Improved Distributed Fractional Coloring Algorithms

Alkida Balliu
Gran Sasso Science Institute, L'Aquila, Italy

Fabian Kuhn
University of Freiburg, Germany

Dennis Olivetti
Gran Sasso Science Institute, L'Aquila, Italy

Abstract

We prove new bounds on the distributed fractional coloring problem in the LOCAL model. A fractional c-coloring of a graph $G = (V, E)$ is a fractional covering of the nodes of G with independent sets such that each independent set I of G is assigned a fractional value $\lambda_I \in [0, 1]$. The total value of all independent sets of G is at most c, and for each node $v \in V$, the total value of all independent sets containing v is at least 1. Equivalently, fractional c-colorings can also be understood as multicolorings as follows. For some natural numbers p and q such that $p/q \leq c$, each node v is assigned a set of at least q colors from $\{1, \ldots, p\}$ such that adjacent nodes are assigned disjoint sets of colors. The minimum c for which a fractional c-coloring of a graph G exists is called the fractional chromatic number $\chi_f(G)$ of G.

Recently, [Bousquet, Esperet, and Pirot; SIROCCO '21] showed that for any constant $\varepsilon > 0$, a fractional $(\Delta + \varepsilon)$-coloring can be computed in $\Delta^{O(\Delta)} + O(\Delta \cdot \log^* n)$ rounds. We show that such a coloring can be computed in only $O(\log^2 \Delta)$ rounds, without any dependency on n.

We further show that in $O\big(\frac{\log n}{\varepsilon}\big)$ rounds, it is possible to compute a fractional $(1 + \varepsilon)\chi_f(G)$-coloring, even if the fractional chromatic number $\chi_f(G)$ is not known. That is, the fractional coloring problem can be approximated arbitrarily well by an efficient algorithm in the LOCAL model. For the standard coloring problem, it is only known that an $O\big(\frac{\log n}{\log \log n}\big)$-approximation can be computed in polylogarithmic time in the LOCAL model. We also show that our distributed fractional coloring approximation algorithm is best possible. We show that in trees, which have fractional chromatic number 2, computing a fractional $(2 + \varepsilon)$-coloring requires at least $\Omega\big(\frac{\log n}{\varepsilon}\big)$ rounds.

We finally study fractional colorings of regular grids. In [Bousquet, Esperet, and Pirot; SIROCCO '21], it is shown that in regular grids of bounded dimension, a fractional $(2 + \varepsilon)$-coloring can be computed in time $O(\log^* n)$. We show that such a coloring can even be computed in $O(1)$ rounds in the LOCAL model.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases distributed graph algorithms, distributed coloring, locality, fractional coloring

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2021.18

1 Introduction & Related Work

The distributed graph coloring problem is at the heart of the area of distributed graph algorithms and it is one of the prototypical problems to study distributed symmetry breaking. Already in [31], Linial showed that deterministically coloring a ring network with $O(1)$ colors and thus more generally coloring n-node graphs of maximum degree Δ with a number of colors that only depends on Δ requires $\Omega(\log^* n)$ rounds. In [39], Naor extended this lower bound to randomized algorithms. Subsequently, over the last three decades, the distributed coloring problem has been studied intensively and we now have a quite good understanding of the complexity of the problem. Mostly, researchers focused on the problem of computing a coloring with $\Delta + 1$ colors, i.e., with the number of colors that can be

obtained by a simple sequential greedy algorithm. In light of the $\Omega(\log^* n)$ lower bounds of [31, 39], there is a long line of research to (deterministically) solve $(\Delta+1)$-coloring in time $f(\Delta)+O(\log^* n)$ for some function f (see, e.g., [5, 8, 9, 17, 21, 29, 36, 45]), where the current best bound of $O(\sqrt{\Delta\log\Delta}+\log^* n)$ was proven in [8, 17, 36]. The complexity of distributed $(\Delta+1)$-coloring has also been studied as a function of n. The randomized complexity has been known to be $O(\log n)$ since the 1980s [1, 25, 31, 35] and it has recently been improved to $O(\log^3\log n)$ [10, 13, 20, 23] and even to $O(\log^* n)$ for graphs of maximum degree $\Delta\geq\log^{2+\varepsilon} n$ [22]. For a long time, the best deterministic $(\Delta+1)$-coloring algorithms had a complexity of $2^{O(\sqrt{\log n})}$ [3, 28, 40] and it was only recently shown in a breakthrough paper by Rozhoň and Ghaffari [42] that the distributed $(\Delta+1)$-coloring problem (and many other distributed graph problems) can be solved in time $\operatorname{poly}\log n$ deterministically. Subsequently, the deterministic complexity of the distributed $(\Delta+1)$-coloring problem has even been improved to $O(\log^2\Delta\cdot\log n)$ [20].

While much of the existing work on distributed vertex coloring is on the $(\Delta+1)$-coloring problem, it is of course also relevant to understand the complexity of more restrictive or more relaxed variants of the problem, for example by considering vertex colorings with more or fewer colors. Already in [31], Linial showed that an $O(\Delta^2)$-coloring can be computed deterministically in time only $O(\log^* n)$. Over the years, there were several papers that considered distributed coloring algorithms to color graphs with at least $\Delta+1$ colors (e.g., [6, 9, 15, 26, 28, 37, 44]). One however needs to use $\omega(\Delta)$ colors to obtain significantly faster distributed coloring algorithms. Colorings with less than $\Delta+1$ colors however require significantly more time. In [31], Linial shows that even on Δ-regular trees, computing an $O(\sqrt{\Delta})$-coloring requires $\Omega(\log_\Delta n)$ rounds deterministically.[1] This was improved in [12], where it is shown that even computing a Δ-coloring of Δ-regular trees requires $\Omega(\log_\Delta n)$ rounds deterministically and $\Omega(\log_\Delta\log n)$ rounds with randomization. There are algorithms that nearly match those bounds [19, 20, 41]. Further, by using network decompositions [3, 14, 33, 40, 42], it is possible to efficiently approximate the best possible vertex coloring [7].[2] In particular, in $\operatorname{poly}\log n$ rounds, it is possible to compute a coloring of a graph G with $O\big(\frac{\log n}{\log\log n}\big)\cdot\chi(G)$ colors.

Another natural relaxation of the vertex coloring problem is the fractional coloring problem. A c-coloring of the nodes V of a graph $G=(V,E)$ can be seen as a partition of V into c independent sets. A fractional c-coloring is an assignment of positive weights λ_I to the independent sets I of G such that for every node $v\in V$, the total weight of the independent sets that contain v is equal to (at least) 1 and such that the total weight of all independent sets is equal to c. The smallest c for which a graph G has a fractional c-coloring is called the fractional chromatic number $\chi_f(G)$ of G. Alternatively, a fractional coloring can be defined as a multicoloring as follows. For two integer parameters p and q ($p\geq q$), a $(p:q)$-coloring is an assignment of (at least) q colors to each node such that adjacent nodes are assigned disjoint sets of colors and such that the total number of distinct colors is equal to p. A $(p:q)$-coloring directly gives a fractional (p/q)-coloring and for any graph G with fractional chromatic number $\chi_f(G)$, there exists a pair of integers p and q for which a $(p:q)$-coloring of G with $p/q=\chi_f(G)$ exists.

---

[1] Linial uses an explicit construction of regular high-girth graphs with large chromatic number for his lower bound, but he remarks that by using the right probabilistic construction, the lower bound on the number of colors can be improved to $\Omega(\Delta/\log\Delta)$.

[2] This approach exploits the standard distributed communication models, which in particular allow unbounded internal computations at all nodes.

From the distributed computing perspective, we believe that fractional colorings are interesting and relevant for two reasons. First, in some cases, where graph colorings are needed in practice, one can also use a fractional coloring. For example, if G describes the possible conflicts between the wireless transmissions of nodes in a radio network, a c coloring of the nodes of G can be used to obtain a TDMA schedule for the nodes of G. The length of such a schedule is equal to the number of colors c and every node can therefore be active in a $\frac{1}{c}$-fraction of all time slots. A $(p:q)$-coloring of the nodes of G can also directly be used to obtain a TDMA schedule of length p and in which every node is active in q of the time slots (i.e., in all the time slots corresponding to its colors). Each node is therefore active in a q/p-fraction of all time slots. By computing a fractional coloring instead of a standard coloring, we can therefore potentially increase the fraction of active slots for each node and thus also the usage of the communication channel. Further, understanding the complexity of distributed fractional coloring will generally improve our understanding of the complexity of distributed coloring: what parts of the difficulty in computing vertex colorings stem from the fact that we need to assign exactly one color to every node (and that we thus need to break symmetries) and what parts of the difficulties remain if we compute fractional colorings, where we can "average" over a possibly larger total number of colors.

We are aware of three previous publications that studied the distributed fractional coloring problem. In [27], it is shown that in any graph G, a fractional (degree $+1$)-coloring with support $N!$ can be computed in a single deterministic communication round (where N is the number of IDs). That is, there is a 1-round algorithm that computes a multicoloring with $N!$ colors such that every node v gets assigned a set of at least $N!/(\deg_G(v)+1)$ colors. That is, when considering fractional colorings, the standard $(\Delta+1)$-coloring problem can be solved in a single time step. It is further shown that a fractional $(1+\varepsilon)$(degree $+1$)-coloring with support $O(\Delta^2 \log N/\varepsilon^2)$ can also be computed in 1 round. In both bounds, N can be replaced by C if an initial proper C-coloring of G is given. Similar results were also shown in [24]. Very recently, Bousquet, Esperet, and Pirot [11] made some interesting further progress on the distributed fractional coloring problem.

In [11], it is shown that although a fractional $(\Delta+1)$-coloring can be computed in a single communication round, in Δ-regular graphs that do not contain $K_{\Delta+1}$ as a subgraph, the fractional Δ-coloring problem requires time $\Omega(\log_\Delta n)$ deterministically and $\Omega(\log_\Delta \log n)$ rounds with randomization. That is, for the fractional Δ-coloring problem, the same lower bounds as for the standard Δ-coloring problem hold.[3] It is also shown that in graphs that do not contain $(\Delta+1)$-cliques, a $(q\Delta+1:q)$-coloring can be computed in time $O(q^3\Delta^{2q}+q\log^* n)$ deterministically. By setting $q=1/\varepsilon$, this implies that for any $\varepsilon>0$, a fractional $(\Delta+\varepsilon)$-coloring can be computed in time $O\big(\Delta^{O(1/\varepsilon)}+\frac{1}{\varepsilon}\cdot\log^* n\big)$. In addition, the paper shows that, for any constant $\varepsilon>0$ and constant integer $d>0$, in regular d-dimensional grids, it is possible to compute a fractional $(2+\varepsilon)$-coloring in $O(\log^* n)$ rounds. Hence, while in bounded degree graphs, deterministically computing a fractional Δ-coloring requires $\Omega(\log n)$ rounds, one can get arbitrarily close in only $O(\log^* n)$ rounds. Moreover, in graphs from a minor-closed family of graphs and with sufficiently large girth, it is possible to compute a fractional $(2+\varepsilon)$-coloring in $O\big(\frac{\log n}{\varepsilon}\big)$ rounds, for any constant $\varepsilon>0$. Those results in particular imply that in some graphs, fractional colorings that are arbitrarily close to the best such colorings can be computed.

[3] It has to be noted that for the standard Δ-coloring problem, the lower bounds hold in Δ-regular trees, but for the fractional Δ-coloring problem, the known lower bounds (presented in [11]) only hold for graphs in which every node is contained in some K_Δ.

In this paper, we improve on the results of [11] in several ways. We here give an overview over our results, for a detailed statement of the results, we refer to Section 3. First, we improve the time to compute a fractional $(\Delta+\varepsilon)$-coloring. We show that a coloring of the same quality as in [11] (i.e., a $(q\Delta+1:q)$-coloring for $q=1/\varepsilon$) can be computed in time $O\big(\frac{1}{\varepsilon^2}\cdot\sqrt{\Delta}\cdot\operatorname{poly}\log\Delta+\frac{1}{\varepsilon}\cdot\log^* n\big)$ and a slightly worse $(q\Delta:q-1)$-coloring for $q=\Theta(\Delta/\varepsilon)$ can be computed deterministically in $O\big(\log^2\big(\frac{1}{\varepsilon}\big)\cdot\sqrt{\Delta}\operatorname{poly}\log\Delta+\big(1+\log_\Delta\frac{1}{\varepsilon}\big)\cdot\log^* n\big)$ communication rounds. Moreover, if we further increase the total number of colors, a fractional $(\Delta+\varepsilon)$-coloring can even be computed deterministically in time $O\big(\log^2\Delta+\log^2\big(\frac{1}{\varepsilon}\big)+\log^3\big(\frac{1}{\varepsilon}\big)/\log\Delta\big)$, i.e., we can improve the time dependency on Δ to polylogarithmic and we can drop the dependency on the number of nodes n altogether. We further show that the dependency on n can also be removed in the algorithm for fractionally coloring grids. In d-dimensional grids, for constant $\varepsilon>0$ and constant d, a $(2+\varepsilon)$ can be computed in $O(1)$ time. In addition, we study the problem of computing fractional colorings that are arbitrarily close to the best possible fractional colorings. For any $\varepsilon>0$, we show that it is always possible to deterministically compute a fractional $(1+\varepsilon)\cdot\chi_f(G)$-coloring of a graph G in time $O\big(\frac{\log n}{\varepsilon}\big)$ and we show that computing such a fractional coloring on trees requires $\Omega\big(\frac{\log n}{\varepsilon}\big)$ rounds even with randomization. Note that this is in contrast to the standard coloring problem for which we are not aware of a $\operatorname{poly}\log n$-time algorithm that computes an approximation to the minimum vertex coloring problem with an approximation ratio that is better than $O\big(\frac{\log n}{\log\log n}\big)$.

The remainder of this paper is organized as follows. In Section 2 we describe the LOCAL model of distributed computing and we give some useful definitions. Section 3 contains the detailed statements of our contributions. In Section 4 we present some generic results that are then used in our algorithms. Section 5 contains deterministic algorithms for computing a fractional $(\Delta+\varepsilon)$-coloring. In Appendix A we show randomized and deterministic algorithms for computing arbitrarily good approximations of the chromatic number of a graph. Then, in Section 6 we present a lower bound of $\Omega(\log n/\varepsilon)$ rounds for computing a fractional $(2+\varepsilon)$-coloring. Finally, Appendix B contains our constant-time algorithm for fractional $(2+\varepsilon)$-coloring on d-dimensional grids.

2 Model and Definitions

2.1 LOCAL model

The model of computation that we consider is the well-known LOCAL model of distributed computing. A distributed network is modeled as a graph where nodes are the computing entities, and edges represent communication links. Each node is equipped with a unique identifier (ID) from $\{1,\dots,n^c\}$ where n is the total number of nodes in the graph and $c\geq 1$ is a constant. Initially, each node knows its own ID and degree, the maximum degree Δ of the graph, and the total number n of the nodes. The computation proceeds in synchronous rounds, where at each round each node sends messages to its neighbors, receives messages from its neighbors, and performs some local computation. In the LOCAL model the size of the messages and the local computation is not bounded. We say that an algorithm correctly solves a task in this model (e.g., a vertex coloring) in time T if each node provides a local output (e.g., a color) within T communication rounds, and the local outputs together yield a correct global solution (e.g., a proper coloring). In the randomized version of the LOCAL model, additionally, each node is equipped with a random bit string. In this paper we will consider both Monte Carlo and Las Vegas randomized algorithms. A T-rounds Monte Carlo algorithm must always terminate within T rounds, and the global output it produces must

be correct with high probability, that is, with probability at least $1 - 1/n^c$, for an arbitrary constant $c \geq 1$. A T-rounds Las Vegas algorithm must terminate within T rounds with high probability, and it must always produce a correct solution. Notice that, since the size of the messages is not bounded, we can see a T-round deterministic or a randomized Monte Carlo algorithm that runs at some node u as a mapping of the T-hop neighborhood of u into an output. This does not hold for Las Vegas algorithms, since there the running time is only bounded with high probability.

2.2 Definitions

We start by defining the notion of $(p : q)$-coloring. Informally, this coloring is an assignment of colors to the nodes of a graph, such that the colors come from a palette of p colors, and such that to each node are assigned at least q different colors. Neighboring nodes must have disjoint sets of colors.

▶ **Definition 1** ($(p : q)$-coloring). *Let $p \geq q \geq 1$. A $(p : q)$-coloring of a graph $G = (V, E)$ is an assignment of a set $X_v \subset [p]$ to each node $v \in V$ such that for all $v \in V$, $|X_v| \geq q$, and for all edges $\{u, v\} \in E$, $X_u \cap X_v = \emptyset$.*

Sometimes we are not interested in the total number of colors, but just in the ratio between the total number of colors and the number of colors assigned to the nodes. This notion is captured by the definition of fractional coloring.

▶ **Definition 2** (Fractional c-coloring). *A fractional c-coloring is a $(p : q)$-coloring satisfying $p/q \leq c$.*

Given a $(p : q)$-coloring, we call p the *support* of the coloring. Naturally, apart from minimizing the ratio p/q and the time for computing a fractional coloring, we also want to minimize the support p.

The minimum value c for which there exists a fractional c-coloring is called fractional chromatic number.

▶ **Definition 3** (Fractional chromatic number). *The fractional chromatic number $\chi_f(G)$ of a graph G is defined as*

$$\chi_f(G) := \inf \left\{ \frac{p}{q} \ : \ G \text{ has a } (p : q)\text{-coloring} \right\} = \min \left\{ \frac{p}{q} \ : \ G \text{ has a } (p : q)\text{-coloring} \right\}.$$

▶ **Definition 4** (Partial coloring). *A partial c-coloring is a coloring of the vertices of a graph such that each node is either colored from a color in $\{1, \ldots, c\}$, or is uncolored. Similarly, a partial $(p : q)$-coloring is a coloring of the vertices of a graph such that each node, either has at least q colors in $\{1, \ldots, p\}$, or is uncolored.*

We now provide some additional definitions that will be useful when describing our algorithms. Given a graph G, we denote with $\deg_G(v)$ the degree of node v in G.

▶ **Definition 5** (List coloring). *In the c_v-list (vertex) coloring problem, each node v is equipped with a list of arbitrary c_v colors, and the goal is to assign to each node a color from its list, such that the resulting outcome is a proper coloring of the graph G. In particular, in the $(\text{degree} + x)$-list coloring problem, each node v has a list of size at least $\deg_G(v) + x$.*

▶ **Definition 6** (Degree-choosability). *A graph $G = (V, E)$ is* degree-choosable *if it admits a c_v-list coloring for any list assignment satisfying $|c_v| \geq \deg_G(v), \forall v \in V$.*

▶ **Definition 7** (Network decomposition). *A (c, d)-network decomposition of the graph G is a partition of the vertices of G into at most c disjoint color classes such that each connected subgraph induced by nodes of color i has strong diameter at most d.*

▶ **Definition 8** (Distance). *Let $G = (V, E)$ be a graph. For a pair of nodes $u, v \in V$, we denote with $\mathsf{dist}(u, v)$ the hop-distance between u and v in G. We also denote the distance between a node $v \in V$ and a set of nodes $S \subseteq V$ as $\mathsf{dist}(v, S) = \min\{\mathsf{dist}(v, u) \mid u \in S\}$.*

▶ **Definition 9** (Ruling set). *An (α, β)-ruling set is a set of nodes satisfying that the nodes in the set are at distance at least α between each other, and nodes not in the set are at distance at most β from nodes in the set.*

3 Our Results

Our first main contribution, presented in Section 5, is in the improvement of the main algorithm of [11]. In particular, we start by showing that a fractional $(\Delta + \varepsilon)$-coloring, with small support, can be obtained in a time that depends only polynomially in Δ and ε. In [11], this dependency is exponential. In the following, we assume $\Delta \geq 3$ [4].

▶ **Theorem 10.** *A $(q\Delta : q-1)$-coloring, for an arbitrary integer $q > 0$, can be deterministically computed in time $O(\alpha^2 \log \Delta \cdot T + \alpha \log^* n)$ in the LOCAL model, where T is the time required to solve the $(\text{degree} + 1)$-list coloring problem given an $O(\Delta^2)$-coloring in input, and where $\alpha = O(1 + \log_\Delta q)$.*

If we set $q = \Theta(\Delta/\varepsilon)$, we obtain the following corollary.

▶ **Corollary 11.** *For any $\varepsilon > 0$, the fractional $(\Delta+\varepsilon)$-coloring problem, with support $O(\Delta^2/\varepsilon)$, can be solved deterministically in time*

$$O\left(\left(\log \Delta + \frac{\log^2(1/\varepsilon)}{\log \Delta}\right) \cdot T + \left(1 + \frac{\log(1/\varepsilon)}{\log \Delta}\right) \cdot \log^* n\right),$$

where T is the time required to solve the $(\text{degree} + 1)$-list coloring problem given an $O(\Delta^2)$-coloring in input.

Since the $(\text{degree} + 1)$-list coloring problem, given an $O(\Delta^2)$-coloring, can be solved in $O(\sqrt{\Delta}\,\text{poly}\log \Delta)$ rounds deterministically [8, 17, 36], we obtain the following corollary.

▶ **Corollary 12.** *For any constant $\varepsilon > 0$, the fractional $(\Delta+\varepsilon)$-coloring problem, with support $O(\Delta^2)$, can be solved in $O(\sqrt{\Delta}\,\text{poly}\log \Delta + \log^* n)$ deterministic rounds.*

Then, we show that we can obtain a different tradeoff between the support and the running time.

▶ **Theorem 13.** *A $(q\Delta+1 : q)$-coloring, for an arbitrary integer $q > 0$, can be deterministically computed in time $O(q^2 \log \Delta \cdot T + q \log^* n)$ in the LOCAL model, where T is the time required to solve the $(\text{degree} + 1)$-list coloring problem given an $O(\Delta^2)$-coloring in input.*

If we set $q = \Theta(1/\varepsilon)$, since $T = O(\sqrt{\Delta}\,\text{poly}\log \Delta)$, we obtain the following corollary, that shows that at the cost of a slightly worse running time, we obtain a better support.

[4] We note that trivial adaptations of our algorithms also work for $\Delta = 2$.

▶ **Corollary 14.** *For any $\varepsilon > 0$, the fractional $(\Delta+\varepsilon)$-coloring problem, with support $O(\Delta/\varepsilon)$, can be solved deterministically in time $O\left(\frac{1}{\varepsilon^2}\cdot\sqrt{\Delta}\cdot\operatorname{poly}\log\Delta+\frac{1}{\varepsilon}\cdot\log^* n\right)$.*

We then prove that, at the cost of drastically increasing the number of colors, it is possible to improve the dependency on Δ, and to entirely remove the dependency on n.

▶ **Theorem 15.** *For any $\varepsilon > 0$, the fractional $(\Delta+\varepsilon)$-coloring problem can be solved deterministically in time*

$$O\left(\left(\log\Delta+\frac{\log^2(1/\varepsilon)}{\log\Delta}\right)\cdot\log(\Delta/\varepsilon)\right)=O\left(\log^2\Delta+\log^2\frac{1}{\varepsilon}+\frac{\log^3(1/\varepsilon)}{\log\Delta}\right).$$

▶ **Corollary 16.** *For any $\varepsilon > 1/\Delta^c$, where $c > 0$ is an arbitrary constant, the fractional $(\Delta+\varepsilon)$-coloring problem can be solved in $O(\log^2\Delta)$ deterministic rounds.*

Our second contribution, presented in Appendix A, is in showing that, by allowing a logarithmic dependency on n in the running time, we can obtain fractional colorings that are arbitrarily close to the optimum. We provide both randomized and deterministic algorithms, that differ in the required support and in the running time. Let $p/q = \chi_f(G)$ be the fractional chromatic number of G. The algorithms that we provide do not require to know p and q, but if these values are provided, or even if just the value of $\chi_f(G)$ is provided, then our algorithms obtain a fractional coloring with smaller support. In particular, if p and q are known to the nodes, let $p' = p$ and $q' = q$. Otherwise, let $p' = \chi c\log n/\varepsilon^2$ and $q' = (1-\varepsilon)p'/\chi_f(G)$, where $\chi = \chi_f(G)$ if $\chi_f(G)$ is known to the nodes, and $\Delta+1$ otherwise. We first show the following.

▶ **Theorem 17.** *Let $G = (V, E)$ be a graph that admits a $(p : q)$ coloring, and let $t = O(\log n/\varepsilon)$, for an arbitrary $\varepsilon > 0$. There is a randomized* LOCAL *algorithm that, with high probability, computes a $(tp' : (1-\varepsilon)tq')$-coloring, that is, a fractional $(1+O(\varepsilon))\frac{p}{q}$-coloring, in $O(\log n/\varepsilon)$ rounds.*

We then show two different deterministic algorithms, that provide different tradeoffs between the support and the running time.

▶ **Theorem 18.** *Let $G = (V, E)$ be a graph that admits a $(p : q)$-coloring, and let $t = O(\operatorname{poly} n/\varepsilon)$, for an arbitrary $\varepsilon > 0$. There is a deterministic* LOCAL *algorithm that computes a $(tp' : (1-\varepsilon)tq')$-coloring, that is, a fractional $(1+O(\varepsilon))\frac{p}{q}$-coloring, in $O(\log n/\varepsilon)$ rounds.*

▶ **Theorem 19.** *Let $G = (V, E)$ be a graph that admits a $(p : q)$ coloring, and let $t = O(\log n/\varepsilon)$, for an arbitrary $\varepsilon > 0$. There is a deterministic* LOCAL *algorithm that computes a $(tp' : (1-\varepsilon)tq')$-coloring, that is, a fractional $(1+O(\varepsilon))\frac{p}{q}$-coloring, in $O(\log n(\log^2 n+\mathrm{ND})/\varepsilon)$ rounds, where $\mathrm{ND} \le \operatorname{poly}\log n$ is the time required to compute an $(O(\log n), O(\log n))$-network decomposition.*

In Section 6 we prove that the $O(\log n/\varepsilon)$ time dependency for computing an almost optimal fractional coloring is necessary, even on trees, and even for randomized algorithms.

▶ **Theorem 20.** *Computing a fractional $(2+\varepsilon)$-coloring on trees in the* LOCAL *model requires $\Omega(\log n/\varepsilon)$, even for randomized algorithms.*

Finally, in Appendix B, we consider grids. In [11], it has been shown that, for any constant ε and d, in d-dimensional grids, it is possible to compute a fractional $(2+\varepsilon)$-coloring in time $O(\log^* n)$. We show that the same problem can be solved in constant time.

▶ **Theorem 21.** *Let G be a d-dimensional grid. For any $\varepsilon > 0$, there is a deterministic* LOCAL *algorithm that computes a fractional $(2+\varepsilon)$-coloring on G, that runs in $2^{O(d^2+d\log\frac{1}{\varepsilon})}$ rounds.*

4 Generic results

In this section, we prove some generic statements that will be useful in the following sections.

4.1 From partial colorings to fractional colorings

We start by showing that, given an algorithm that computes a partial fractional coloring satisfying that each node has some fixed probability of being colored, then it is possible, at the cost of increasing the total number of colors, to compute a proper fractional coloring.

▶ **Lemma 22.** *Assume there exists a randomized algorithm $A(n, \varepsilon)$ that runs in $T(n, \varepsilon)$ rounds and, with probability at least $1 - f$, such that $f = f(n, \varepsilon) \leq \varepsilon$, computes a partial $(p : q)$-coloring satisfying that each node is uncolored with probability at most ε. Then, there exists a randomized algorithm that runs in $T(n, \varepsilon/4)$-rounds and, for an arbitrary f', with probability at least $1 - f'$, computes a $(p' : q')$-coloring, where $p' = pt$, $q' = (1 - \varepsilon)qt$, and $t = O(\frac{1}{\varepsilon} \log \frac{n}{f'})$.*

Proof. We run $A(n, \varepsilon/4)$ for $t = \frac{6}{\varepsilon} \log \frac{n}{f'}$ times in parallel. Clearly, the running time is still $T(n, \varepsilon/4)$. For each execution, we use an independent palette of colors of size p, and hence we obtain a palette of pt total colors. We need to prove that, with probability at least $1 - f'$, we assign at least $(1 - \varepsilon)qt$ colors to each node.

Consider an arbitrary node u. Let $X_i = 1$ if node u is uncolored during execution i, and $X_i = 0$ otherwise. By assumption, a node is uncolored with probability at most $\varepsilon/4 + f(n, \varepsilon/4) \leq \varepsilon/2$, hence $P(X_i = 1) \leq \varepsilon/2$. Let $X = \sum_{i=1}^{t} X_i$. By linearity of expectation, we get that $\mathbb{E}[X] \leq \varepsilon t/2$. By a Chernoff bound, we get that:

$$P(X \geq \varepsilon t) \leq e^{\frac{-\varepsilon t}{6}} \leq \frac{f'}{n}.$$

By a union bound we get that each node is uncolored in at most εt colorings with probability at least $1 - f'$. Hence, with probability at least $1 - f'$, each node receives q colors for at least $(1 - \varepsilon)t$ times. ◀

4.2 From private randomness to shared randomness

We now prove a useful lemma, that allows us to reduce the number of random bits used by a randomized algorithm. We will later use this lemma to derandomize fractional coloring algorithms without increasing the support too much. Note that, in the statement of the lemma, the number of bits used by the original algorithm does not play a role in the resulting algorithm. In fact, as a starting point, we essentially only need to know that the amount of randomness, as a function of n, can be bounded.

▶ **Lemma 23.** *Let A be a randomized algorithm that runs in T rounds and solves some problem P with probability of success at least $1 - f$, by using $b = b(n)$ bits of private randomness, where nodes have no private inputs except for their random bit strings and their identifiers. Then, there exists a randomized algorithm A' that uses only $O(\log \frac{n}{f})$ bits of shared randomness and solves P in T rounds with probability of success at least $1 - 2f$.*

Proof. The proof follows a standard argument along the lines of Newman's theorem in communication complexity (see, e.g., [30]). Let $B = Nb$, where $N = n^c$, for some constant $c \geq 1$, is the size of the ID space. Note that any algorithm that uses b bits of private

randomness can be trivially converted into an algorithm that uses B bits of shared randomness. Also, note that by fixing the string of shared randomness used by the nodes, we obtain a deterministic algorithm.

Let $\ell = 2^B$, and let $R = \{R_1, \ldots, R_\ell\}$ be the set of possible shared random bit string assignments. We will prove below, using the probabilistic method, that there is a set $S \subseteq R$ of size $k = O(\frac{n^2}{f})$ satisfying that, for any graph G of n nodes, algorithm A fails for at most a $2f$ fraction of the strings in S. This means that we can construct an algorithm A' that, given a bit string of shared randomness of length $O(\log k) = O(\log \frac{n}{f})$, chooses an element from S uniformly at random, and then executes the T-round algorithm A by using that bit string. This algorithm A' runs in T rounds and solves P with a failure probability of at most $2f$.

We now prove that, by choosing S at random, there is non-zero probability that it satisfies the requirements. We construct $S = \{s_1, \ldots, s_k\}$ by sampling with replacement k strings from R. Consider an arbitrary graph G of n nodes. Let $X_i = 1$ if the execution of A on G by using the bit string s_i would fail. Otherwise, let $X_i = 0$. Since s_i has been chosen uniformly at random, and since the failure probability of A is at most f, then $P(X_i = 1) \leq f$. By linearity of expectation, it holds that $\mathbb{E}[X] \leq fk$. Let $X = \sum_{1 \leq i \leq k} X_i$. Note that the variables are clearly independent. Hence, by a Chernoff bound, we get that

$$P(X \geq 2fk) \leq e^{-\frac{fk}{3}}.$$

If $X \geq 2fk$ we say that S is *bad* for G. Note that there are at most 2^{n^2} graphs of n nodes with no private inputs, and that there are at most $\binom{N}{n}$ possible ID assignments on each graph. Hence, by a union bound, the probability that S is bad for *some* graph of n nodes is at most

$$\binom{N}{n} \cdot 2^{n^2} \cdot e^{-\frac{fk}{3}} \leq e^{cn \ln n + n^2 - \frac{fk}{3}}.$$

Choosing $k = 6n^2/f$ ensures that the above expression is strictly less than 1, for n sufficiently large. Note that $k = O(\frac{n^2}{f})$. ◀

4.3 From randomized fractional colorings to deterministic fractional colorings

We now show that, given a randomized algorithm for fractional coloring, it is possible to obtain a deterministic algorithm with the same running time, at the cost of increasing the support.

▶ **Lemma 24.** *Assume there exists a randomized T-round algorithm A that computes a $(p : q)$-coloring with probability at least $1 - f$. Then, there exists a deterministic T-round algorithm that computes a $(p' : q')$-coloring, where $p' = pt$, $q' = (1 - 2f)qt$, and $t = 2^{O(\log \frac{n}{f})}$.*

Proof. We first apply Lemma 23 to reduce the amount of randomness required by algorithm A to $B = O(\log \frac{n}{f})$ bits of shared randomness, obtaining a new algorithm A' that runs in T rounds and computes a $(p : q)$ coloring with probability at least $1 - 2f$. We cannot directly run A': since nodes are not provided with shared randomness, this is not possible. Instead, we run A' in parallel for all possible $t = 2^B$ values of the shared random bit string assignment. In each run, we use an independent palette of p colors, and hence we use pt colors in total. Since algorithm A' succeeds with probability at least $1 - 2f$, then in at least a fraction $1 - 2f$ of the executions all nodes receive at least q colors. Hence, we obtain that each node receives at least $(1 - 2f)qt$ colors in total. ◀

5 Fast algorithm

In this section, we present an algorithm that is able to compute a fractional $(\Delta+\varepsilon)$-coloring, with a running time that depends only polynomially in Δ, and that requires small support. Later, we will show how to modify the algorithm to obtain a logarithmic dependency in Δ, at the cost of having a much larger support. More formally, we start by proving the following theorem.

▶ **Theorem 10.** *A $(q\Delta : q-1)$-coloring, for an arbitrary integer $q > 0$, can be deterministically computed in time $O(\alpha^2 \log\Delta \cdot T + \alpha \log^* n)$ in the* LOCAL *model, where T is the time required to solve the* (degree + 1)*-list coloring problem given an $O(\Delta^2)$-coloring in input, and where $\alpha = O(1 + \log_\Delta q)$.*

High level idea

Our algorithm, on a high level, works as follows. We first compute a clustering of the graph by computing an (α, β)-ruling set, for suitable parameters α and β, and by connecting each node to the cluster centered at the nearest ruling set node. We then proceed by coloring the nodes in a special way. In particular, we compute q different colorings, such that each coloring is a Δ-coloring of the graph, where some nodes are allowed to be uncolored, and such that each node is uncolored in at most one of the q colorings.

In order to prove that such a coloring can be computed efficiently, we will exploit an important property of the computed clustering: for each cluster, it holds that it either contains many nodes (at least q), or it contains a degree-choosable component. We will show that this implies that we can color the nodes such that, in each cluster that contains a degree-choosable component, the Δ-coloring problem is solved properly, while in the other clusters we can choose a specific node to leave uncolored. If the cluster is large enough, we can have a different uncolored node for each of the q colorings, obtaining that each node is uncolored for at most one coloring.

The computed coloring allows us to assign at least $q-1$ colors to each node of the graph, where the colors come from a palette of size $q\Delta$, and hence we obtain a $(q\Delta : q-1)$-coloring. By choosing the right size of the clusters, we can also prove that the running time is polynomial in Δ, and more precisely that it is only slightly worse than the time required to solve the (degree + 1)-list coloring problem, which we use as a subroutine.

5.1 The clustering

We start by proving that it is possible to compute a clustering of a graph in such a way that it satisfies some desirable properties. In particular, we prove the following lemma.

▶ **Lemma 25.** *Let $\alpha = c(1 + \log_\Delta q)$, for some constant c and an arbitrary integer $q > 0$. It is possible to compute a clustering of a graph G such that the following holds.*

- *Each cluster has a strong diameter of at most $2\alpha^2 \log\Delta$.*
- *Each cluster contains:*
  - *at least q nodes, or*
  - *a node of degree at most $\Delta - 1$, or*
  - *a degree-choosable component, such that all neighbors of the nodes in the degree-choosable component are also contained in the cluster.*

This clustering can be computed in $O(\alpha^2 \log\Delta + \alpha \log^* n)$ deterministic rounds.

In order to prove this lemma, we will use the following lemmas present in the literature.

▶ **Lemma 26** (Lemma 8 of [19]). *Let $G = (V, E)$ be a graph and $v \in V$ be a node such that inside the r-radius neighborhood of v there are no degree-choosable components and every node has degree Δ. Then, for each even r, there are at least $(\Delta - 1)^{r/2}$ nodes at distance r from v.*

▶ **Lemma 27** (Ruling sets, [4, 43]). *A $(2, \beta)$-ruling set can be computed in time $O(\beta\Delta^{2/\beta} + \log^* n)$ deterministic rounds.*

We are now ready to prove Lemma 25.

Proof. We start by computing an $(\alpha, (\alpha-1)\alpha \log \Delta)$-ruling set, by computing a $(2, \alpha \log \Delta)$-ruling set on $G^{\alpha-1}$, the $(\alpha-1)$th power of G. By applying Lemma 27 with $\beta = \alpha \log \Delta$, this ruling set can be computed in $T = O(\alpha^2 \log \Delta + \alpha \log^* n)$ deterministic rounds. Then, in additional $O(\alpha^2 \log \Delta)$ rounds, each node finds the nearest ruling set node (breaking ties arbitrarily), and joins its cluster.

Clearly, the running time requirement is satisfied. We need to prove that this clustering satisfies the required properties. The first property is clearly satisfied, since $\alpha^2 \log \Delta$ is an upper bound on the distance between an arbitrary node and its nearest ruling set node.

We now show that the second property is also satisfied. Let v be the ruling set node of an arbitrary cluster C, that is, its center. By the definition of (α, β)-ruling set, all nodes at distance at most $\lfloor \alpha/2 - 1 \rfloor$ from v are all contained in C. Consider the set S of nodes at distance at most k from v, where $k \in \{\lfloor \alpha/2 - 2 \rfloor, \lfloor \alpha/2 - 3 \rfloor\}$ is even. If there is a node of degree at most $\Delta - 1$ in S, then the property is clearly satisfied. Otherwise, all nodes in S have degree Δ, and by Lemma 26 we get that either there is a degree-choosable component in S, or there are at least $(\Delta - 1)^{k/2}$ nodes in S. In the first case, note that the neighbors of nodes in S are all contained in C, and hence the property is satisfied. In the second case, we obtain that the cluster contains at least $(\Delta - 1)^{\lfloor c(1+\log_\Delta q)/2 - 3 \rfloor/2}$ nodes, that, for a large enough constant value of c, is at least q, and hence the property is satisfied in this case as well. ◀

5.2 The algorithm

We now describe an algorithm that computes a fractional $(\Delta + \varepsilon)$-coloring. In the following, we will make use of the notion of partial Δ-coloring (see Definition 4).

On a high level, the main idea of the algorithm is to compute q (possibly) different partial Δ-colorings (where each of the colorings come from a different palette), such that each node is uncolored in at most 1 of the colorings. In this way, we can assign $q - 1$ colors to each node, from a palette of $q\Delta$ total colors. We now show how the properties stated in Lemma 25 can be used for this purpose.

▶ **Lemma 28.** *Let G be a graph that is clustered according to Lemma 25, where each cluster that contains at least q nodes also contains a special marked node. Let T be the time required to solve the* (degree + 1)*-list coloring problem, and let R be the bound on the diameter of the clusters. Then, in $O(T \cdot R)$ deterministic rounds it is possible to solve the partial Δ-coloring problem such that only marked nodes remain uncolored.*

Proof. We prove this lemma by slightly modifying the core of the deterministic Δ-coloring algorithm presented in Ghaffari et al. [19]. Each node v starts by spending R rounds to gather its entire cluster C_v. In this way, it can see if there is a marked node, or a node with degree at most $\Delta - 1$, or a degree-choosable component (at least one of these cases must apply). If there is a marked node z in C_v, let $S_{C_v} = \{z\}$, otherwise, if there is a node z'

with degree at most $\Delta - 1$, then let $S_{C_v} = \{z'\}$, otherwise, let S_{C_v} be the set of nodes of an arbitrary degree-choosable component guaranteed to exist by Lemma 25. Note that, without additional communication, v can compute its distance from the nearest node s in S_{C_v}.

Let G_i be the subgraph induced by nodes at distance i from their nearest node in the set, that is, $G_i = \{u \mid \mathsf{dist}(u, S_{C_u}) = i\}$. Let $G_{>i}$ (resp. $G_{\geq i}$) be the graph induced by all nodes contained in G_j, for all $j > i$ (resp. $j \geq i$). Note that, for all $i > 0$, the nodes of G_i, in $G_{\geq i}$, have degree at most $\Delta - 1$, since the maximum degree in G is Δ, and all nodes in G_i have at least one neighbor in G_{i-1}. Also, note that $G_{>R}$ is empty.

We proceed as follows. Assume that $G_{>i}$ is properly Δ-colored (we start with $i = R$, where the statement trivially holds, since $G_{>R}$ is empty), and let $c(u)$ be the color of a node u in $G_{>i}$. We show that we can Δ-color $G_{\geq i}$. For each node v in G_i, we define its list of available colors as $L_v = \{1, \ldots, \Delta\} \setminus \{c(u) \mid u \in N(v) \cap G_{>i}\}$. Since the degree of nodes of G_i in $G_{\geq i}$ is at most $\Delta - 1$, then L_v defines a (degree + 1)-list coloring instance of G_i, that can be solved in T rounds. By iterating this procedure for $i = R, \ldots, 1$, we obtain that all nodes of G, except the ones in $S = \bigcup\{S_{C_v} \mid v \in V\}$, are properly Δ-colored.

Finally, we handle the nodes in S. If S_{C_v} contains a marked node, we just leave it uncolored. Otherwise, if S_{C_v} contains a node with degree at most $\Delta - 1$, we color it with an arbitrary available color. Otherwise, if S_{C_v} contains a degree-choosable component, then, for each node $u \in S_{C_v}$, we define L_u as above. This time, L_u defines a degree-list coloring instance. Note that, in general, the degree-list coloring problem may be unsolvable, but this is never the case in a degree-choosable component, by definition. Since, for each pair of clusters C_1, C_2, S_{C_1} and S_{C_2} are non-adjacent, then it is possible to solve all the degree-list coloring instances in parallel, by brute force, in R rounds. ◀

We are now ready to describe the algorithm. First of all, nodes start by computing an $O(\Delta^2)$-coloring. Then, nodes compute the clustering described in Lemma 25. Let $R = O(\alpha^2 \log \Delta)$ be the maximum diameter of the clusters. Then, nodes spend R rounds to check the type of their cluster, that is, if there is a degree-choosable component satisfying the required property, or if there are at least q nodes, or if there is a node with degree at most $\Delta - 1$. In all clusters C containing at least q nodes, we choose q arbitrary distinct nodes $\{v_{C,1}, \ldots, v_{C,q}\}$. Then, we apply Lemma 28 for q times in parallel. During the application i, the nodes that are considered marked are $\{v_{C,i}\}$. We obtain $C_1, \ldots, C_q$ partial Δ-colorings of G, such that each node is uncolored in at most one coloring. Hence, we use a palette of $q\Delta$ colors, such that each node has at least $q - 1$ colors, that is, we obtain a $(q\Delta : q - 1)$-coloring.

Time complexity

The previously described algorithm computes a $(q\Delta : q-1)$-coloring. Hence, in order to prove Theorem 10, we need to give a bound on its running time. Computing the $O(\Delta^2)$-coloring can be done in $O(\log^* n)$ rounds. Computing the clustering requires $O(\alpha^2 \log \Delta + \alpha \log^* n)$ rounds. Gathering the cluster requires $O(\alpha^2 \log \Delta)$ rounds. The application of Lemma 28 requires $O(T \cdot \alpha^2 \log \Delta)$ rounds, where T is the time for solving a (degree + 1)-list coloring instance given an $O(\Delta^2)$-coloring. Recall that $\alpha = c(1 + \log_\Delta q)$. Hence, we obtain an overall time complexity of $O(\alpha^2 \log \Delta \cdot T + \alpha \log^* n)$, where $\alpha = O(1 + \log_\Delta q)$.

5.3 Faster algorithm

We now show that, at the cost of drastically increasing the number of colors, we can improve the dependency on Δ, and entirely remove the dependency on n. In particular, we will prove the following.

▶ **Theorem 15.** *For any $\varepsilon > 0$, the fractional $(\Delta + \varepsilon)$-coloring problem can be solved deterministically in time*

$$O\left(\left(\log\Delta + \frac{\log^2(1/\varepsilon)}{\log\Delta}\right)\cdot\log(\Delta/\varepsilon)\right) = O\left(\log^2\Delta + \log^2\frac{1}{\varepsilon} + \frac{\log^3(1/\varepsilon)}{\log\Delta}\right).$$

We start by explaining how to remove the $O(\alpha\log^* n)$ dependency from the algorithm of Theorem 10, obtaining the following intermediate result.

▶ **Lemma 29.** *For any $\varepsilon > 0$, the fractional $(\Delta+\varepsilon)$-coloring problem can be solved deterministically in time $O\left(\left(\log\Delta + \frac{\log^2(1/\varepsilon)}{\log\Delta}\right)\cdot T\right)$, where T is the time required to solve the $(\text{degree}+1)$-list coloring problem given an $O(\Delta^2)$-coloring in input.*

Proof. On a high level, the $\log^* n$ dependency appears in the time complexity of the algorithm for two reasons:

- The algorithm spends $O(\alpha\log^* n)$ rounds for computing the clustering, and this is because, in order to compute an (α,β)-ruling set, we first need to find an $O(\Delta^{2\alpha})$-coloring of G^α.
- The algorithm spends $O(\log^* n)$ rounds to find an $O(\Delta^2)$ coloring of G. Note that, given an $O(\Delta^{2\alpha})$-coloring of G^α, we can reduce this runtime to $O(\log^*\Delta^{2\alpha})$.

Hence, if, in G^α, nodes are already provided with an $O(\Delta^{2\alpha})$-coloring, then we can get rid of the $\log^* n$ dependency, obtaining a faster algorithm, that requires only $O(\alpha^2\log\Delta\cdot T + \log^*\Delta^{2\alpha}) = O(\alpha^2\log\Delta\cdot T)$ rounds. Unfortunately, finding such a coloring requires $\Omega(\log^* n)$ rounds [32]. In order to overcome this issue, we do the following. We first spend a constant number of rounds to find a color randomly, such that each node has a large enough probability of being colored. Then, we run Theorem 10 on the subgraph induced by nodes that are colored correctly. In this way, we obtain an algorithm that computes a partial fractional coloring, satisfying that each node is uncolored with some fixed probability. We can then apply Lemma 22 and Lemma 24 to obtain a deterministic algorithm that finds a proper fractional coloring.

In order to find the required coloring, we proceed as follows. Each node picks a color uniformly at random from a palette of c colors, for some value c to be fixed later. Then, each node v checks its α-radius neighborhood, and if there is a node u with the same color, then v becomes uncolored. For any node u contained in the α-radius neighborhood of v, we have that $P(u \text{ and } v \text{ pick different colors}) \geq 1 - 1/c$. Let $\bar{\Delta} = \Delta^\alpha$ be an upper bound on the degree of G^α. Then, $P(u \text{ is colored}) \geq (1-1/c)^{\bar{\Delta}}$, that by the Bernoulli inequality is at least $1 - \bar{\Delta}/c$. Hence, by applying the algorithm of Theorem 10 on the subgraph induced by colored nodes, we obtain an algorithm that runs in $O(\alpha^2\log\Delta\cdot T)$ rounds and computes a partial $(q\Delta : q-1)$-coloring where each node is uncolored with probability at most $\bar{\Delta}/c$. We can now apply Lemma 22 (note that, since our algorithm never fails, then $f = 0$), obtaining a randomized algorithm that terminates in $O(\alpha^2\log\Delta\cdot T)$ rounds and computes a $(q\Delta t : (1-\bar{\Delta}/c)(q-1)t)$-coloring, for some t that depends on the target failure probability f' and n. Then, we can apply Lemma 24 to obtain a deterministic algorithm that runs in $O(\alpha^2\log\Delta\cdot T)$ rounds and computes a $(q\Delta t' : (1-2f')(1-\bar{\Delta}/c)(q-1)t')$-coloring, for some t' that depends on f' and n. By fixing $c = q\bar{\Delta}$ and $f' = \frac{1}{2q}$, we obtain a deterministic algorithm for computing a fractional $\frac{q\Delta}{(1-1/q)^2(q-1)}$-coloring, that, for $q = O(\Delta/\varepsilon)$, is a fractional $(\Delta + O(\varepsilon))$-coloring. The running time, for such a value of q, is $O((\log\Delta + \log^2(1/\varepsilon)/\log\Delta)\cdot T)$, as required. ◀

Proof of Theorem 15

Proof. We now explain how to remove the dependency on T, and obtain Theorem 15. Let $q = \Theta(\Delta/\varepsilon)$. In order to obtain a faster algorithm, we start by showing that we can replace the dependency on T in Lemma 28, with $O(\log q)$, at the cost of leaving some nodes uncolored. Hence, we obtain a "sloppy" version of Lemma 28. Recall that T is the time required to run a procedure that solves a (degree + 1)-list coloring instance. Instead of running such a procedure, we run a procedure that partially solves an instance in $O(\log q) = O(\log \frac{\Delta}{\varepsilon})$ rounds, such that a node remains uncolored with probability at most $1/q$. This can be achieved by letting each node try to pick an available color uniformly at random for $O(\log q)$ times [25].

We now show that it is possible to obtain an algorithm that computes a partial $(\Delta : 1)$-coloring satisfying that each node is uncolored with probability at most $2/q$. First, we compute the clustering as in the original algorithm, that is, by applying Lemma 25. Then, on clusters of size at least q, we mark a node of the cluster chosen uniformly at random. By applying the sloppy version of Lemma 28, we obtain what we need.

Finally, by applying Lemma 22 and Lemma 24, we obtain a deterministic algorithm that solves fractional $(\Delta + O(\varepsilon))$-coloring in time $O((\log \Delta + \frac{\log^2(1/\varepsilon)}{\log \Delta}) \cdot \log(\Delta/\varepsilon))$, proving the theorem. ◀

5.4 Better support

We now provide a different algorithm, that has slightly worse running time compared to the one of Theorem 10, but that is able to give a fractional $(\Delta + \varepsilon)$-coloring with smaller support. In particular, we will prove the following theorem.

▶ **Theorem 13.** *A $(q\Delta+1 : q)$-coloring, for an arbitrary integer $q > 0$, can be deterministically computed in time $O(q^2 \log \Delta \cdot T + q \log^* n)$ in the* LOCAL *model, where T is the time required to solve the* (degree + 1)*-list coloring problem given an $O(\Delta^2)$-coloring in input.*

We will exploit the following lemma, first presented in [2], and used also in [11].

▶ **Lemma 30** (Proposition 8 of [2])**.** *Let $q \geq 1$ be an integer and let $P = (v_1, \ldots, v_{2q+1})$ be a path. Assume that for $i \in \{1, 2q+1\}$ the vertex v_i has a list L_{v_i} of at least $q+1$ colors, and for any $2 \leq i \leq 2q$, v_i has a list L_{v_i} of at least $2q+1$ colors. Then, each vertex v_i of P can be assigned a subset $S_i \subseteq L_{v_i}$ of q colors, so that adjacent vertices are assigned disjoint sets.*

Similarly as in the algorithm of Theorem 10, we start by computing an $(\alpha, (\alpha-1)\alpha \log \Delta)$-ruling set, by computing a $(2, \alpha \log \Delta)$-ruling set on $G^{\alpha-1}$, the $(\alpha-1)$-th power of G. This time, we choose $\alpha = 4q + 4$. Then, we also compute an $O(\Delta^2)$ coloring. Then, we form clusters by letting each node u join the cluster C_v centered at the nearest ruling set node v. By the definition of (α, β)-ruling set, all nodes at distance at most $\lfloor \alpha/2 - 1 \rfloor$ from v are contained in C_v. Hence, in C_v, there exists at least one induced path of $\lfloor \alpha/2 - 1 \rfloor$ nodes satisfying that all neighbors of nodes in P are fully contained in C_v (that is, the path obtained by taking v and the nodes contained in some shortest path from v to some node at distance $\lfloor \alpha/2 - 2 \rfloor$ from v). Note that $\lfloor \alpha/2 - 1 \rfloor = 2q + 1$. Let S_v be the set of nodes of the path.

Similarly as in the proof of Lemma 28, we can spend $O(\alpha^2 \log \Delta \cdot T)$ rounds to find a partial Δ-coloring that leaves uncolored only nodes in $\bigcup S_v$. This partial coloring can be trivially converted into a partial $(q\Delta : q)$-coloring where each colored node has exactly q colors. Then, for each node u contained in some S_v, we can define its list of available colors

as $\{1, \ldots, q\Delta + 1\} \setminus \{C(z) \mid z \in N(u)\}$, where $C(z)$ is the set of colors assigned to node z. Note that, from the list of available colors of u, we removed at most q colors for each neighbor of u. Hence, the endpoints of the path satisfy $|L_u| \geq q + 1$ since they have at most $\Delta - 1$ colored neighbors, and the inner nodes satisfy $|L_u| \geq 2q + 1$ since they have at most $\Delta - 2$ colored neighbors. Hence, by applying Lemma 30 we get that nodes can complete the coloring by brute force, since paths are not connected to each other.

We spend $O(q^2 \log \Delta + q \log^* n)$ rounds to compute a ruling set, $O(\log^* n)$ rounds to compute the $O(\Delta^2)$ coloring. The rest of the algorithm requires $O(q^2 \log \Delta \cdot T)$ rounds. Each node receives q colors, from a palette of total $q\Delta + 1$ colors. Hence, the theorem follows.

6 Lower bound

In this section, we show a lower bound of $\Omega(\log n/\varepsilon)$ rounds for computing a fractional $(2+\varepsilon)$-coloring, which holds already on trees, and even for randomized algorithms. We first show that there exist graphs with girth $\Omega(\log n/\varepsilon)$ and fractional chromatic number strictly larger than $2 + \varepsilon$. Then we argue that, if we take an algorithm that, on trees, achieves a fractional $(2+\varepsilon)$-coloring in $o(\log n/\varepsilon)$ rounds, and we execute it on such graphs, it must fail, since the obtained fractional coloring would be too good. We get our contradiction on the existence of such an algorithm by arguing that, in time $o(\log n/\varepsilon)$, a node cannot distinguish whether it is on a tree or on these graphs.

We start by describing a graph family of interest. Let $\mathcal{G}^*$ be a graph family that contains all graphs with n nodes, $m = O(n)$ edges, girth $\Omega(\log n)$, and where the largest independent set has size at most n/c, for some large constant c, and for infinite values of n. Such a family of graphs is known to exist [34]. Starting from a graph $G = (V, E) \in \mathcal{G}^*$, we construct a graph H by replacing all edges of G with a path of length $2k + 1$, for some $k = \Theta(1/\varepsilon)$. In other words, let $e = \{u, v\}$ be an edge in G. We replace e by a path $P^e = (u, w_1^e, w_2^e, \ldots, w_{2k}^e, v)$. We refer to nodes in P^e as *path nodes*, and we refer to the $w_1^e, w_2^e, \ldots, w_{2k}^e$ nodes as *inner path nodes*. Let $\mathcal{G}$ be the family that contains all such graphs. By construction, a graph $H \in \mathcal{G}$ has $N = n + 2km$ nodes and girth $\Omega(\log n/\varepsilon)$. We now show the following lemma about the fractional chromatic number of these graphs.

▶ **Lemma 31.** *Let H be a graph in $\mathcal{G}$. The fractional chromatic number of H is strictly larger than $2/(1 - c'\varepsilon)$, for some constant $c' > 0$.*

Proof. Let S be any independent set of H. We modify S and compute a new independent set S' of H such that $|S'| \geq |S|$ and S' contains exactly mk inner path nodes. Note that this implies that for each path $P^e = (u, w_1^e, w_2^e, \ldots, w_{2k}^e, v)$, at most one of the two endpoints can be in S'. Let $P^e = (u, w_1^e, w_2^e, \ldots, w_{2k}^e, v)$ be a path in H where at most $k - 1$ inner path nodes are in S. There are two cases: either at most one of the two endpoints is in S, or both u and v are in S.

In the former case, we modify S in the following way. W.l.o.g., let u be a node not in S. We start from u and compute an optimal independent set inside P^e sequentially, by starting with w_1^e in the independent set and then putting every other node in the set. This procedure puts in the independent set k inner path nodes of P^e, and note that the obtained set is still independent.

In the latter case, we remove node u from the set, and then we proceed as in the former case. Note that the obtained set is still independent, and it is not smaller. In fact, before the modification, there were at most $k - 1$ nodes of $P^e \setminus \{u, v\}$ in S, and hence at most k nodes of $P^e \setminus \{v\}$ in S, and after the modification we have k nodes of $P^e \setminus \{u, v\}$ in the set.

We call S' the independent set resulting from the above operations. Notice that $|S'| \geq |S|$. Recall that each graph in $\mathcal{G}$ is obtained from a graph $G \in \mathcal{G}^*$ where the largest independent set has size at most n/c. Note also that, by construction, if we project S' onto G, then we obtain and independent set of G. Hence,

$$\begin{aligned}
|S'| &\leq \frac{n}{c} + mk = \frac{N}{2} - \frac{(c-2)n}{2c} = \frac{N}{2}\left(1 - \frac{2(c-2)n}{2cN}\right) = \frac{N}{2}\left(1 - \frac{(c-2)n}{c(n+2km)}\right) \\
&= \frac{N}{2}\left(1 - \frac{(c-2)n}{c(n+2\frac{c_1}{\varepsilon}c_2 n)}\right) \text{ where } k = \frac{c_1}{\varepsilon} \text{ and } m = c_2 n \\
&= \frac{N}{2}\left(1 - \frac{c-2}{c(1+2\frac{c_1}{\varepsilon}c_2)}\right) < \frac{N}{2}\left(1 - c_3\varepsilon\right) \text{ for some } c_3 > 0 \text{ that depends on } c_1 \text{ and } c.
\end{aligned}$$

Hence, any color can be assigned to strictly less than a fraction $(1 - c_3\varepsilon)/2$ of the nodes, implying that the fractional chromatic number of H is strictly larger than $2/(1 - c_3\varepsilon)$. ◀

From the above lemma we get the following corollary.

▶ **Corollary 32.** *There exists an infinite family of graphs of girth $\Omega(\log n/\varepsilon)$ and fractional chromatic number strictly larger than $2 + \varepsilon$.*

We are now ready to prove our main theorem.

▶ **Theorem 33.** *Computing a $(2+\varepsilon)$-fractional coloring on trees in the* LOCAL *model requires $\Omega(\log n/\varepsilon)$, even for randomized algorithms.*

Proof. The proof follows a standard indistinguishability argument, already used to prove, e.g., that coloring trees with $o(\Delta/\log\Delta)$ colors requires $\Omega(\log_\Delta n)$ rounds, even for randomized algorithms [32]. For simplicity we prove our claim for the deterministic case, but it can be extended to the randomized case with standard techniques.

Suppose there exists an algorithm A_T that, on trees, computes a fractional $(2+\varepsilon)$-coloring in $o(\log n/\varepsilon)$ rounds. Let G be a graph satisfying Corollary 32. In $o(\log n/\varepsilon)$ rounds, each node in G does not see any cycle, and hence we could run algorithm A_T on G and it would not notice that it is being run on a graph that is not a tree. However, A_T must fail on G, since, by Corollary 32, the fractional chromatic number of G is strictly larger than $2 + \varepsilon$. Hence, suppose we run A_T on G and it fails on the neighboring nodes u and v who, as an output of A_T, got some common color (note that the algorithm may also fail on just one node by assigning to it too few colors, but the lower bound argument follows in the same way).

Recall that a t-round algorithm in the LOCAL model can be seen as a mapping from t-radius neighborhoods into outputs. Let B_u and B_v be the views of radius t of nodes u and v, respectively, that A_T then maps into the outputs of u and v. Let $B = B_u \cup B_v$. The subgraph in B does not contain cycles and it has radius $o(\log n/\varepsilon)$. Starting from B, we construct a tree that contains B, and where we add additional nodes in order to obtain an n-node tree T (note that the additional nodes can be added without altering the t-radius views of u and v, since, in B, there exists at least one node at distance at least t from both of them). Nodes u and v in T have the same exact view as in G, hence they output the same improper fractional coloring, meaning that A_T fails on T, which is a contradiction. Hence, A_T cannot exist, proving the theorem. ◀

References

1. N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986.
2. Yves Aubry, Jean-Christophe Godin, and Olivier Togni. Every triangle-free induced subgraph of the triangular lattice is (5m, 2m)-choosable. *Discret. Appl. Math.*, 166:51–58, 2014.
3. B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network decomposition and locality in distributed computation. In *Proc. 30th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 364–369, 1989.
4. Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. Distributed lower bounds for ruling sets. In *61st IEEE Annual Symposium on Foundations of Computer Science, (FOCS)*, pages 365–376, 2020.
5. L. Barenboim. Deterministic (Δ+ 1)-coloring in sublinear (in Δ) time in static, dynamic, and faulty networks. *Journal of the ACM*, 63(5):1–22, 2016.
6. L. Barenboim and M. Elkin. Deterministic distributed vertex coloring in polylogarithmic time. In *Proc. 29th Symp. on Principles of Distributed Computing (PODC)*, pages 410–419, 2010.
7. L. Barenboim, M. Elkin, and C. Gavoille. A fast network-decomposition algorithm and its applications to constant-time distributed computation. In *Proc. 22nd Coll. on Structural Information and Communication Complexity (SIROCCO)*, pages 209–223, 2015.
8. L. Barenboim, M. Elkin, and U. Goldenberg. Locally-iterative distributed $(\Delta + 1)$-coloring below Szegedy-Vishwanathan barrier, and applications to self-stabilization and to restricted-bandwidth models. In *Proc. 37th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 437–446, 2018.
9. L. Barenboim, M. Elkin, and F. Kuhn. Distributed $(\Delta + 1)$-coloring in linear (in Δ) time. *SIAM Journal on Computing*, 43(1):72–95, 2015.
10. L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. The locality of distributed symmetry breaking. *Journal of the ACM*, 63:20:1–20:45, 2016.
11. Nicolas Bousquet, Louis Esperet, and Fraigniaudnçois Pirot. Distributed algorithms for fractional coloring. In *Proc. 28th Coll. on Structural Information and Communication Complexity (SIROCCO)*, pages 15–30, 2021.
12. S. Brandt, O. Fischer, J. Hirvonen, B. Keller, T. Lempiäinen, J. Rybicki, J. Suomela, and J. Uitto. A lower bound for the distributed Lovász local lemma. In *Proc. 48th ACM Symp. on Theory of Computing (STOC)*, pages 479–488, 2016.
13. Y.-J. Chang, W. Li, and S. Pettie. An optimal distributed $(\Delta + 1)$-coloring algorithm? In *Proc. 50th ACM Symp. on Theory of Computing (STOC)*, 2018.
14. M. Elkin and O. Neiman. Distributed strong diameter network decomposition. In *Proc. 35th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 211–216, 2016.
15. Michael Elkin, Seth Pettie, and Hsin-Hao Su. (2Δ - l)-edge-coloring is much easier than maximal matching in the distributed setting. In *Proc. 26th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 355–370, 2015.
16. Salwa Faour and Fabian Kuhn. Approximating Bipartite Minimum Vertex Cover in the CONGEST Model. In *24th International Conference on Principles of Distributed Systems (OPODIS)*, pages 29:1–29:16, 2021.
17. P. Fraigniaud, M. Heinrich, and A. Kosowski. Local conflict coloring. In *Proc. 57th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 625–634, 2016.
18. M. Ghaffari, D. Harris, and F. Kuhn. On derandomizing local distributed algorithms. In *Proc. 59th Annual Symposium on Foundations of Computer Science, (FOCS)*, pages 662–673, 2018.
19. Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, and Yannic Maus. Improved distributed Δ-coloring. *Distributed Comput.*, 34(4):239–258, 2021.
20. Mohsen Ghaffari and Fabian Kuhn. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. In *Proc. 62nd IEEE Symp. on Foundations of Computer Science (FOCS)*, 2021.

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

22 Magnús M. Halldórsson, Alexandre Nolin, and Tigran Tonoyan. Ultrafast distributed coloring of high degree graphs. *CoRR*, abs/2105.04700, 2021. `arXiv:2105.04700`.

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

24 Henning Hasemann, Juho Hirvonen, Joel Rybicki, and Jukka Suomela. Deterministic local algorithms, unique identifiers, and fractional graph colouring. *Theor. Comput. Sci.*, 610:204–217, 2016.

25 Öjvind Johansson. Simple distributed *Delta*+1-coloring of graphs. *Inf. Process. Lett.*, 70(5):229–232, 1999.

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

27 F. Kuhn. Local multicoloring algorithms: Computing a nearly-optimal TDMA schedule in constant time. In *Proc. Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 613–624, 2009.

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

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

30 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1996.

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

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

33 N. Linial and M. Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993.

34 Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Comb.*, 8(3):261–277, 1988.

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

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

37 Yannic Maus. Distributed graph coloring made easy. In *Proc. 33rd ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 362–372, 2021.

38 Gary L Miller, Richard Peng, and Shen Chen Xu. Parallel graph decompositions using random shifts. In *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures*, pages 196–203, 2013.

39 Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM J. on Discrete Math.*, 4(3):409–412, 1991.

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

41 A. Panconesi and A. Srinivasan. The local nature of Δ-coloring and its algorithmic applications. *Combinatorica*, 15(2):255–280, 1995.

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

43 Johannes Schneider, Michael Elkin, and Roger Wattenhofer. Symmetry breaking depending on the chromatic number or the neighborhood growth. *Theor. Comput. Sci.*, 509:40–50, 2013.

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

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

A Approximating the fractional chromatic number

In this section, we show that it is possible to find arbitrarily good approximations of the fractional chromatic number. In particular, given a graph G with fractional chromatic number $\chi_f(G) = p/q$, we provide randomized and deterministic algorithms that are able to find a fractional $(1+\varepsilon)\chi_f(G)$-coloring, for any $\varepsilon > 0$. Our algorithms use a different amount of total colors, depending on whether the nodes know p and q or not, or if they know $\chi_f(G)$.

On a high level, we show that it is possible to cluster a graph such that each node is unclustered with probability at most ε, and such that any pair of clusters is at least 2 hops apart (that is, for any two nodes that are in different clusters it holds that they do not share an edge). In this way, inside each cluster, we can optimally solve the fractional $\chi_f(G)$-coloring, or find a good-enough approximation if p and q are not known. Then, by applying Lemma 22 and Lemma 24, we obtain randomized and deterministic algorithms for computing a fractional coloring that approximates the fractional chromatic number.

A.1 Computing a clustering

In order to obtain a clustering of the graph, we slightly modify the clustering algorithm of Miller, Peng, and Xu [38] (MPX), to make it compute clusters that are 2 hops apart, such that each node is unclustered with probability at most ε, and each cluster has weak diameter $O(\log n/\varepsilon)$.

▶ **Lemma 34.** *Given a graph $G = (V, E)$, there is a randomized algorithm that computes a 2-hops-apart clustering of G such that each node is unclustered with probability at most ε, and each cluster has weak diameter $O(\log n/\varepsilon)$ with high probability. This algorithm terminates in $O(\log n/\varepsilon)$ rounds with high probability.*

Proof. Since we are going to use a modified version of the MPX procedure, we start by describing the standard MPX procedure.

Each node u chooses independently a *shift* δ_u from an exponential distribution with parameter $\gamma = \varepsilon/2$. Let the *shifting distance* from u to v be denoted as $\mathsf{dist}\_{\delta}(u, v) = \mathsf{dist}(u, v) - \delta_u$. Each node v is then assigned to the cluster C_u centered at the node u that, among all nodes in G, minimizes the value of the shifted distance $\mathsf{dist}\_{\delta}(u, v)$, breaking ties arbitrarily. Miller, Peng, and Xu [38] proved that, with high probability, each cluster has radius $O(\log n/\varepsilon)$. While the original procedure is designed to work in the PRAM model, it is folklore that it can be easily converted into a distributed algorithm that terminates in $O(\log n/\varepsilon)$ rounds with high probability. This procedure obtains a partitioning of the vertices into clusters satisfying the following properties.

- Each cluster is connected, that is, for any two nodes u and v it holds that, if both are in the same cluster C, then the nodes in the shortest path between u and v are also in C.
- Each cluster has strong diameter $O(\log n/\varepsilon)$ with high probability.
- Each edge has probability at most ε to be an intercluster edge.

We run this procedure, and then we modify the obtained clustering in the following way.[5] Let $\{u, v\} \in E$ be an edge such that its endpoints u and v are on different clusters. For each such edge, we choose the endpoint with the smallest identifier and we remove it from the cluster. We say that these nodes are *unclustered*. After this operation it holds that, for any two nodes that are in different clusters, they do not share an edge, meaning that the clusters are at least 2 hops apart, as desired. Notice that, by removing nodes from the clusters we may lose the guarantee on the strong diameter of the clusters. However, it still holds that the *weak* diameter of each cluster is $O(\log n/\varepsilon)$ w.h.p., and this is enough for our purposes.

What is left to show is that each node is unclustered with probability at most ε. Consider an unclustered node u. Let $C_{u'}$ be the cluster centered at node u' where u belonged to before being removed. Since u is unclustered, it means that there is a node w neighbor of u such that, before removing nodes from the clusters, nodes u and w were assigned to different clusters (otherwise u would not be unclustered). Let $C_{w'}$ be the cluster centered at w' that was initially assigned to node w. By construction of these clusters, we have that $\mathsf{dist\_}_\delta(u', u) \leq \mathsf{dist\_}_\delta(w', w) + 1$ and, at the same time, $\mathsf{dist\_}_\delta(w', w) \leq \mathsf{dist\_}_\delta(u', u) + 1$. Hence, $|\mathsf{dist\_}_\delta(u', u) - \mathsf{dist\_}_\delta(w', w)| \leq 1$, implying that $|\mathsf{dist\_}_\delta(u', u) - \mathsf{dist\_}_\delta(w', u)| \leq 2$. In other words, the absolute value of the difference between the smallest and the second smallest shifting distance of an unclustered node is at most 2. In [38] it has been proven that, for each node, the probability that this event happens is at most $2\gamma = \varepsilon$. We thus obtain that each node is unclustered with probability at most ε. ◀

In the remaining of this section we use the variables p' and q', that are defined as follows.

- If p and q are known to the nodes, then $p' = p$ and $q' = q$.
- Otherwise, let $p' = \chi c \log n/\varepsilon^2$ and $q' = (1-\varepsilon)p'/\chi_f(G)$, where $\chi = \chi_f(G)$ if $\chi_f(G)$ is known to the nodes, and $\Delta + 1$ otherwise.

A.2 Solving a partial fractional coloring

We now prove that, by using the clustering algorithm of Lemma 34, it is possible to find a partial $(p' : q')$-coloring.

▶ **Lemma 35.** *There exists a randomized $O(\log n/\varepsilon)$-round algorithm A that computes a partial $(p' : q')$-coloring satisfying that, with probability at least $1 - 1/n$, each node is uncolored with probability at most ε.*

Proof. Note that the algorithm described in Lemma 34 is Las Vegas, but we can turn it into a Monte Carlo algorithm by truncating its execution after $O(\log n/\varepsilon)$ steps, and leave unclustered every node that did not terminate. Since the original algorithm terminates in $O(\log n/\varepsilon)$ rounds with high probability (that is, at least $1 - 1/n$), then this new algorithm always terminates in $O(\log n/\varepsilon)$ rounds, which is also an upper bound on the diameter of the clusters, and leaves each node unclustered with probability at most $\varepsilon + 1/n$ by a union bound. Hence, by slightly scaling ε, we obtain the same guarantees as the original algorithm.

Hence, we start by running the (Monte Carlo variant of the) clustering algorithm. Then, since each cluster has weak diameter at most $R = O(\log n/\varepsilon)$, we can spend R rounds for computing in parallel, in each cluster, by brute force, a $(p' : q')$-coloring (we will later argue why such a coloring always exists). Note that this is possible even if q' is not known to the nodes, as they can just find the best possible solution. Since unclustered nodes do not get a color, and since clusters are 2 hops apart, then there are no neighboring nodes that get the same color.

[5] A similar modification has been used, for example, in [16].

We need to argue why a $(p' : q')$-coloring always exists. Let G be a graph that is $(p : q)$-colored. We show that there is a randomized process that, with non-zero probability, produces a $(p' : q')$-coloring. By the probabilistic method, this implies that G admits a $(p' : q')$-coloring, and hence that also each cluster admits a $(p' : q')$-coloring.

We sample with replacement p' colors from p colors. Consider an arbitrary node u. Let $X_i = 1$ if, during the i-th sampling, we sample a color that node u has among its colors. Otherwise, let $X_i = 0$. Since node u has at least q out of the p possible colors, then $P(X_i = 1) \geq q/p$. Let $X = \sum_{i=1}^{p'} X_i$. By linearity of expectation, it holds that $\mathbb{E}[X] \geq \frac{p'q}{p} = \frac{p'}{\chi_f(G)} = q'/(1-\varepsilon)$. By a Chernoff bound, we get that

$$P(X \leq q') \leq e^{-\frac{\varepsilon^2}{2} \cdot \frac{p'}{\chi_f(G)}} = e^{-\frac{\varepsilon^2 c_\chi \log n}{\varepsilon^2 2 \chi_f(G)}} \leq n^{-c/2}.$$

By a union bound, we have that each node has less than q' colors with probability at most $n^{1-c/2}$. By choosing $c \geq 4$ we get that each node has at least q' colors with probability at least $1 - 1/n$. ◀

A.3 Putting things together

We now prove the existence of randomized and deterministic algorithms for approximating the fractional chromatic number, with running time $O(\log n/\varepsilon)$. Note that, for $\varepsilon < 1/n$, we can trivially solve the problem by gathering the entire graph on a node and brute forcing a solution. Hence, in the following, assume $\varepsilon \geq 1/n$.

Lemma 35 guarantees the existence of a randomized algorithm that runs in $O(\log n/\varepsilon)$ rounds, and with probability at least $1 - f$, where $f = 1/n$, computes a partial $(p' : q')$-coloring satisfying that each node is uncolored with probability at most ε. By applying Lemma 22, we obtain that there exists a randomized algorithm, that also runs in $O(\log n/\varepsilon)$ rounds and, with probability at least $1 - f'$, where $f' = 1/n^c$ for an arbitrary constant $c \geq 1$, computes a partial $(p'' : q'')$-coloring, where $p'' = p't$, $q'' = (1-\varepsilon)q't$, and $t = O(\log n/\varepsilon)$. Hence, we obtain the following.

▶ **Theorem 17.** *Let $G = (V, E)$ be a graph that admits a $(p : q)$ coloring, and let $t = O(\log n/\varepsilon)$, for an arbitrary $\varepsilon > 0$. There is a randomized* LOCAL *algorithm that, with high probability, computes a $(tp' : (1-\varepsilon)tq')$-coloring, that is, a fractional $(1 + O(\varepsilon))\frac{p}{q}$-coloring, in $O(\log n/\varepsilon)$ rounds.*

Then, starting from this algorithm, we can apply Lemma 24, where $f = 1/n^c$ for an arbitrary constant $c \geq 1$, and obtain that there exists a deterministic algorithm that also runs in $O(\log n/\varepsilon)$ rounds, and computes a partial $(p''' : q''')$-coloring, where $p''' = p''t'$, $q''' = (1-2f)q''t'$, and $t' = \operatorname{poly} n$. Since $\varepsilon \geq 1/n \geq f$, then this means that, compared to the randomized algorithm, we lose at most an $O(\varepsilon)$ fraction of colors. Hence, we obtain the following.

▶ **Theorem 18.** *Let $G = (V, E)$ be a graph that admits a $(p : q)$-coloring, and let $t = O(\operatorname{poly} n/\varepsilon)$, for an arbitrary $\varepsilon > 0$. There is a deterministic* LOCAL *algorithm that computes a $(tp' : (1-\varepsilon)tq')$-coloring, that is, a fractional $(1 + O(\varepsilon))\frac{p}{q}$-coloring, in $O(\log n/\varepsilon)$ rounds.*

A.4 Less colors

We now show an alternative way for derandomizing the algorithm of Theorem 17, obtaining the same guarantees on the number of colors, but with a slightly worse running time. For this purpose, we use the following result of Ghaffari, Harris, and Kuhn [18].

▶ **Theorem 36** (Theorem 1.1 of [18]). *Any r-round randomized* LOCAL *algorithm for a locally checkable problem can be transformed into a deterministic* LOCAL *algorithm with complexity $O(r(\log^2 n + \mathrm{ND}))$, where* ND *is the time required to compute an $(O(\log n), O(\log n))$-network decomposition.*

We note that Theorem 1.1 of [18] applies to randomized algorithms that satisfy the following requirements.

1. They always terminate within r rounds.
2. Each node sets a flag to 1 if its output is incorrect, and to 0 otherwise. With high probability, the flag should be 0.
3. Each node should be able to check, in $O(1)$ rounds, whether its flag has the correct value.

In order to apply Theorem 36 and derandomize the algorithm of Theorem 17, we need to show that we can tweak the algorithm to satisfy the above requirements. The algorithm presented in Theorem 17 clearly satisfies the first requirement. However, it is not compatible with the second one, since the guarantee on the quality of the coloring does not always hold. More precisely, the number of colors obtained by a node only holds with high probability, and for this reason, if q' is not known, a node may not notice that its output is incorrect, and it fails to set its flag correctly. Let us see how to handle this issue.

If q' is known to the nodes, then they can just set their flag to 1 if they have strictly less than $(1-\varepsilon)q't$ colors, and 0 otherwise. If q' is not known, we perform a preprocessing step to compute a value of q' (that may be different for different nodes), that will then be used by the nodes to decide whether to set their flag or not. The preprocessing step works as follows. Let T be the running time of the algorithm that we want to derandomize. Each node v spends $2T$ rounds to gather its $2T$-radius neighborhood. Then, it checks, in that neighborhood, what is the maximum value q' for which there exists a $(p't : (1-\varepsilon)q't)$-coloring. Observe that the obtained value q' is a lower bound on the number of colors that v, while executing the algorithm of Theorem 17, is able to obtain when it belongs to a cluster. This follows from the fact that, any cluster where v belongs to must be fully contained in its $2T$-radius neighborhood. Note that the preprocessing step also guarantees that the third requirement of Theorem 1.1 of [18] is satisfied, since each node can just check whether the flag is set correctly depending on the number of colors that it has and the value of q'.

Hence, by combining Theorem 36 with the obtained variant of the algorithm of Theorem 17, we obtain the following theorem.

▶ **Theorem 19.** *Let $G = (V, E)$ be a graph that admits a $(p : q)$ coloring, and let $t = O(\log n/\varepsilon)$, for an arbitrary $\varepsilon > 0$. There is a deterministic* LOCAL *algorithm that computes a $(tp' : (1-\varepsilon)tq')$-coloring, that is, a fractional $(1+O(\varepsilon))\frac{p}{q}$-coloring, in $O(\log n(\log^2 n + \mathrm{ND})/\varepsilon)$ rounds, where $\mathrm{ND} \leq \mathrm{poly}\log n$ is the time required to compute an $(O(\log n), O(\log n))$-network decomposition.*

B Grids

In [11], it has been shown that, for any constant ε and d, in d-dimensional grids, it is possible to compute a fractional $(2+\varepsilon)$-coloring in time $O(\log^* n)$. We show that the same problem can be solved in constant time.

The algorithm of [11] computes a $(2q + 4 \cdot 6^d : q)$-coloring that runs in $O(d\ell(2\ell)^d + d\ell \log^* n)$ rounds, where $\ell = q + 2 \cdot 6^d$. The running time is dominated by the time required to compute a maximal independent set on G^ℓ, where the distance is taken w.r.t. the infinity norm (that is, for two nodes u and v with coordinates $(u_1, \ldots, u_d)$ and $(v_1, \ldots, v_d)$ their

distance is $\max_{1\leq i\leq d}\{|u_i - v_i|\}$). In fact, a maximal independent set can be computed in time $O(\Delta + \log^* n)$, and on G^ℓ we have that $\Delta = (2\ell+1)^d$, and hence there is an overhead of $O(d\ell)$ in the runtime. After computing the independent set, the rest of the algorithm requires just $O(d\ell(2\ell)^d)$ rounds. By setting $q = 2^{O(d+\log\frac{1}{\varepsilon})}$, this algorithm gives a fractional $(2+\varepsilon)$-coloring in time $T_{\text{coloring,n}} + T_{\text{rest}}$, where $T_{\text{coloring},c} = 2^{O(d+\log\frac{1}{\varepsilon})}\log^* c$ and $T_{\text{rest}} = 2^{O(d^2+d\log\frac{1}{\varepsilon})}$. Similarly as in the proof of Lemma 29, we can replace the $\log^* n$ dependency with $\log^* c$, if nodes are provided with a distance-$d\ell$ c-coloring.

We compute a partial distance-$d\ell$ coloring by letting nodes pick a color uniformly at random. Nodes that obtain an invalid coloring, uncolor themselves. We would like to execute the algorithm of [11] on the subgraph induced by colored nodes, but we cannot, since the subgraph is not a grid anymore. In order to solve this issue, we consider only nodes satisfying that, within their running time, they cannot notice that the graph is not a grid. We call these nodes *happy*. In other words, a node is happy if and only if, within the running time of the algorithm, it does not see any uncolored node. On these nodes, by a standard indistinguishability argument, the algorithm must work correctly. Note that this running time depends on c, but we will pick a value of c satisfying that the total running time is anyways strictly less than kT_{rest}, for some large enough constant k. The probability that a node is colored is at least $1 - \Delta/c$. Since a node sees at most $d^{kT_{\text{rest}}}$ nodes within its running time, the probability that a node is happy is at least $1 - \frac{\Delta}{c}d^{kT_{\text{rest}}}$, meaning that a node is unhappy with probability at most $\frac{\Delta}{c}d^{kT_{\text{rest}}}$. We want this probability to be at most ε, and for that, we can pick $c = \Delta d^{kT_{\text{rest}}}/\varepsilon$. Note that, for such a value of c, $T_{\text{coloring},c} + T_{\text{rest}} \leq kT_{\text{rest}}$, as required.

Hence, there is an algorithm that in $2^{O(d^2+d\log\frac{1}{\varepsilon})}$ rounds computes a partial fractional $(2+\varepsilon)$-coloring satisfying that each node is uncolored with probability at most ε. By applying Lemma 22 and Lemma 24, we obtain the following.

▶ **Theorem 21.** *Let G be a d-dimensional grid. For any $\varepsilon > 0$, there is a deterministic* LOCAL *algorithm that computes a fractional $(2+\varepsilon)$-coloring on G, that runs in $2^{O(d^2+d\log\frac{1}{\varepsilon})}$ rounds.*