_0/(c_5 \cdot \chi)$ for $c_5 := 2048c_1^2$. Since by assumption $\chi \in O(\Delta_0/\log n)$ using Corollary 2 the actual number deviates by at most 1/2 of its expectation w.h.p.. Therefore, for at least $\Delta_0/(c_5 \cdot \chi)$ nodes $u \in N_0(v) \cap S_c$ there exists another node $w \in N_0(v) \setminus \{u\} \cap S_c$ with the same color. Thus, to color all of these $\Delta_0/(c_5 \cdot \chi)$ nodes only $\Delta_0/(2c_5 \cdot \chi)$ colors are used. □

Theorem 15. *If* $\Delta \in \Omega(\log^{1+1/\log^* n} n)$ *and* $\chi \in O(\Delta/\log^{1+1/\log^* n} n)$ *then Algorithm* FastRandColoring *computes a* $(1-1/O(\chi))\Delta$ *coloring in time* $O(\log\chi + \log^* n)$ *w.h.p..*

Proof. Extending Theorem 14 to all nodes using Theorem 10 we have w.h.p. that each node v with $d_0(v) \geq \Delta_0/2$ has at most $(\Delta_0+1)\cdot(1-1/(c_5\chi))$ uncolored neighbors after the first two choices. However, node v is allowed to use $d(v)+1$ colors and, additionally, half of the conserved colors, i.e., $s(v)/2 = \Delta_0/(8c_2\chi) \geq \log^{1+1/\log^* n} n/(4c_5)$ (see Theorem 14), to get a color itself. When executing Algorithm *Delta-Plus1Coloring* [24] the maximum degree is reduced by a factor 2 in $O(1)$ rounds as long as it is larger than $\Omega(\log n)$ due to Theorem 8 in [24]. The time until the maximum degree Δ is less than $s(v)/4$ is given by $O(\log\Delta_0 - \log s(v)) = O(\log\Delta_0 - \log(\Delta_0/(32c_2\chi))) = O(\log\chi)$. Thus, we have at least 2Δ colors available, i.e., at least $\log^{1+1/\log^* n} n/(4c_5)$ additional colors, when calling Algorithm *ConstDeltaColoring*[24]. Therefore, the remaining nodes are colored in time $O(\log^* n)$ using Corollary 14 [24]. Nodes with initial degree $d_0(v) < \Delta_0/2$ can use $|H(v)| := \Delta_0 + 1 - s(v)/2 \geq \Delta_0 + 1 - d_0(v)/2 \geq \Delta_0/2 \in \Omega(\log^{1+1/\log^* n} n)$ (since one cannot save more colors $s(v)$ than there are neighbors $d_0(v)$, i.e $s(v) \leq d_0(v)$) colors to color $d(v)+1$ nodes. After the first two attempts to get a color we have $d(v) \leq d_0(v) - 2s(v) < \Delta_0/2 - 2s(v) \leq (\Delta_0 + 1 - s(v)/2)/2 = |H(v)|/2$. The first inequality follows since any saved color c implies that at least two neighbors $u, w \in N_0(v)$ are colored with the same color c. Thus, as for the case $d_0(v) \geq \Delta_0/2$ there are at least $2\cdot d(v) \leq |H(v)|$ colors with $|H(v)| \in \Omega(\log^{1+1/\log^* n} n)$ available to color $d(v)$ nodes. □

8 Conclusion

It is still an open problem, whether deterministic $\Delta+1$ coloring in a general graph is possible in time $\Delta^{1-\epsilon} + \log^* n$ for a constant ϵ. Our algorithm indicates that this might well be the case, since we broke the bound for a wide class of graphs.

Though it is hard in a distributed setting – and sometimes not even possible – to use less than $\Delta+1$ colors, we feel that one should also keep an eye on the original definition of the coloring problem in a distributed environment: Color a graph with as little colors as possible. To strive for a $\Delta+1$ coloring is of much appeal and gives interesting insights but as we have shown (in many cases) better bounds regarding the number of used colors and the required time complexity can be achieved by taking the chromatic number of the graph into account.

References

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

[2] S. Arora and E. Chlamtac. New approximation guarantee for chromatic number. In *Symp. on Theory of computing (STOC)*, 2006.

[3] B. Awerbuch. Complexity of network synchronization. *J. ACM*, 32(4), 1985.

[4] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network decomposition and locality in distributed computation. In *Symp. on Foundations of Computer Science (FOCS)*, 1989.

[5] L. Barenboim and M. Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using nash-williams decomposition. In *PODC*, 2008.

[6] L. Barenboim and M. Elkin. Distributed (δ + 1)-coloring in linear (in δ) time. In *Symp. on Theory of computing (STOC)*, 2009.

[7] L. Barenboim and M. Elkin. Deterministic distributed vertex coloring in polylogarithmic time. In *Symp. on Principles of distributed computing (PODC)*, 2010.

[8] A. Blum. New approximation algorithms for graph coloring. *Journal of the ACM*, 41:470–516, 1994.

[9] B. Bollobas. Chromatic nubmer, girth and maximal degree. *Discrete Math.*, 24:311–314, 1978.

[10] C. Gavoille, R. Klasing, A. Kosowski, L. Kuszner, and A. Navarra. On the complexity of distributed graph coloring with local minimality constraints. *Networks*, 54(1), 2009.

[11] B. Gfeller and E. Vicari. A Randomized Distributed Algorithm for the Maximal Independent Set Problem in Growth-Bounded Graphs. In *Sy. on Principles of Distributed Computing*, 2007.

[12] D. A. Grable and A. Panconesi. Fast distributed algorithms for Brooks-Vizing colorings. *J. Algorithms*, 37(1):85–120, 2000.

[13] M. M. Halldórsson and J. Radhakrishnan. Greed is good: approximating independent sets in sparse and bounded-degree graphs. In *STOC*, 1994.

[14] Ö. Johansson. Simple distributed Δ+1-coloring of graphs. volume 70, 1999.

[15] M. Karchmer and J. Naor. A fast parallel algorithm to color a graph with delta colors. *J. Algorithms*, 9(1):83–91, 1988.

[16] A. Korman, J.-S. Sereni, and L. Viennot. Toward more localized local algorithms: removing assumptions concerning global knowledge. In *PODC*, 2011.

[17] F. Kuhn. Weak Graph Coloring: Distributed Algorithms and Applications. In *Symp. on Parallelism in Algorithms and Architectures (SPAA)*, 2009.

[18] F. Kuhn and R. Wattenhofer. On the Complexity of Distributed Graph Coloring. In *Symp. on Principles of Distributed Computing (PODC)*, 2006.

[19] N. Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[20] M. Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.

[21] G. D. Marco and A. Pelc. Fast distributed graph coloring with O(Δ^2) colors. In *SODA*, 2001.

[22] A. Panconesi and A. Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Symp. on Theory of computing (STOC)*, 1992.

[23] J. Schneider and R. Wattenhofer. A Log-Star Distributed Maximal Independent Set Algorithm for Growth-Bounded Graphs. In *Symp. on Principles of Distributed Computing (PODC)*, 2008.

[24] J. Schneider and R. Wattenhofer. A New Technique For Distributed Symmetry Breaking. In *Symp. on Principles of Distributed Computing (PODC)*, 2010.

[25] J. Schneider and R. Wattenhofer. Distributed Coloring Depending on the Chromatic Number or the Neighborhood Growth. In *SIROCCO*, 2011.

[26] J. Schneider and R. Wattenhofer. Trading Bit, Message, and Time Complexity of Distributed Algorithms. In *Int. Symp. on Distributed Computing (DISC)*, 2011.