Decentralized Distributed Graph Coloring II: Degree+1-Coloring Virtual Graphs

Maxime Flin
Reykjavík University, Iceland

Magnús M. Halldórsson
Reykjavík University, Iceland

Alexandre Nolin
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Abstract

Graph coloring is fundamental to distributed computing. We give the first general treatment of the coloring of virtual graphs, where the graph H to be colored is locally embedded within the communication graph G. Besides generalizing classical distributed graph coloring (where $H = G$), this captures other previously studied settings, including cluster graphs and power graphs.

We find that the complexity of coloring a virtual graph depends linearly on the edge congestion of its embedding. The main question of interest is how fast we can color virtual graphs of constant congestion. We find that, surprisingly, these graphs can be colored nearly as fast as ordinary graphs. Namely, we give a $O(\log^4 \log n)$-round algorithm for the deg+1-coloring problem, where each node is assigned more colors than its degree.

This can be viewed as a case where a distributed graph problem can be solved even when the operation of each node is decentralized.

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

Keywords and phrases Graph Coloring, Distributed Algorithms, Virtual Graphs, Congestion, Dilation

Digital Object Identifier 10.4230/LIPIcs.DISC.2024.24

Related Version *Full Version*: `https://arxiv.org/abs/2408.11041` [21]

Funding *Maxime Flin*: Icelandic Research Fund (grant 2310015).
Magnús M. Halldórsson: Icelandic Research Fund (grant 217965).

1 Introduction

We give the first full treatment of distributed graph coloring under bandwidth constraints. Namely, we treat the general case when the input graph H differs from the communication graph G. Previously, the problem was studied for cases when $H = G$ (e.g., [54, 7, 41]) or when H has a particular layout in G (e.g., $H = L(G)$ [4, 43], $H = G^2$ [39, 40, 19], or $H = G^k$ [8]).

Most distributed graph algorithms assume that the input graph H is equivalent to the communication network infrastructure G. In the LOCAL model, this is often without loss of generality, as simulating a round of LOCAL on H while communicating on $G = (V_G, E_G)$ without bandwidth restriction is trivial as long as adjacent vertices in H are $O(1)$-hops away in G. When we restrict message size, however, naive simulation is prohibitively inefficient. The delivery of individual messages to each neighbor of a node can slow down the algorithm by a factor proportional to degrees, which might be as high as $n = |V_H|$. Handling cases where $H \neq G$ is an overarching issue in the design of CONGEST algorithms (e.g., in [34, 30, 33, 60, 29, 22, 36, 59, 28]) that is salient when using a CONGEST algorithm

as a subroutine (e.g., local rounding [15] used in [28]) or when modifying the input graph (e.g., contracting edges [33, 22]). We attempt to study *how bandwidth constraints affect distributed algorithms solving problems on graphs whose description is itself distributed on a communication network*. In this paper, we focus on symmetry breaking and thus ask:

How efficiently can H be colored when distributed on a network G?

Coloring problems are of fundamental importance to distributed graph algorithms. In fact, in its seminal paper [52], Linial studied the locality of 3-coloring cycles. A long line of work [52, 54, 61, 7, 45, 10, 60] showed that $\Delta+1$-coloring could be achieved in $\text{poly}(\log\log n)$ rounds of LOCAL. Further work extended the result to local list sizes [42], and small messages [29, 41, 44]. We extend these results to embedded graphs in nearly the same number of rounds while using local color lists (in a slightly weaker sense than in [42]).

1.1 Virtual Graphs

Before answering our research question, we clarify the meaning of *embedding* a graph H into a network G. We give here a high-level definition and expound on the formal definitions in Section 2. For clarity, we refer to $H = (V_H, E_H)$ as the input or virtual graph while $G = (V_G, E_G)$ is the communication network. We call elements of V_H vertices or nodes while elements of V_G are machines; elements of E_H are edges or conflicts while elements of E_G are links.

We set the definition of embedded virtual graphs forth by specifying which machine knows about which vertex and edge of H. Each vertex $v \in V_H$ is mapped to a set $V(v) \subseteq V_G$ of machines such that *vertices $u, v \in H$ are adjacent (in H) only if their support intersect*, i.e., $V(v) \cap V(u) \neq \emptyset$. We also assume that each support $V(v)$ is equipped with a spanning tree $T(v)$ (called support tree) that can be used to perform aggregation. We assume that machines $w \in V_G$ know about all the supports they belong to – the set of v such that $V(v) \ni w$ – as well as which support tree their adjacent links belong to. Each edge $uv \in E_H$ is mapped to a machine $w \in V(u) \cap V(v)$ in the intersection of the two nodes' supports, which knows about the existence of that edge. Figure 1 exemplifies such an embedding.

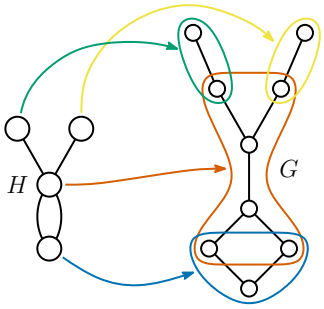

Figure 1 A virtual graph H (on the left) embedded on a network G (on the right). On this example, there is a unique choice of support trees; they have congestion $\mathsf{c} = 1$ and dilation $\mathsf{d} = 3$.

It is convenient to design algorithms for H as a sequence of (virtual) rounds with the same three-step structure[1]: first, broadcast a message to all vertices on the support; second, machines at intersections of supports perform local computations; third, converge-cast the

[1] we emphasize, however, that algorithms are not limited to this scheme and can communicate on the network more cleverly.

result of these computations on the support trees. Naturally, the efficiency of any such algorithm is limited by (1) the diameter of the support trees and (2) the number of trees using the same edge. We call the former the *dilation* and the latter the *congestion*. In some cases, most of the effort is in computing a good embedding, meaning with small enough dilation and congestion. For instance, in [22], the struggle is in finding $n^{o(1)}$-congestion embeddings for various sparsifiers. In this paper, besides direct applications, we assume the embedding is given as part of the input.

Last but not least, we allow H to be a *multi*-graph (without self-loops) to capture the fact that supports can intersect in multiple places. For instance, in Figure 1, the central vertex is adjacent to the bottom vertex through two paths in the network. While distinguishing between the number of incident edges and adjacent vertices is not always necessary, it is crucial for graph coloring, especially when – like in this paper – the number of colors used by each vertex depends on its degree.

1.2 Our Contributions

Our conceptual contribution is an explicit formalization of the notion of *virtual graphs* that captures the aforementioned examples. We show that the key parameters of congestion c and the dilation d essentially capture the hardness of the coloring problem. On one hand, they limit the efficiency of any $\deg+1$-coloring algorithm:

▶ **Theorem 1.** *Any constant-error algorithm for 3-coloring a 2-regular virtual graph H embedded on a network with bandwidth b, congestion c, and dilation d, requires $\Omega(\frac{\mathsf{c}}{\mathsf{b}} + \mathsf{d} \cdot \log^* n)$ rounds in the worst-case.*

We emphasize that the lower bound applies to algorithms working for any given embedding. It applies to all such algorithms, and not just those following the three-step process described in Section 1.1.

Conversely, we provide a nearly optimal upper bound for coloring virtual graphs. Applied to the CONGEST model – when $H = G$ – its complexity nearly matches the state-of-the-art $O(\log^3 \log n)$ round complexity of [41, 44].

▶ **Theorem 2.** *Let H be a virtual graph on network G with $|V_G| = n$ machines, bandwidth $\mathsf{b} = O(\log n)$, congestion $\mathsf{c} \leqslant n$ and dilation d. There exists an algorithm to $\deg+1$-color H in $O(\mathsf{cd} \cdot \log^4 \log n)$ rounds. More precisely, at the end of the algorithm, each vertex $v \in V_H$ has a color $\varphi(v) \in \{1, 2, \ldots, \deg(v) + 1\}$ where $\deg(v)$ is the number of edges incident to v in H.*

A key reason for considering the $\deg+1$-coloring problem is that we forgo using some frequently assumed global knowledge – here, the maximum degree Δ. This is the source of substantial technical challenges, sketched in Section 1.3. That virtual nodes can be connected with multiplicity breaks several classic arguments, hence requires novel ideas to reach the usual goals of providing nodes with excess colors, and classifying them according to their potential in that respect. Our adaptation of the Ghaffari-Kuhn algorithm (see the full version [21, Section 7]) to our distributed paradigm might be of independent interest.

1.3 Technical Overview

The Lower Bound. We prove lower bounds on the congestion and dilation separately. Since a $o(\mathsf{d}\log^* n)$ round algorithm for coloring virtual graphs implies a $o(\log^* n)$ round LOCAL algorithm for coloring cycles, the lower bound on the dilation follows from [52, 57]. To prove the lower bound on the congestion, we provide a probability distribution on gadgets (a 2-regular 16-vertex graph) where vertices are partitioned between two sets V_A and V_B. The gadget is such that if Alice (respectively Bob) knows all vertices and edges incident to V_A (respectively V_B), then for Alice and Bob to assign colors to their vertices such that the coloring is proper, they must communicate $\Omega(1)$ bits. A classic direct sum argument shows that solving k independent copies of this communication problem requires $\Omega(k)$ bits of communication. Finally, we embed the coloring problem on a graph where Alice's vertices are separated from Bob's through a bridge, causing congestion to be $\mathsf{c} = k$.

The Upper Bound: Inaccurate Degrees. The main challenge for coloring virtual graphs is that vertices do not have direct access to their list of available colors (or palette). Previous work [40, 19] demonstrated that it was not necessary if vertices could instead estimate certain local density parameters. While in [39, 19, 20] these density parameters were defined in term of Δ – the *globally known* maximum degree – in this paper, we assume no such global knowledge and aim to use local list sizes; hence, we require a different notion of local sparsity/density. We adapt our definition of embedding to encompass each vertex's local view of its degree. Concretely, we color a multi-graph H where each vertex uses one more color than it has incident edges. We call a vertex *inaccurate* if its number of incident edges is a constant factor larger than its number of adjacent neighbors. Inaccurate vertices require special treatment, for they can skew estimates of local sparsity. Since we use a number of colors dependent on the number of incident edges while each neighbor blocks at most one color, inaccurate vertices are always guaranteed to have an abundance of free colors. After detecting them, we defer coloring inaccurate vertices to the very end of the algorithm.

The Upper Bound: Providing Enough Colors. Every sublogarithmic randomized coloring algorithm [45, 10, 42] has three phases. First, they compute a partial coloring where each vertex has either *low degree* or *many excess colors compared to its uncolored number of neighbors.* Second, they use randomization and symmetry-breaking techniques to take advantage of this excess and color high-degree vertices ultrafast. Third, low-degree vertices are handled fast due to their low degree. In [45, 10, 42], the algorithm produces excess colors by a single-round randomized color trial. When vertices cannot access their palette [3, 19, 18], they resort to approximations that require generating more excess colors in the densest regions on the graph. We follow the same general approach with some major modifications. First, the use of local-list size partially breaks the analysis of slack generation from [41] (and the one of [42] cannot be implemented fast on virtual graphs). Our main technical contribution is to provide sufficient assumptions for a color trial algorithm to generate enough excess colors even when vertices can have small lists (Lemma 11, see [21, Section 5] of the full version). In general, these added assumptions introduce substantial modifications to the accounting of colors throughout the algorithm (see [21, Section 6.1] of the full version).

The Upper Bound: Low-Degree Vertices. Contrary to previous work [40, 19], all high-degree – larger than some $\text{poly}(\log n)$ – vertices are colored with high probability (rather than reducing uncolored degrees to $O(\log n)$). This implies that, for low-degree vertices, colors can be represented using $O(\log \deg) = O(\log\log n)$ bits. The algorithm for coloring

low-degree nodes follows the shattering framework of [7]. First, vertices try random colors for $O(\log \log n)$ rounds. This reduces the uncolored parts of the graph to $\text{poly}(\log n)$-sized components. Since nodes do not know their palette, we provide an algorithm for sampling colors likely-enough to succeed. Then, uncolored vertices learn a list of uncolored-degree+1 colors from their palette with an algorithm similar to a binary search. Finally, we simulate the deterministic algorithm of [35] efficiently and complete the coloring. Our main contributions – our algorithms for sampling colors and learning palettes – can be found in Sections 7.1 and 7.2 of the full version [21], respectively.

1.4 Related Work

Distributed coloring has been intensively studied. See, e.g., [52, 6, 61, 7, 46, 24, 45, 10, 56, 39, 35, 41, 42, 25] and references therein. The focus is usually on simple graphs, where the degree refers to the number of neighbors. The state-of-the-art LOCAL algorithm for degree+1-coloring (in terms of n only) is the $\tilde{O}(\log^2 \log n)$-round algorithm obtained by plugging the $\tilde{O}(\log^2 n)$-round deterministic algorithm of [27] into the shattering framework of [42]. In CONGEST, authors of [44] show how to implement shattering with small messages; hence, using the $O(\log^3 n)$-round deterministic algorithm of [35], the resulting complexity is $O(\log^3 \log n)$. Besides degrees being defined slightly differently, results of [42, 35, 27] are also more general in the sense that vertices can use *any list* of degree+1-colors (not necessarily $\{1, 2, \ldots, \deg(v) + 1\}$). Handling less constrained lists of colors in virtual graphs appears out of reach of current techniques; in fact, the problem has yet to be tackled in the simpler settings of cluster graphs and power graphs.

Virtual Graphs. Virtual graphs are ubiquitous in distributed graph algorithms and we make no attempt to be exhaustive. They refer to cases where the input graph differs from the communication network, though the formalism varies by use case. Here, we list occurrences of greatest relevance.

1. Many algorithms modify the input graph – e.g., by contracting an edge or removing a vertex and adding an edge between each neighbor – throughout the execution. This happens, e.g., in [33, 22]. In such cases, the algorithms embed the modified graph into the network while ensuring low congestion. Authors of [30, 59, 2] show that under some assumptions on the graph (e.g., planarity or excluded minor) then low-congestion shortcuts can be found efficiently, leading to drastic improvements on the round complexity.
2. Recent network decomposition algorithms [60, 29, 28] compute clusters – i.e., sets of vertices – by growing increasingly large sets of vertices. Hence, computations are held through the three-step aggregation process described in Section 1.1. That is, these algorithms are computing sequences of virtual graphs (including support trees) with $\text{poly}(\log n)$ dilation and congestion.
3. Finally, the local rounding framework introduced in [16] and perfected in [15] runs a defective-coloring subroutine on virtual graphs. They describe d2-multigraphs, a special case of virtual graphs used to implement their algorithm in CONGEST. They care for parallel edges since they compute a coloring, like us. Besides, their rounding algorithm has been used by network decomposition algorithms [28, 27] and thus had to be implemented on virtual graphs.

Our formalism for virtual graph captures all mentioned examples (with & without congestion, with & without parallel edges).

Scheduling & Routing. Congestion and dilation are natural parameters in routing problems, where they measure the maximum overlap and length of the delivery paths of a set of packets. Scheduling, in this context, refers to organizing the packets' delivery along their paths, taking into account congestion constraints. Naive scheduling leads to a $O(\mathsf{cd})$ delivery time, which can be hard to improve upon distributedly. Asymptotically optimal $\Theta(\mathsf{c}+\mathsf{d})$ schedules exist and can be computed efficiently given global knowledge of the paths [49, 50].

The routing literature is expansive and growing to this day [48, 51, 26, 32, 38]. While parallel delivery of information is crucial to our virtual graph algorithms, our problems are quite distinct from typical routing questions, as we usually aggregate and broadcast information rather than deliver it from a single source to a single target. In particular, we often change the information during its delivery. Even for our more complex tasks, a naive scheduling in $O(\mathsf{cd})$ remains possible. We leave open the question of whether the $O(\mathsf{cd})$ dependency can be improved to $O(\mathsf{c}+\mathsf{d})$ (see Problem 3 in Section 5 for more).

Power Graphs. Recently, there has been a growing interest in bandwidth-efficient algorithms for power graphs [39, 40, 5, 55, 19, 8]. Theorem 2 improves on previous work about distance-2 coloring [39, 40, 19] by handling a more general problem (see Section 2.1), by reducing the number of colors used by each vertex to its pseudo-degree (rather than, say, using Δ^2+1 colors which depends on a global parameter), and by improving the runtime by several $O(\log\log n)$ factors.

Other Models. The Congested Clique [53] can be viewed as a virtual graph model on the opposite end of the spectrum, where the communication graph is a clique. It has a $O(1)$-round deterministic algorithm for deg +1-list-coloring [11], building on similar results for $\Delta+1$-coloring [9, 12].

Sibling Paper. In a sibling paper [20], we treat cluster graphs, a particular type of virtual graphs, focusing on high-degree graphs. We give a $O(\log^* n)$-round algorithm for $\Delta+1$-coloring cluster graphs when $\Delta=\Omega(\log^{21} n)$. A key technical contribution is coloring so-called put-aside sets in extremely dense subgraphs, which we build on in this paper. That paper introduces essential primitives that apply to general virtual graphs, particularly operations on the communication backbone, including broadcast, aggregation, and palette queries. It also contains a fingerprinting technique for approximating the sizes of neighborhoods.

1.5 Outline of Paper

In the next section, we describe the modeling of virtual graphs and show how they capture two important settings. We present the main ideas behind the lower bound in Section 3. The high-level view of the algorithm is given in Section 4.3 along with key definitions, before describing some open questions in Section 5.

The detailed descriptions of various parts of the algorithm are deferred to the full version of this paper [21]. In [21, Section 5] we give a result on slack generation, generalizing previous arguments to $deg+1$-colorings (of both sparse and dense nodes). The coloring of different parts of the graph is split into several sections: the dense-but-not-too-dense part in [21, Section 6], the low-degree nodes in [21, Section 7], while the extremely-dense are in [21, Appendix C] as it builds heavily on the sibling paper [20]. The details of the lower bound are in [21, Appendix D]. Further appendices feature various algorithmic steps that are non-trivial adaptations or modifications of previous work, including almost-clique decomposition in [21, Appendix F].

1.6 Preliminaries

Mathematical Notation. For an integer $t \geqslant 1$, let $[t] \stackrel{\text{def}}{=} \{1, 2, \dots, t\}$. For a function $f : \mathcal{X} \to \mathcal{Y}$, when $X \subseteq \mathcal{X}$, we write $f(X) \stackrel{\text{def}}{=} \{f(x) : x \in X\}$; and when $Y \subseteq \mathcal{Y}$, we write $f^{-1}(Y) \stackrel{\text{def}}{=} \{x \in X : f(x) \in Y\}$. We abuse notation and write $f^{-1}(y) \stackrel{\text{def}}{=} f^{-1}(\{y\})$. For $X \subseteq \mathcal{X}$, we write $f_{|X} : X \to \mathcal{Y}$ for the restriction of f to X. Throughout the paper, we hide overhead due to congestion c and dilation d by writing $\widehat{O}(f)$ for $O(\mathsf{cd} \cdot f)$.

Graphs & Multi-Graphs. A multi-graph $H = (V_H, E_H)$ is defined by a set of vertices V_H and sets $E_H(u, v)$ describing all edges between u and v (and $E_H(u, v) = \emptyset$ if u and v are not adjacent). When each set $E_H(u, v)$ contains at most one edge (H has no parallel edges), we say the graph is ***simple***. The neighbors of v in H are $N_H(v) \stackrel{\text{def}}{=} \{u \in V_H : E_H(u, v) \neq \emptyset\}$. The ***pseudo-degree*** of v in H is $\deg(v; H) \stackrel{\text{def}}{=} \sum_{u \in V_H} |E_H(u, v)|$, its number of incident edges. Its ***degree*** counts its neighbors $|N_H(v)|$. When H is clear from context, we drop the subscript and write $N(v) = N_H(v)$ and $\deg(v) = \deg(v; H)$. An unordered pair $\{u, v\} \subseteq V_H$ is called an ***anti-edge*** or ***non-edge*** if $E_H(u, v) = \emptyset$.

Colorings. For any integer $q \geqslant 1$, a ***partial q-coloring*** is a function $\varphi : V_H \to [q] \cup \{\perp\}$ where $\perp$ means "not colored". The domain $\mathsf{dom}\,\varphi \stackrel{\text{def}}{=} \{v \in V_H : \varphi(v) \neq \perp\}$ of φ is the set of colored nodes. A coloring φ is ***total*** when all nodes are colored, i.e., $\mathsf{dom}\,\varphi = V_H$; and we say it is ***proper*** if $\perp \in \varphi(\{u, v\})$ or $\varphi(v) \neq \varphi(u)$ whenever $E_H(u, v) \neq \emptyset$. We write that $\psi \succeq \varphi$ when a partial coloring ψ ***extends*** φ: for all $v \in \mathsf{dom}\,\varphi$, we have $\psi(v) = \varphi(v)$. The ***uncolored degree*** $|N_\varphi(v)| \stackrel{\text{def}}{=} |N(v) \setminus \mathsf{dom}\,\varphi|$ of v with respect to φ is the number of uncolored neighbors of v. The ***uncolored pseudo-degree*** $\deg_\varphi(v)$ of v with respect to φ counts its number of incident edges to uncolored neighbors. The ***palette*** of v with respect to a coloring φ is $L_\varphi(v) = [\deg(v) + 1] \setminus \varphi(N(v))$, the set of colors we can use to extend φ at v.

2 Virtual Graphs

In distributed algorithmics, we consider ***communication graphs*** or ***networks*** $G = (V_G, E_G)$ where elements of V_G are ***machines*** that communicate by sending messages on incident ***links*** – unordered pairs of E_G – simultaneously in synchronous rounds. We assume machine $v \in V_G$ is provided a $O(\log |V_G|)$-bits unique identifiers ID_v to break symmetry. For randomized algorithms, they can also access local random bits. Messages are limited to b bits, where b is called the ***bandwidth*** of the network. Unless stated explicitly, it is assumed that $\mathsf{b} = \Theta(\log |V_G|)$.

We define our notion of virtual graphs formally. We shall always refer to the conflict graph by H and to the communication graph by G. Vertices/nodes and edges refer only to elements of H, while machines and links are used for G.

▶ **Definition 3** (Virtual Graph). *Let $G = (V_G, E_G)$ be a* simple *graph. A virtual graph on G is a* multi-*graph $H = (V_H, E_H)$ where each vertex $v \in V_H$ is mapped to a set $V(v) \subseteq V_G$ of machines called the* **support** *of v. Whenever two nodes are adjacent in H their supports intersect, i.e., if $E_H(u, v) \neq \emptyset$ then $V(u) \cap V(v) \neq \emptyset$. Each machine $w \in V_G$ knows the set $V^{-1}(w)$ of vertices whose supports contains it.*

When bandwidth is not an issue, we can work directly with the representation of Definition 3. We can compute a breadth-first spanning tree $T(v) \subseteq E_G$ on each support $V(v)$ for distributing information, and then simulate a local algorithm on this support structure. With bandwidth constraints, we need to be more careful.

▶ **Definition 4** (Embedded Virtual Graph)**.** *Let H be a virtual graph on G such that $|V_H| \leqslant \text{poly}(|V_G|)$. Suppose that (1) for each vertex $v \in V_H$, there is a tree $T(v) \subseteq E_G$ spanning $V(v)$; and (2) for each edge $e \in E_H(u, v)$ there is a machine $m(e) \in V(u) \cap V(v)$. We call $T(v)$ the* **support tree** *of v and $m(e)$ the machine* **handling** *edge e. Each machine w knows the set of edges $m^{-1}(w)$ it handles as well as, for each incident link $\{w, w'\} \in E_G$, the set $T^{-1}(ww')$ of support trees it belongs to.*

Given support trees, it is convenient to design our algorithms as a sequence of rounds each consisting of a three-step process: broadcast, local computation on edges, followed by converge-cast. We use two parameters to quantify the overhead cost of aggregation on support trees. The ***congestion*** c of H is the maximum number of trees using the same link. The ***dilation*** d is the maximum height of a tree $T(v)$ in G. Formally,

$$\mathsf{c} \stackrel{\text{def}}{=} \max_{e \in E_G} |T^{-1}(e)| \quad \text{and} \quad \mathsf{d} \stackrel{\text{def}}{=} \max_{v \in V_H} \left(\max_{u \in T(v)} \text{dist}_{T(v)}(v, u) \right). \tag{1}$$

Congestion and dilation are natural bottlenecks for virtual graphs. In Theorem 1, we show that $\Omega(\mathsf{c}/\mathsf{b} + \mathsf{d} \log^* n)$ rounds are needed for our coloring task given b bandwidth in the communication graph. Conversely, Theorem 2 shows that coloring in $O(\mathsf{cd} \cdot \log^4 \log n)$ rounds is feasible for any embedding.

▶ Remark 5. A few remarks are in order.

1. The degrees in H can be computed as $\deg(v) = \sum_{w \in T(v)} |m^{-1}(w)|$ by aggregation on support trees, which is why we ask that edges have designated handlers. Counting exactly the number of distinct neighbors for all vertices appears to be challenging (i.e., requires $\widetilde{\Omega}(|N_H(v)|)$ rounds).
2. By running a BFS from a single source (or from multiple sources but in *disjoint* subgraphs), we can count the exact the number of neighbors the source has. However, running this algorithm from multiple vertices creates congestion proportional to that number of vertices.
3. It is, per se, easy to compute low-diameter support trees for all vertices, e.g., by BFS, but a poor selection of edges could easily lead to high congestion. It is an open question if trees of both low diameter and congestion can be computed efficiently (see Section 5).

2.1 Implications

Our framework captures several models and problems studied in the distributed graph literature. We review them quickly.

It is helpful to see the communication network $G = (V_G, E_G)$ through its ***subdivision graph***: the bipartite graph $S_G = (V_G, E_G, \{\{u, e\} : u \in e \in E_G\})$ with machines on the left, links on the right, and a link between $v \in V_G$ and $e \in E_G$ if and only if v is an endpoint of e. Simulating a round of communication on S_G takes one round of communication of G (conversely, one round on G takes two rounds of S_G). Defining our virtual graphs on S_G rather than G allows us to put conflict on links. We call the links of S_G ***half-links***. [2]

[2] A common alternative representation is to stipulate that edges are between vertices with adjacent supports, i.e., $uv \in E_H$ implies that $\exists w \in V(v), x \in V(u)$ s.t. $wx \in E_G$. If we extend each support $V(v)$ in G to include also the incident link nodes in S_G, then two supports in S_G intersect whenever they are adjacent in G. Hence, our formulation encompasses this variant.

2.1.1 Application 1: Cluster Graphs

A ***cluster graph*** $\mathcal{C}$ on a communication graph $G = (V_G, E_G)$ is a graph where vertices are disjoint sets $C_x \subseteq V_G$ called ***clusters*** with a designated machine $\mathsf{leader}(x) \in C_x$. Each cluster C_x induces a connected graph of small diameter in G. Two clusters are adjacent if and only if they are connected by a link. A round of communication on H consists of 1) broadcasting a b-bit message from $\mathsf{leader}(x)$ to all machines in C_x, 2) communication on inter-cluster links, and 3) aggregate a $\operatorname{poly}\log n$-bit message (e.g., a sum or a min) to $\mathsf{leader}(x)$. They appear in several places, from maximum flow algorithms [33, 22] to network decomposition [60, 29].

Clearly, cluster graphs are captured by our definition of virtual graphs: for $C_x \in V_H$, let $V(C_x)$ be C_x plus the half-links going out of C_x and $T(C_x)$ be a BFS tree spanning $V(C_x)$. Theorem 2 implies we can color cluster graphs fast:

▶ **Corollary 6.** *Cluster graphs can be* $\deg+1$*-colored, w.h.p., in* $O(\mathsf{d} \cdot \log^4 \log n)$ CONGEST *rounds where* d *is the maximum (strong) diameter of a cluster, i.e., of* $H[C_x]$ *over all* C_x*.*

In [20], we show that $\Delta+1$-coloring high-degree cluster graphs (where $\Delta \geqslant \operatorname{poly}(\log n)$) can be done in $O(\log^* n)$ rounds. Corollary 6 is the first non-trivial distributed algorithm for degree+1-coloring cluster graphs.

2.1.2 Application 2: Coloring Power Graphs

For some integer $t \geqslant 1$, the ***t-th power graph*** of G is the graph G^t on vertices V_G where there is an edge $\{u, v\}$ when $\operatorname{dist}_G(u, v) \leqslant t$. A ***distance-$t$ coloring*** of G is a coloring of G^t. Concretely, it is a coloring where nodes receive colors different from the ones in their t-hop neighborhood. Our framework provides a unified view of distance-t colorings: the same algorithm provides fast algorithms for all values of $t \geqslant 1$.

▶ **Lemma 7.** *Let* $t \geqslant 1$ *and* $G = (V_G, E_G)$ *be a graph with maximum (distance-1) degree* Δ*. We can define a virtual graph* $H = (V_H, E_H)$ *on the subdivision graph* S_G *of* G *such that* $V_H = V_G$ *and a* $\deg+1$*-coloring of* H *is a* $\Delta^t + 1$*-coloring of* G^t*. Moreover, the congestion is* $\mathsf{c} = O(\Delta^{\lfloor \frac{t-1}{2} \rfloor})$*, the dilation is* $\mathsf{d} = t$*, and the embedding is computable in* $O(t\mathsf{c})$ *rounds.*

Proof. For each node $v \in V_G$, its support tree $T(v)$ in G is set to be the *t-hop BFS tree in the subdivision graph S_G rooted at v*. For any pair $u, v \in V_H$, the edge set $E_H(u, v) = \emptyset$ if and only if $\operatorname{dist}_G(u, v) > t$. Otherwise, $E_H(u, v)$ contains an edge for each *simple uv-paths* in $T(u) \cup T(v)$ in G. As there are at most $\sum_{i=1}^{t-1} \Delta(\Delta-1)^i \leqslant \Delta^t$ simple paths of length at most t starting from v in G. Hence, each vertex is incident to at most Δ^t edges in H. Thus, any $\deg+1$-coloring on H is a distance-t coloring of G with $\Delta^t + 1$ colors and from the definition of edges in H, a proper coloring on H is also proper on G^t.

The bound on the dilation is immediate. To verify the congestion on a half-link ev, observe that there are at most $\Delta^{\lfloor \frac{t-1}{2} \rfloor}$ nodes (of G) that are within distance t of v in S_G, and therefore at most that many support trees using that half-link.

We map each simple uv-path in $T(u) \cup T(v)$ to its middle machine in S_G. It is unique, as S_G is bipartite and u, v are on the same side. Each machine $w \in T(u) \cap T(v)$ knows its distances to u in $T(u)$ and to v in $T(v)$, and thereby knows if it is the middle machine. To compute the embedding, we have each machine learn its distance-t neighborhood in S_G, with the distance it has to each machine in it. This is done as follows: initially, each machine v prepares a message $(\mathsf{ID}_v, 1)$, which it sends to its Δ direct neighbors in G. Then, for each positive integer $i \leqslant \lfloor \frac{t-1}{2} \rfloor$, each machine sends a message of the form $(\mathsf{ID}_u, i+1)$ to its direct

neighbors for each message (ID_u, i) it has received, of which there are at most Δ^i. Sending all messages for a fixed i takes $O(\Delta^{\lfloor \frac{t-1}{2} \rfloor}) = O(\mathsf{c})$ rounds, hence a total $O(t\mathsf{c})$ complexity. At the end of this process, each machine v knows to which support trees $T(u)$ it belongs, and for each simple path of length at most t in S_G between two nodes u, u' s.t. $v \in T(u) \cap T(u')$, v knows whether it is its midpoint and should thus handle the edge. ◀

For any $t \geqslant 1$, Theorem 2 and Lemma 7 imply that there is a distributed algorithm communicating on S_G with $O(\log n)$ bandwidth that computes a $\Delta^t + 1$-coloring of G^t. Since a round of communication on S_G can be simulated in one round of communication on G, it shows the following corollary.

▶ **Corollary 8.** *For any $t \geqslant 1$, there is a randomized* CONGEST *algorithm for Δ^t+1-coloring G^t that runs in $O(t\Delta^{\lfloor (t-1)/2 \rfloor} \log^4 \log n)$ rounds w.h.p.*

Furthermore, the specific structure of power graphs allows for broadcast and aggregation over support trees to be done in only $O(\Delta^{\lfloor (t-1)/2 \rfloor}) = O(\mathsf{c} + \mathsf{d})$ rounds instead of $O(\mathsf{cd}) = O(t\Delta^{\lfloor (t-1)/2 \rfloor})$. The runtime in Corollary 8 can be improved to $O(\Delta^{\lfloor (t-1)/2 \rfloor} \log^4 \log n)$ as a result.

It is not too difficult to see that – by a reduction to set disjointness – verifying an *arbitrary (or random)* distance-t coloring needs $\tilde{\Omega}(\Delta^{\lfloor \frac{t-1}{2} \rfloor})$ rounds in CONGEST [23]. However, no super-constant lower bounds are known for computing distance-t colorings in CONGEST when $t \geqslant 3$ and $\Delta \gg \log n$.

3 Lower Bounds: Overview

In this section, we sketch the main ideas behind our lower bound. We show the following theorem:

▶ **Theorem 1.** *Any constant-error algorithm for 3-coloring a 2-regular virtual graph H embedded on a network with bandwidth b, congestion c, and dilation d, requires $\Omega(\frac{\mathsf{c}}{\mathsf{b}} + \mathsf{d} \cdot \log^* n)$ rounds in the worst-case.*

This implies as immediate corollary the same lower bound for the more general problem of deg +1-coloring virtual graphs. The single statement is actually the concatenation of two independent lower bounds, one relative to congestion and bandwidth, and the other relative to dilation.

The dilation lower bound is straightforward, following directly from the seminal $\Omega(\log^* n)$ lower bounds on 3-coloring [52, 57]. We refer readers to Appendix D.2 of the full version [21].

As the congestion lower bound makes lengthy use of technical tooling from communication complexity literature largely unrelated to the rest of the paper, we defer most details to Appendix D of the full version [21]. Here, we chiefly describe the virtual graphs used for our lower bound and give intuition behind the complexity of coloring them.

Proof Structure of the Congestion Lower Bound. The congestion-related part of our lower bound is obtained through a reduction from communication complexity. Our overall proof plan is as follows:

- We introduce a 2-player communication complexity task in which said players must coordinate to 3-color a 16-node 2-regular graph. Each player only knows the edges incident to half of the vertices and is in charge of outputting half of the coloring.
- We show that this task is nontrivial, and in particular, that it has $\Omega(1)$ information complexity, a complexity measure which lower bounds communication complexity.

- From known direct-sum results on information complexity, we get that solving $\mathsf{c}/8$ independent copies of the task has information complexity $\Omega(\mathsf{c})$.
- We introduce a virtual graph of congestion c and constant dilation in which we embed $\mathsf{c}/8$ instances of the task, i.e., deg +1-coloring the virtual graph solves the $\mathsf{c}/8$ instances.
- We observe: any T-round algorithm for deg +1-coloring virtual graphs over communication graphs with congestion c given bandwidth b implies a $O(T\mathsf{b})$ communication complexity algorithm for solving $\mathsf{c}/8$ copies of the nontrivial task.
- We conclude: the round complexity T of any such distributed algorithm for deg +1-coloring must necessarily be at least $\Omega(\mathsf{c}/\mathsf{b})$.

The Communication Complexity Gadget. We define the communication complexity task we use in Definition 9. While this definition is a generalization with an arbitrary even number of nodes on both sides, for our purposes, we will only use the gadget with the parameter k set to $k = 4$, i.e., with 8 nodes on Alice and Bob's sides. See Figure 2 for a illustration of our gadget.

▶ **Definition 9** (Matching 3-coloring task). *In the* $\mathsf{M3COL}_k$ *task, a* $4k$*-node graph is initially uncolored. Its nodes are split into two equal parts – left and right – given to Alice and Bob. Alice and Bob receive a perfect matching over their respective sets of* $2k$ *nodes. For each* $i \in [2k]$*, the* i*th left node is connected to the* i*th right node. At the end of the communication protocol, Alice must output a color in* $\{1, 2, 3\}$ *for each left node, and Bob must do the same for the right nodes, such that the coloring is valid with respect to the graph received as input.*

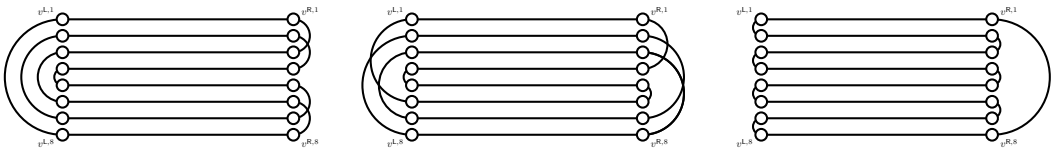

Figure 2 Three possible inputs to the communication complexity task.

The crux of the argument is to show that the $\mathsf{M3COL}_4$ task cannot be solved without communication. This can be intuitively seen by noticing that there can be at most 3 nodes on which Alice *always* outputs the same color regardless of her input matching (same on Bob's side). Indeed, as there are only 3 colors, a fourth node with a fixed color means that two nodes would receive the same color on Alice's side regardless of her matching. This implies an error when said two nodes are connected in Alice's matching. Generalizing this idea to randomized algorithms allows us to show that an algorithm without communication necessarily makes an error with some constant probability [3] .

▶ **Lemma 10.** *Any zero communication protocol for* $\mathsf{M3COL}_4$ *fails with probability at least* $\frac{1}{196}$ *over the uniform input distribution.*

Embedding the Gadget. Embedding the gadgets into a virtual graph is then done with the following communication network: we consider two stars (depth-1 trees) with c leaves; we connect the two stars by a single link between their roots $w^{\mathsf{L,com}}$ and $w^{\mathsf{R,com}}$. The support of each node on the left is made of an edge of the left star with the central edge, while the

[3] This intuition also explains why we take gadgets with 8 nodes on each side and not less: a smaller gadget would be solvable without communication by fixing the color of (up to) 3 nodes on each side.

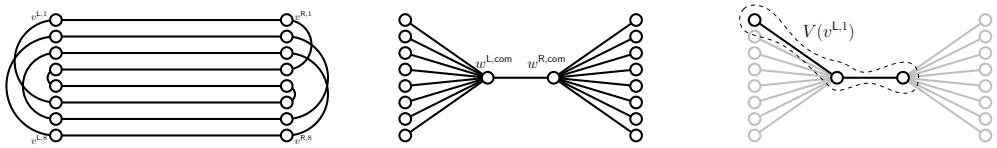

Figure 3 Examples of a virtual graph H with a single gadget (left), a communication network G (middle) in which H can be embedded, and the support of the top left virtual node (right).

support of each node on the right is just an edge in the right star. $w^{\mathsf{L,com}}$ handles the edges in the left matching, while $w^{\mathsf{R,com}}$ handles the edges of the right matching as well as the edges between the left and right sides of the virtual graph. See Figure 3 for an illustration.

The proof of Lemma 10, with its implication for the information complexity of the task, and ultimately, our $\Omega(\mathsf{c}/\mathsf{b})$ lower bound for 3-coloring graphs of degree 2 (Theorem 1), are all deferred to Appendix D.1 of the full version [21].

4 Coloring Algorithm

The goal of this section is to present the main technical ideas behind Theorem 2.

▶ **Theorem 2.** *Let H be a virtual graph on network G with $|V_G| = n$ machines, bandwidth $\mathsf{b} = O(\log n)$, congestion $\mathsf{c} \leqslant n$ and dilation d. There exists an algorithm to $\deg+1$-color H in $O(\mathsf{cd} \cdot \log^4 \log n)$ rounds. More precisely, at the end of the algorithm, each vertex $v \in V_H$ has a color $\varphi(v) \in \{1, 2, \ldots, \deg(v) + 1\}$ where $\deg(v)$ is the number of edges incident to v in H.*

We give necessary definitions and self-contained statements of each of the main steps of our algorithm. First, we discuss the concept of *slack* and present the means by which we measure and produce it. We then introduce a version of the sparse-dense decomposition tailored to our needs. Finally, we describe the main steps of our algorithm. For more details on individual steps of the algorithm, we refer readers to the full version of this paper [21].

4.1 Slack

Intuitively, the slack measures how easily a vertex gets colored. More formally, it is used to bound from below the number of colors available to a vertex when its neighbors are trying to get colored. There are several types of slack that occur.

Savings. Whenever a neighbor uses a color that is either outside v's palette or the same color as another neighbor, then v *saves* a color. Under a given partial coloring φ, this is quantified by the ***savings slack*** of v from coloring φ:

$$\xi_\varphi(v, S) \stackrel{\text{def}}{=} |S \cap \mathsf{dom}\,\varphi| - |\varphi(S) \cap [\deg(v) + 1]| \tag{2}$$

We write $\xi_\varphi(v)$ for $\xi_\varphi(v, N(v))$.

Redundancy. In degree+1-coloring (unlike $\Delta + 1$-coloring), slack can also occur when v has a shortage of neighbors with a high enough degree. We measure this with the ***redundancy*** of v defined as

$$\rho_v \stackrel{\text{def}}{=} \max_{t \leqslant |N(v)|/12} |N_H(v)| - t - |\{u \in N(v) : \deg(u) + 1 > t\}| \ . \tag{3}$$

In other words, there is a $t \leqslant |N(v)|/12$ such that, even if all high-degree neighbors (larger than t) use different colors, at least ρ_v colors remain available to v.

Inaccuracy. The difference between the palette size and the number of neighbors is the ***inaccuracy*** of the node:

$$\delta_v = \deg(v) - |N_H(v)| \ . \tag{4}$$

In our setting, this is caused by parallel edges. A vertex with $\delta_v > \delta$ is called δ-***inaccurate*** and δ-***accurate*** otherwise.

Permanent & Temporary slack. The aforementioned forms of slack (savings, redundancy, and inaccuracy) are ***permanent***, meaning that they do not decrease as we extend the coloring. Another way to provide slack to a vertex is by keeping some of its uncolored neighbors inactive. This artificially reduces degrees – thus contention – without reducing the number of available colors, thereby providing slack. This is called ***temporary slack*** as it perishes when we eventually color the inactive neighbors.

Slack Generation. While redundancy and inaccuracies do not depend on the coloring, vertices get savings only if we manage to same-color its neighbors. We show in [21, Section 5] that a classic one-round algorithm of "trying a random color" creates enough slack for deg+1-colorings. This generalizes results for $\Delta+1$-coloring [58, 14, 41]. It also generalizes a method of [1, Lemma 4.10] for deg+1-coloring that applies to the sparse and uneven nodes (assuming $\deg(v) = |N_H(v)|$). `SlackGeneration` creates color savings probabilistically. The savings expected from a random color trial are measured by the unevenness and sparsity, which we now define.

The savings we expect due to high-degree neighbors using colors beyond $\deg(v)+1$ is captured by the ***unevenness*** of v. Within a subgraph induced by a set $S \subseteq V_H$, it is defined as

$$\eta_v(S) \stackrel{\text{def}}{=} \sum_{u \in S} \frac{[\deg(u)+1] \setminus [\deg(v)+1]}{[\deg(u)+1]} = \sum_{u \in S} \frac{(\deg(u) - \deg(v))^+}{\deg(u)+1} \ . \tag{5}$$

We write $\eta_v = \eta_v(N(v))$ for succinctness. A vertex such that $\eta_v > \eta$ is called η-***uneven*** and η-***balanced*** otherwise.

The savings we expect from colors reused by multiple neighbors is quantified by the ***sparsity*** of v. The sparsity of v is defined as

$$\zeta_v \stackrel{\text{def}}{=} \frac{1}{|N_H(v)|} \left(\binom{|N_H(v)|}{2} - \frac{1}{2} \sum_{u \in N_H(v)} |N_H(u) \cap N_H(v)| \right) \ . \tag{6}$$

Note that $\frac{1}{2}\sum_{u \in N_H(v)} |N_H(u) \cap N_H(v)|$ counts the number of edges in $N_H(v)$ *without multiplicity*, even if H is not simple. Hence $\zeta_v \cdot |N_H(v)|$ counts the number of edges missing in $N_H(v)$, without multiplicity. A vertex such that $\zeta_v > \zeta$ is called ζ-***sparse*** and ζ-***dense*** otherwise.

▶ **Lemma 11** (Slack Generation). *Let $V_{\mathsf{sg}} \subseteq V_H$ and let φ_{sg} be the coloring produced by running Algorithm 2 in $H[V_{\mathsf{sg}}]$ avoiding colors $\leqslant r$. Let $v \in V_{\mathsf{sg}}$ be a node satisfying $\deg(v) \leqslant 3|N_H(v)|/2$, $|N(v) \setminus V_{\mathsf{sg}}| < (\zeta_v + \eta_v)/4$, $\zeta_v \geqslant 48r$, and $\rho_v \leqslant (\zeta_v + \eta_v)/12$. Then*

$$\xi_{\varphi_{\text{sg}}}(v) \geqslant \gamma_{11} \cdot (\zeta_v + \eta_v) \quad \textit{with probability} \quad 1 - \exp(-\Theta(\zeta_v + \eta_v))$$

4.2 Almost-Clique Decomposition

In Lemma 12, we describe a structural decomposition partitioning vertices according to their ability to receive slack and of which type. All sub-logarithmic distributed coloring algorithms [45, 10, 42, 20] use such a decomposition. We adapt [1] to account for inaccuracies in degrees (Property 2). Lemma 12 partitions vertices into high- and low-degree vertices based on the threshold $\Delta_{\mathsf{low}} = \Theta\big(\log^{21} n\big)$. Each requires a different approach and, in particular, we do not need to argue that low-degree vertices receive slack. We prove Lemma 12 in [21, Appendix F] to preserve the flow of the paper.

▶ **Lemma 12.** *There exists an algorithm that, for any multi-graph $H = (V_H, E_H)$ and $\varepsilon \in (0, 1/100)$, computes in $\widehat{O}(1/\varepsilon^6)$ rounds an ε-***almost-clique decomposition***: a partition $V_H = V_{\mathsf{low}} \cup V_{\mathsf{high}}$ and $V_{\mathsf{high}} = V_{\mathsf{in}} \cup V_\star \cup V_{\mathsf{dense}}$ such that*

1. *each $v \in V_{\mathsf{low}}$ has $\deg(v) \leqslant 2\Delta_{\mathsf{low}}$ and $v \in V_{\mathsf{high}}$ has $\deg(v) \geqslant \Delta_{\mathsf{low}}$;*
2. *each $v \in V_{\mathsf{in}}$ is $\Omega(\varepsilon^3|N(v)|)$-inaccurate and each $v \in V_{\mathsf{high}} \setminus V_{\mathsf{in}}$ has $\deg(v) \leqslant (1+\varepsilon^3)|N(v)|$;*
3. *each $v \in V_\star$ has $\zeta_v + \eta_v + |N(v) \cap V_{\mathsf{in}}| \geqslant \gamma_{12} \cdot \deg(v)$ for a constant $\gamma_{12} = \gamma_{12}(\varepsilon) \in (0, 1)$;*
4. *V_{dense} is partitioned into ε-almost-cliques: sets $K \subseteq V_{\mathsf{high}}$ such that*
   a. *$|N_H(v) \cap K| \geqslant (1-\varepsilon)|K|$, for each $v \in K$,*
   b. *$\deg(v) \leqslant (1+\varepsilon)|K|$, for each $v \in K$, and*
   c. *$|N_H(v) \setminus K| \leqslant O_\varepsilon(\zeta_v + \eta_v + |N(v) \cap V_{\mathsf{in}}|)$.*

Let $\Delta_K \stackrel{\text{def}}{=} \max_{v \in K} \deg(v)$. From Lemma 12, it holds for each almost-clique K that $(1-\varepsilon)|K| \leqslant \Delta_K \leqslant (1+\varepsilon)|K|$, and that for every $v \in K$, $\deg(v) \geqslant (1-2\varepsilon)\Delta_K$. Every pair of vertices in K has $(1-2\varepsilon)|K|$ neighbors in common in K, and hence $H[K]$ has (strong-)diameter at most two.

For $v \in V_{\mathsf{dense}}$, let K_v denote the almost-clique containing v. We denote by $A_v = K_v \setminus N_H(v)$ its ***anti-neighborhood*** and by $a_v = |K_v| - \deg(v, H \cap K_v)$ its ***pseudo-anti-degree***. We call $E_v = N_H(v) \setminus K_v$ the ***external-neighborhood*** and $e_v = \deg(v, H \setminus K_v)$ its ***pseudo-external-degree***. Importantly, *pseudo-external and pseudo-anti-degrees count multiplicities of edges in the conflict graph.* We split the contribution to δ_v (Equation (4)) between external and internal neighbors:

$$\delta_v = \delta_v^e + \delta_v^a, \quad \text{where} \quad \delta_v^e \stackrel{\text{def}}{=} e_v - |E_v| \quad \text{and} \quad \delta_v^a \stackrel{\text{def}}{=} |A_v| - a_v \ . \tag{7}$$

For almost-clique K, we denote average values by $a_K = \sum_{v \in K} a_v/|K|$ and $e_K = \sum_{v \in K} e_v/|K|$.

4.3 The High-Level Algorithm

We can now describe the main steps of our algorithm. At high level, we compute the decomposition of Lemma 12, run slack generation in $V_{\mathsf{sg}} = V_{\mathsf{high}} \setminus (V_{\mathsf{cabal}} \cup V_{\mathsf{in}})$ and color each part of the decomposition in a precise order. Necessary conditions and guarantees for each step of Algorithm 1 are given in the corresponding propositions.

Parameters. Let C_1 be some large universal constant. Let us set the following parameters

$$\varepsilon = 1/2000 \ , \quad \ell = C_1 \cdot \log^{1.2} n \ , \quad \text{and} \quad r = C_1 \cdot \log^{1.1} n \ , \tag{8}$$

where ℓ is chosen to asymptotically dominate $\Theta(\log^{1.1} n)$, which is the minimum palette size for MultiColor Trial, and r sets the number of reserved colors. We call colors from $[r] = \{1, 2, \ldots, r\}$ reserved because we use them exclusively for multicolor trials. Let $V_{\mathsf{low}}, V_{\mathsf{in}}, V_\star, V_{\mathsf{dense}}$ be an ε-almost-clique decomposition of the high-degree vertices. We define

$$\mathcal{K}_{\mathsf{cabal}} = \{K : e_K < \ell\} \ , \quad V_{\mathsf{cabal}} = \{v \in V_{\mathsf{dense}} : K_v \in \mathcal{K}_{\mathsf{cabal}}\} \quad \text{and} \quad V_{\mathsf{sg}} = V_{\mathsf{high}} \setminus (V_{\mathsf{cabal}} \cup V_{\mathsf{in}}) \ .$$

Algorithm 1 The deg +1-coloring algorithm.

1	`ComputeACD`	(Lemma 12)
2	`SlackGeneration` in $V_{\mathsf{sg}} = V_{\mathsf{high}} \setminus (V_{\mathsf{cabal}} \cup V_{\mathsf{in}})$ without using colors $[r]$	(Lemma 11)
3	`ColoringVstar` without using colors $[r]$	(Proposition 13)
4	`ColoringNonCabals`	(Proposition 14)
5	`ColoringCabals`	(Proposition 15)
6	`ColoringInaccurate`	(Proposition 16)
7	`ColoringLowDegree`	(Proposition 17)

After running Slack Generation in V_{sg}, w.h.p., all the vertices in $V_\star$ have enough slack to get colored by MultiColor Trial. Proposition 13 achieves this coloring with additional post-conditions necessary for coloring non-cabals (Proposition 14). In words, we extend the coloring φ_{sg} produced by slack generation such that $V_\star$ is totally colored, the coloring in $V_H \setminus V_\star$ coincides with φ_{sg} and reserved colors are not used (not even in $V_\star$). Proof of Proposition 13 is given in [21, Section 4].

▶ **Proposition 13** (Coloring $V_\star$). *Suppose φ_{sg} is the coloring produced by slack generation. In $\widehat{O}(\log^* n)$ rounds, we compute $\varphi \succeq \varphi_{\mathrm{sg}}$ such that, w.h.p., we have $V_\star \subseteq \mathsf{dom}\,\varphi$, $\varphi_{|V_H \setminus V_\star} = \varphi_{\mathrm{sg}|V_H \setminus V_\star}$ and $\varphi(V_H) \cap [r] = \emptyset$.*

Non-cabal dense vertices are colored by Algorithm 3 in [21, Section 6]. They are colored immediately after coloring $V_\star$, and the conditions needed for Proposition 14 follow from those guaranteed by Proposition 13. The algorithm combines primitives from various recent randomized coloring algorithms [42, 17, 19] (and [20]), all needing non-trivial adaptation to the current setting. Instead of applying MultiColor Trials directly after the synchronized color trial, we use the slower $O(\log\log n)$-round Slice Color algorithm of [19] to find an orientation where nodes have $O(\log n)$ uncolored out-neighbors. This allows us to use a fixed number of only $r = \Theta(\log^{1.1} n)$ reserved colors in the final application of MultiColor Trials, simplifying the (already intricate) treatment. Finally, a significant effort is needed to add up all sources of slack and show that dense vertices always have enough colors in the clique palette (see [21, Section 6.1]).

▶ **Proposition 14** (Coloring Non-Cabals). *Suppose φ is a coloring such that $\mathsf{dom}\,\varphi \subseteq V_{\mathsf{sg}}$, $\varphi_{|V_{\mathsf{dense}}} = \varphi_{\mathrm{sg}|V_{\mathsf{dense}}}$ and $\varphi(V_H) \cap [r] = \emptyset$. In $\widehat{O}(\log\log n \cdot \log^* n)$ rounds, we color all vertices in $V_{\mathsf{dense}} \setminus V_{\mathsf{cabal}}$.*

Cabals are colored by Algorithm 4 in [21, Appendix C]. The approach to color V_{cabal} is similar to Proposition 14 except for two major differences. First, vertices do not receive slack from slack generation, so we instead resort to *put-aside sets* [42]. Second, coloring put-aside sets requires a different approach that was developed in [20].

▶ **Proposition 15** (Coloring Cabals). *Suppose φ is a coloring such that $V_{\mathsf{cabal}} \cap \mathsf{dom}\,\varphi = \emptyset$. Then, there exists a $\widehat{O}(\log\log n \cdot \log^* n)$-round algorithm coloring all nodes in V_{cabal} with high probability.*

The inaccurate nodes have enough slack regardless of the coloring (Equation (4)) and are easily colored at the end in the same way as $V_\star$.

▶ **Proposition 16** (Coloring Inaccurate Nodes). *We can color all vertices in V_{in} in $\widehat{O}(\log^* n)$ rounds.*

Proof Sketch. The inaccuracy means that each vertex v in V_{in} has $\Omega(\varepsilon^3 \deg(v))$ colors available in $[\deg(v)+1]$ under any (possibly partial) coloring. Like for $V_\star$, we color V_{in} with $O(\varepsilon^{-12}\log\varepsilon^{-1}) = O(1)$ iterations of Random Color Trial and $O(\log^* n)$ iterations of MultiColor Trial where $\mathcal{C}(v) = [\deg(v)+1]$ and $\gamma = \Theta(\varepsilon^3)$. ◀

Low-degree nodes are colored in [21, Section 7].

▶ **Proposition 17** (Coloring Low-Degree Nodes). *Suppose φ is a coloring such that $V_{\mathsf{high}} = V_H \setminus V_{\mathsf{low}} = \mathsf{dom}\,\varphi$. In $\widehat{O}(\log^4 \log n)$ rounds, we compute a total coloring of H.*

Proof of Theorem 2. By Lemma 12, we compute the ε-almost-clique decomposition in $\widehat{O}(1/\varepsilon^6) = \widehat{O}(1)$ rounds. Running Slack Generation takes $\widehat{O}(1)$ rounds (see Algorithm 2). By Propositions 13–15, we extend the coloring to all vertices of V_{high} in $\widehat{O}(\log\log n \cdot \log^* n)$ rounds. By Proposition 17, low-degree vertices are colored in $\widehat{O}(\log^4 \log n)$ rounds. Overall, the round complexity is dominated by the coloring of low-degree vertices. ◀

5 Open Problems

The most natural immediate question following our work is:

▶ **Problem 1.** *Can we color virtual graphs in $\mathsf{cd}\cdot\mathrm{poly}(\log\log n)$ rounds using lists $\{1,2,\dots,|N_H(v)|+1\}$ for each $v \in V_H$?*

The issue is with dense vertices whose anti-degree is hard to approximate accurately. In [20], we show that it is possible to $\Delta+1$-color in $\widehat{O}(\log^* n)$ rounds when $\Delta = \max_v |N_H(v)|$ is the maximum number of neighbors (and $\Delta \gg \log^{21} n$). However, whether the technique used to approximate anti-degrees can be generalized to $|N(v)|+1$-coloring is unclear. Using MultiColor Trials, it is possible to $(1+\varepsilon)|N_H(v)|$-color in $\mathsf{cd}\cdot\mathrm{poly}(\log\log n)$ rounds.

▶ **Problem 2.** *When is it possible to compute low-congestion support trees efficiently?*

We assumed that a support tree was given in G for each node of H (or could be easily deduced, as in the case of distance-2 coloring). It is easy, per se, to find some support tree for each node, e.g., by BFS, but this could significantly affect the congestion. It is known [30, 37, 47, 31] that for some families of graph, one can compute embeddings with low congestion. Conversely, for some problems (such as MST), on general graphs $\Omega(\sqrt{n})$ congestion is unavoidable [13]. It is a highly interesting question whether low-congestion support trees could be computed efficiently for local problems.

▶ **Problem 3.** *Can we color virtual graphs in $O((\mathsf{c}+\mathsf{d})\,\mathrm{poly}(\log\log n))$ rounds?*

Throughout the paper, our main goal was showing that coloring can be achieved in $\mathrm{poly}(\log\log n)$ rounds of broadcast and aggregation over the supports of the virtual nodes. We mostly ignored the runtime of these broadcast and aggregation operations, known to be achievable in $O(\mathsf{cd})$ rounds, and requiring $\Omega(\mathsf{c}+\mathsf{d})$. The naive runtime is already optimal in some restricted cases (when $\mathsf{c} \in O(1)$ or $\mathsf{d} \in O(1)$), but not in general. While $O(\mathsf{c}+\mathsf{d})$ schedules are known to exist for standard packet routing (with fixed paths), our problem is a proper generalization of the usual routing scenario. We also need schedules that are distributedly computable. Problem 3 asks whether our subroutines can be performed faster, possibly also pipelined, certainly an exciting open question. It is essentially an independent scheduling question, despite its implications for the main results of our paper.

▶ **Problem 4.** *Can we* $\Delta^{O(t)}$*-color* G^t *in* $O(\Delta^{\lfloor (t-1)/2 \rfloor - \Omega(1)} \operatorname{poly} \log n)$ *rounds of* CONGEST*?*

We showed that the complexity of coloring needs to grow linearly with the congestion, but this was only shown existentially for a specific class of instances. Can this dependence on congestion be avoided? In particular, the complexity of distance-3 coloring is a major open question, where congestion is necessarily linear in Δ.

References

1 Noga Alon and Sepehr Assadi. Palette sparsification beyond $(\Delta + 1)$ vertex coloring. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 176 of *LIPIcs*, pages 6:1–6:22. LZI, 2020. doi:10.4230/LIPICS.APPROX/RANDOM.2020.6.

2 Ioannis Anagnostides, Christoph Lenzen, Bernhard Haeupler, Goran Zuzic, and Themis Gouleakis. Almost universally optimal distributed Laplacian solvers via low-congestion shortcuts. *Distributed Computing*, 36(4):475–499, 2023. doi:10.1007/S00446-023-00454-0.

3 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta+1)$ vertex coloring. In *the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 767–786, 2019. Full version at arXiv:1807.08886. doi:10.1137/1.9781611975482.48.

4 Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed edge coloring in time polylogarithmic in Δ. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 15–25. ACM, 2022. doi:10.1145/3519270.3538440.

5 Reuven Bar-Yehuda, Keren Censor-Hillel, Yannic Maus, Shreyas Pai, and Sriram V Pemmaraju. Distributed approximation on power graphs. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 501–510, 2020. doi:10.1145/3382734.3405750.

6 Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Morgan & Claypool Publishers, 2013. doi:10.2200/S00520ED1V01Y201307DCT011.

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

8 Leonid Barenboim and Uri Goldenberg. Speedup of distributed algorithms for power graphs in the CONGEST model. Technical report, arXiv, 2023. doi:10.48550/arXiv.2305.04358.

9 Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The complexity of $(\Delta + 1)$ coloring in congested clique, massively parallel computation, and centralized local computation. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 471–480, 2019. Full version at arXiv:1808.08419.

10 Yi-Jun Chang, Wenzheng Li, and Seth Pettie. Distributed $(\Delta + 1)$-coloring via ultrafast graph shattering. *SIAM Journal of Computing*, 49(3):497–539, 2020. doi:10.1137/19M1249527.

11 Sam Coy, Artur Czumaj, Peter Davies, and Gopinath Mishra. Optimal (degree+1)-coloring in congested clique. In *the Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 261 of *LIPIcs*, pages 46:1–46:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ICALP.2023.46.

12 Artur Czumaj, Peter Davies, and Merav Parter. Simple, deterministic, constant-round coloring in congested clique and MPC. *SIAM J. Comput.*, 50(5):1603–1626, 2021. doi:10.1137/20M1366502.

13 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. In *the Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 363–372, 2011. doi:10.1145/1993636.1993686.

14 Michael Elkin, Seth Pettie, and Hsin-Hao Su. $(2\Delta - 1)$-edge-coloring is much easier than maximal matching in the distributed setting. In *the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 355–370, 2015. doi:10.1137/1.9781611973730.26.

15 Salwa Faour, Mohsen Ghaffari, Christoph Grunau, Fabian Kuhn, and Václav Rozhon. Local distributed rounding: Generalized to MIS, matching, set cover, and beyond. In *the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4409–4447. SIAM, 2023. doi:10.1137/1.9781611977554.CH168.

16 Manuela Fischer. Improved deterministic distributed matching via rounding. In *the Proceedings of the International Symposium on Distributed Computing (DISC)*, volume 91 of *LIPIcs*, pages 17:1–17:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.DISC.2017.17.

17 Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. Coloring fast with broadcasts. In *the Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 455–465. ACM, 2023. doi:10.1145/3558481.3591095.

18 Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. A distributed palette sparsification theorem. In *the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2024.

19 Maxime Flin, Magnús M. Halldórsson, and Alexandre Nolin. Fast coloring despite congested relays. In *the Proceedings of the International Symposium on Distributed Computing (DISC)*, 2023. doi:10.4230/LIPICS.DISC.2023.19.

20 Maxime Flin, Magnús M. Halldórsson, and Alexandre Nolin. Decentralized distributed graph coloring: Cluster graphs. Technical report, arXiv, May 2024. In submission. doi:10.48550/arXiv.2405.07725.

21 Maxime Flin, Magnús M. Halldórsson, and Alexandre Nolin. Decentralized distributed graph coloring II: degree+1-coloring virtual graphs. Technical report, arXiv, 2024. Full version of this paper. doi:10.48550/arXiv.2408.11041.

22 Sebastian Forster, Gramoz Goranci, Yang P. Liu, Richard Peng, Xiaorui Sun, and Mingquan Ye. Minor sparsifiers and the distributed laplacian paradigm. In *the Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2021. doi:10.1109/FOCS52979.2021.00099.

23 Pierre Fraigniaud, Magnús M. Halldórsson, and Alexandre Nolin. Distributed testing of distance-k colorings. In *the Proceedings of the International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 2020. doi:10.1007/978-3-030-54921-3_16.

24 Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local Conflict Coloring. In *the Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 625–634, 2016. doi:10.1109/FOCS.2016.73.

25 Marc Fuchs and Fabian Kuhn. List defective colorings: Distributed algorithms and applications. In *the Proceedings of the International Symposium on Distributed Computing (DISC)*, LIPIcs, 2023. doi:10.4230/LIPICS.DISC.2023.22.

26 Mohsen Ghaffari. Near-optimal scheduling of distributed algorithms. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 3–12, 2015. doi:10.1145/2767386.2767417.

27 Mohsen Ghaffari and Christoph Grunau. Faster deterministic distributed MIS and approximate matching. In *the Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 1777–1790. ACM, 2023. doi:10.1145/3564246.3585243.

28 Mohsen Ghaffari, Christoph Grunau, Bernhard Haeupler, Saeed Ilchi, and Václav Rozhon. Improved distributed network decomposition, hitting sets, and spanners, via derandomization. In *the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2532–2566. SIAM, 2023. doi:10.1137/1.9781611977554.CH97.

29 Mohsen Ghaffari, Christoph Grunau, and Václav Rozhoň. Improved deterministic network decomposition. In *the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2021. arXiv:2007.08253.

30 Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks II: Low-congestion shortcuts, MST, and Min-Cut. In *the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 202–219. SIAM, 2016. doi:10.1137/1.9781611974331.CH16.

31 Mohsen Ghaffari and Bernhard Haeupler. Low-congestion shortcuts for graphs excluding dense minors. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 213–221. ACM, 2021. doi:10.1145/3465084.3467935.

32 Mohsen Ghaffari, Bernhard Haeupler, and Goran Zuzic. Hop-constrained oblivious routing. In *the Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 1208–1220. ACM, 2021. doi:10.1145/3406325.3451098.

33 Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. Near-optimal distributed maximum flow. *SIAM J. Comput.*, 47(6):2078–2117, 2018. doi:10.1137/17M113277X.

34 Mohsen Ghaffari and Fabian Kuhn. Distributed minimum cut approximation. In *International Symposium on Distributed Computing*, pages 1–15. Springer, 2013. doi:10.1007/978-3-642-41527-2_1.

35 Mohsen Ghaffari and Fabian Kuhn. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. In *the Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 1009–1020. IEEE, 2021. doi:10.1109/FOCS52979.2021.00101.

36 Mohsen Ghaffari and Goran Zuzic. Universally-optimal distributed exact min-cut. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 281–291. ACM, 2022. doi:10.1145/3519270.3538429.

37 Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. Low-congestion shortcuts without embedding. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 451–460. ACM, 2016. doi:10.1145/2933057.2933112.

38 Bernhard Haeupler, Shyamal Patel, Antti Roeyskoe, Cliff Stein, and Goran Zuzic. Polylog-competitive deterministic local routing and scheduling. *CoRR*, abs/2403.07410, 2024. doi:10.48550/arXiv.2403.07410.

39 Magnús M. Halldórsson, Fabian Kuhn, and Yannic Maus. Distance-2 coloring in the CONGEST model. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 233–242, 2020. doi:10.1145/3382734.3405706.

40 Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Alexandre Nolin. Coloring fast without learning your neighbors' colors. In *the Proceedings of the International Symposium on Distributed Computing (DISC)*, pages 39:1–39:17, 2020. doi:10.4230/LIPIcs.DISC.2020.39.

41 Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Tigran Tonoyan. Efficient randomized distributed coloring in CONGEST. In *the Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 1180–1193. ACM, 2021. Full version at arXiv:2105.04700.

42 Magnús M. Halldórsson, Fabian Kuhn, Alexandre Nolin, and Tigran Tonoyan. Near-optimal distributed degree+1 coloring. In *the Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 450–463. ACM, 2022. doi:10.1145/3519935.3520023.

43 Magnús M. Halldórsson and Alexandre Nolin. Superfast coloring in CONGEST via efficient color sampling. *Theor. Comput. Sci.*, 948:113711, 2023. doi:10.1016/J.TCS.2023.113711.

44 Magnús M. Halldórsson, Alexandre Nolin, and Tigran Tonoyan. Overcoming congestion in distributed coloring. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 26–36. ACM, 2022. doi:10.1145/3519270.3538438.

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

46 D. Hefetz, Y. Maus, F. Kuhn, and A. Steger. A polynomial lower bound for distributed graph coloring in a weak LOCAL model. In *the Proceedings of the International Symposium on Distributed Computing (DISC)*, pages 99–113, 2016.

47 Shimon Kogan and Merav Parter. Low-congestion shortcuts in constant diameter graphs. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 203–211. ACM, 2021. doi:10.1145/3465084.3467927.

48 Frank Thomson Leighton, Bruce M. Maggs, Abhiram G. Ranade, and Satish Rao. Randomized routing and sorting on fixed-connection networks. *J. Algorithms*, 17(1):157–205, 1994. doi:10.1006/JAGM.1994.1030.

49 Frank Thomson Leighton, Bruce M. Maggs, and Satish Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–186, 1994. doi:10.1007/BF01215349.

50 Frank Thomson Leighton, Bruce M. Maggs, and Andréa W. Richa. Fast algorithms for finding O(congestion + dilation) packet routing schedules. *Combinatorica*, 19(3):375–401, 1999. doi:10.1007/S004930050061.

51 Christoph Lenzen. Optimal deterministic routing and sorting on the congested clique. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 42–50. ACM, 2013. doi:10.1145/2484239.2501983.

52 Nati Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992. doi:10.1137/0221015.

53 Zvi Lotker, Boaz Patt-Shamir, and David Peleg. Distributed MST for constant diameter graphs. *Distributed Computing*, 18(6):453–460, 2006. doi:10.1007/S00446-005-0127-6.

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

55 Yannic Maus, Saku Peltonen, and Jara Uitto. Distributed symmetry breaking on power graphs via sparsification. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 157–167. ACM, 2023. doi:10.1145/3583668.3594579.

56 Yannic Maus and Tigran Tonoyan. Local conflict coloring revisited: Linial for lists. In *the Proceedings of the International Symposium on Distributed Computing (DISC)*, pages 16:1–16:18, 2020. doi:10.4230/LIPIcs.DISC.2020.16.

57 Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM J. on Comp.*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.

58 Bruce A. Reed. ω, Δ, and χ. *J. Graph Theory*, 27(4):177–212, 1998. doi:10.1002/(SICI)1097-0118(199804)27:4<177::AID-JGT1>3.0.CO;2-K.

59 Václav Rozhon, Christoph Grunau, Bernhard Haeupler, Goran Zuzic, and Jason Li. Undirected (1+ε)-shortest paths via minor-aggregates: near-optimal deterministic parallel and distributed algorithms. In *the Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 478–487, 2022. doi:10.1145/3519935.3520074.

60 Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *the Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 350–363, 2020.

61 Johannes Schneider and Roger Wattenhofer. A new technique for distributed symmetry breaking. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 257–266. ACM, 2010. doi:10.1145/1835698.1835760.

A Color Trials

We state here some classical symmetry breaking algorithms with additional assumptions for virtual graphs. The main difference with LOCAL or CONGEST version of these algorithm is that vertices sample in color space $\mathcal{C}(v)$ instead of their palette.

▶ **Lemma 18** (Random Color Trial). *Let $\gamma \in (0,1)$ be universal constants known to all nodes. Let φ be a coloring, $S \subseteq V \setminus \mathsf{dom}\,\varphi$ a set of uncolored nodes, and sets $\mathcal{C}(v) \subseteq [\deg(v)+1]$ for each $v \in S$ such that*

1. *v can sample a uniform color in $\mathcal{C}(v)$ in $O(1)$ rounds,*
2. *$|\mathcal{C}(v)| \geqslant \Theta(\gamma^{-1}\log n)$,*
3. *$|L_\varphi(v) \cap \mathcal{C}(v)| \geqslant \gamma|\mathcal{C}(v)|$, and*
4. *$|L_\varphi(v) \cap \mathcal{C}(v)| \geqslant \gamma|N_\varphi(v) \cap S|$.*

Let $\psi \succeq \varphi$ be the coloring produced by `TryColor`*. Then, w.h.p., each $w \in V_H$ has uncolored degree in S*

$$|N_\psi(w) \cap S| \leqslant \max\left\{(1-\gamma^4/64)|N_\varphi(w) \cap S|,\ \Theta(\gamma^{-4}\log n)\right\}.$$

The algorithm ends after $\widehat{O}(1)$ rounds and $\psi(v) \in \mathcal{C}(v)$ for all $v \notin \mathsf{dom}\,\varphi$.

The MultiColorTrial in Lemma 19 is adapted from [43] to sample colors from a restricted known color space.

▶ **Lemma 19** (MultiColorTrial, adapted from [43]). *Let φ be a (partial) coloring of H, $S \subseteq V_H \setminus \mathsf{dom}\,\varphi$, and $\mathcal{C}(v) \subseteq [\deg(v)+1]$ be a color space for each node. Suppose that there exists some constant $\gamma > 0$ known to all nodes such that*

1. *$\mathcal{C}(v)$ is known to all machines in $V(v)$; and*
2. *$|L_\varphi(v) \cap \mathcal{C}(v)| - |N_\varphi(v) \cap S| \geqslant \max\{2|N_\varphi(v) \cap S|, \Theta(\log^{1.1} n)\} + \gamma|\mathcal{C}(v)|$.*

Then, there exists an algorithm computing a coloring $\psi \succeq \varphi$ such that, w.h.p., all nodes of S are colored and $\psi(v) \in \mathcal{C}(v)$ for each $v \in S$. The algorithm runs in $\widehat{O}(\gamma^{-1}\log^* n)$ rounds.

B Pseudo-Code

Algorithm 2 `SlackGeneration`.

1 Each $v \in V_{\mathsf{sg}}$ joins V^{active} w.p. $p_{\mathsf{g}} = 1/20$.
2 Each $v \in V^{\mathsf{active}}$ samples $c(v) \in \{r+1, r+2, \ldots, \deg(v)+1\}$ uniformly at random.
3 Let $\varphi_{\mathrm{sg}}(v) = c(v)$ if $v \in V^{\mathsf{active}}$ and $c(v) \notin c(N_H^+(v))$. Otherwise, set $\varphi_{\mathrm{sg}}(v) = \perp$.

Algorithm 3 `ColoringNonCabals`.

Input: *A coloring φ such as described in Proposition 14*
Output: *A coloring $\psi \succeq \varphi$ such that $V_{\mathsf{dense}} \setminus V_{\mathsf{cabal}} = \mathsf{dom}\,\psi$*

1 `ColorfulMatching` when $a_K \geqslant \Omega(\log n)$ `// Let` φ_{cm} `be the coloring produced`
2 `ColoringOutliers` with $\mathcal{C}(v) = [r+1, \deg(v)+1]$
3 `SynchronizedColorTrial`
4 `TryColor` for $O(1)$ rounds with $\mathcal{C}(v) = L_\varphi(K_v) \cap [r+1, \deg(v)+1]$
5 `SliceColor` with $\mathcal{C}(v) = L_\varphi(K_v) \cap [r+1, \deg(v)+1]$
Let $\mathcal{L}_1, \ldots, \mathcal{L}_{O(\log\log n)}$ be the layers produced by `SliceColor`.
6 **for** $i = O(\log\log n)$ *to* 1 **do**
7 `MultiColorTrial` with $\mathcal{C}(v) = [r]$ in $\mathcal{L}_i$

Algorithm 4 Cabals.

Let $r' \stackrel{\text{def}}{=} 150\ell$, where $\ell = C_1 \log^{1.2} n$ is as described in Equation (8).

`ColorfulMatching`.

`ColoringOutliers` with $\mathcal{C}(v) = [\deg(v) + 1] \setminus [r']$.

`ComputePutAside` $P_K \subseteq I_K$.

`SynchronizedColorTrial` with $S_K = K \setminus (\mathsf{dom}\,\varphi \cup P_K)$

`SliceColor` with $\mathcal{C}(v) = [\deg(v) + 1] \setminus [r']$

Let $\mathcal{L}_1, \ldots, \mathcal{L}_{O(\log\log n)}$ be the layers produced by `SliceColor`

for $i = O(\log\log n)$ *to* 1 **do**

    `MultiColorTrial` with $\mathcal{C}(v) = [r']$ in $\mathcal{L}_i$

`ColorPutAsideSets`