Distributed Vertex Coloring in Bandwidth Constrained Models

Maxime Flin

Dissertation submitted to the Department of Computer Science at Reykjavík University in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

12 August 2025

Supervisor	Magnús M. Halldórsson Professor, Reykjavik University, Iceland
Thesis Committee	Fabian Kuhn Professor, Freiburg University, Germany
	Luca Aceto Professor, Reykjavik University, Iceland
	Merav Parter Professor, Weizmann Institute, Israel
Examiner	Yi-Jun Chang NUS Presidential Young Professor, National University of Singapore

ISBN 978-9935-539-84-7 Print version
ISBN 978-9935-539-85-4 Electronic version
Author's orcid 0009-0005-2693-0470

Abstract

This thesis studies the $(\Delta + 1)$-coloring problem in message-passing models under severe bandwidth constraints. In those models, a network of n computers is modeled as a graph G with n vertices that communicate in synchronous rounds over the edges of the graph. Our goal is to devise an algorithm with the smallest possible number of rounds at the end of which each vertex picks a color in $\{1, 2, \ldots, \Delta + 1\}$ — where Δ is the maximum degree of G — such that adjacent vertices have different colors. This task is central to distributed graph algorithms for it models symmetry breaking challenges ubiquitous in network computing and large-scale computing.

Despite its simplicity in the centralized setting, and considerable attention since the seminal work of Linial (*SIAM J. Comput.*, 1992) that introduced distributed graph algorithms, the complexity of $(\Delta + 1)$-coloring remains poorly understood. In recent years, a series of developments culminated in the first $\text{poly}(\log \log n)$-round algorithm for $(\Delta + 1)$-coloring, achieved by Chang, Li, and Pettie (*SIAM J. Comput.*, 2020) jointly with Rozhon and Ghaffari (*STOC*, 2020). Shortly thereafter, Halldórsson, Kuhn, Maus, and Tonoyan (*STOC*, 2021) further showed that this complexity is achievable with small $O(\log n)$-bit messages. Still, their algorithm requires strong forms of centralization. This prompted us to explore how severe bandwidth constraints affect the ability of distributed networks to get $(\Delta + 1)$-colored fast, making a threefold contribution.

First, we show that distributed graphs in which nodes merely broadcast one $O(\log n)$-bit message per round can be $(\Delta+1)$-colored in $\text{poly}(\log \log n)$ rounds, and even in $O(\log^* n)$ rounds when Δ is at least polylogarithmic. This broadcast constraint severely limits nodes, which might otherwise emit as many as $O(n \log n)$ bits per round. In fact, our algorithm is simple enough to be implemented in even weaker models, achieving the same $\text{poly}(\log \log n)$ round complexity if each node reads its received messages in a streaming fashion, using only $\text{poly}(\log n)$-bit of memory.

Second, we further reduce the number of bits emitted and received by each vertex. We design a distributed algorithm that $(\Delta + 1)$-colors a graph in $O(\log^2 \Delta) + \text{poly}(\log \log n)$ rounds using $O(\log n)$-bit messages, with vertices communicating with $O(\log^4 n)$ other vertices. At the heart of this

algorithm is an extension of the celebrated palette sparsification theorem by Assadi, Chen, and Khanna (*SODA*, 2019). They show that to $(\Delta+1)$-color a graph, it suffices if every vertex limits its color choice to $O(\log n)$ independently sampled colors in $\{1, 2, \ldots, \Delta+1\}$. However, computing the resulting coloring required a centralized approach. The main result of this part is a distributed palette sparsification theorem: a $O(\log^2 \Delta)$-round algorithm for $(\Delta+1)$-coloring the graph given random lists of $O(\log^2 n)$ random colors per node. We also prove that any algorithm of this kind has round complexity $\Omega\Big(\frac{\log \Delta}{\log \log n}\Big)$.

Finally, we color virtual graphs: graphs locally embedded within the communication network. It can be seen as an intermediate setting between that of the first and second parts, where vertices can interact with all their neighbors but only through processes akin to aggregation on trees. Besides generalizing classical distributed graph coloring, this captures other previously studied settings, including cluster graphs and power graphs. We show that the complexity of coloring a virtual graph depends on the dilation d and edge congestion c of its embedding. Surprisingly, we find that virtual graphs can be colored nearly as fast as ordinary graphs. Namely, we give a $O(\mathsf{cd} \cdot \log^* n)$-round algorithm for virtual graphs with Δ at least polylogarithmic and a $\mathsf{cd} \cdot \mathrm{poly}(\log \log n)$-round algorithm for graphs with $\Delta \leqslant \mathrm{poly}(\log n)$. This shows that the $(\Delta+1)$-coloring problem can be solved quickly even when the nodes themselves are decentralized.

Résumé

Cette thèse étudie le problème de la $(\Delta+1)$-coloration dans des modèles d'échanges de messages sous de fortes contraintes de bande passante. Dans ces modèles, un réseau de n ordinateurs est modélisé comme un graphe G avec n sommets qui communiquent en rounds synchrones sur les arêtes du graphe. Notre objectif est de concevoir un algorithme avec le plus petit nombre possible de rounds à la fin desquels chaque sommet choisit une couleur dans $\{1, 2, \ldots, \Delta + 1\}$ — où Δ est le degré maximum de G — de telle sorte que chaque paire de sommets adjacents recoive des couleurs différentes. Cette tâche est centrale pour les algorithmes de graphes distribués car elle modélise des défis de rupture de symétrie omniprésents dans le calcul distribué et le calcul à grande échelle.

Malgré sa simplicité dans un contexte centralisé, et malgré un effort de recherche considérable depuis le travail fondateur de Linial (*SIAM J. Comput.*, 1992) introduisant les algorithmes de graphes distribués, la complexité de la $(\Delta+1)$-coloration distribuée reste mal comprise. Ces dernières années, une série de développements a abouti au premier algorithme en $\text{poly}(\log\log n)$ rounds pour la $(\Delta+1)$-coloration, obtenu conjointement par Chang, Li, et Pettie (*SIAM J. Comput.*, 2020) avec Rozhon et Ghaffari (*STOC*, 2020). Peu après, Halldórsson, Kuhn, Maus, et Tonoyan (*STOC*, 2021) ont montré que cette complexité est atteignable avec de petits messages de $O(\log n)$ bits. Cependant, leur algorithme nécessite des fortes formes de centralisation. Cela nous a incités à explorer comment des contraintes de bande passante importantes affectent la capacité des réseaux distribués à se $(\Delta + 1)$-colorer rapidement. Notre contribution est en trois parties.

Premièrement, nous montrons que les graphes distribués dans lesquels les nœuds broadcast un message de $O(\log n)$ bits par round peuvent être $(\Delta + 1)$-colorés en $\text{poly}(\log\log n)$ rounds, et même en $O(\log^* n)$ rounds lorsque Δ est au moins polylogarithmique. Cette contrainte de diffusion limite sévèrement les nœuds, qui pourraient autrement émettre jusqu'à $O(n \log n)$ bits par round. En outre, notre algorithme est suffisamment simple pour être implémenté dans des modèles encore plus faibles, atteignant

la même complexité de $\text{poly}(\log\log n)$ rounds si chaque nœud lit ses messages reçus de manière continue, en utilisant seulement $\text{poly}(\log n)$ bits de mémoire.

Deuxièmement, nous réduisons davantage le nombre de bits émis et reçus par chaque sommet. Nous concevons un algorithme distribué qui $(\Delta+1)$-colore un graphe en $O(\log^2 \Delta) + \text{poly}(\log\log n)$ rounds en utilisant des messages de $O(\log n)$ bits, avec des sommets communiquant avec $O(\log^4 n)$ autres sommets. Au cœur de cet algorithme se trouve une extension du renommé théorème de sparsification de palette par Assadi, Chen, et Khanna (*SODA*, 2019). Ils montrent que pour $(\Delta+1)$-colorer un graphe, il suffit que chaque sommet limite son choix à $O(\log n)$ couleurs indépendamment échantillonnées dans $\{1, 2, \ldots, \Delta+1\}$. Cependant, le calcul de la coloration résultante nécessitait une approche centralisée. Le résultat principal de cette partie est un théorème de sparsification de palette distribué : un algorithme en $O(\log^2 \Delta)$ rounds pour $(\Delta+1)$-colorer le graphe étant donné des listes aléatoires de $O(\log^2 n)$ couleurs aléatoires par nœud. Nous prouvons également que tout algorithme de ce type a une complexité de $\Omega\left(\frac{\log \Delta}{\log\log n}\right)$ rounds.

Enfin, nous colorions des graphes virtuels : des graphes localement plongés dans le réseau de communication. Cela peut être vu comme un cadre intermédiaire entre celui des première et deuxième parties, où les sommets peuvent interagir avec tous leurs voisins mais seulement par des processus similaires à l'agrégation sur des arbres. Outre la généralisation de la coloration de graphes distribués classiques, cela capture d'autres cadres précédemment étudiés, y compris les graphes de clusters et les puissances de graphes. Nous montrons que la complexité de la coloration d'un graphe virtuel dépend de la dilatation d et de la congestion des arêtes c de son plongement. Nous trouvons que les graphes virtuels peuvent être colorés presque aussi rapidement que les graphes ordinaires. Plus précisément, nous donnons un algorithme en $O(\mathsf{cd} \cdot \log^* n)$ rounds pour les graphes virtuels avec Δ au moins polylogarithmique et un algorithme en $\mathsf{cd} \cdot \text{poly}(\log\log n)$ rounds pour les graphes avec $\Delta \leqslant \text{poly}(\log n)$. Cela montre que le problème de la $(\Delta+1)$-coloration peut être résolu rapidement même lorsque les nœuds eux-mêmes sont décentralisés.

Acknowledgements

First and foremost, I wish to thank my advisor Magnús M. Halldórsson, for his relentless enthusiasm and availability to work on research. I am very grateful for the countless hours we spent discussing computer science, for the most part. I am deeply grateful for the trust you placed in me, as well as for granting me the freedom to explore various research ideas or travel around, even before conferences deadlines.

Je voudrais également remercier Alexandre Nolin pour avoir été un second mentor pendant ces trois dernières années. Notamment pour m'avoir aidé à naviguer les difficultés techniques du coloriage distribué quand je suis arrivé en Islande, en sus d'innombrables autres sujets depuis — des inégalités de concentration au Weinschorle.

I am also very thankful to Luca Aceto, Yi-Jun Chang, Fabian Kuhn, Merav Parter for agreeing to be members of my thesis committee. Luca Aceto in particular for his attentive mentoring of all the PhD students at Reykjavik University.

I would also like to thank my all co-authors (in alphabetical order): Keren Censor-Hillel, Tomer Even, Mohsen Ghaffari, Magnus M. Halldórsson, Fabian Kuhn, and Parth Mittal. Our collaboration has been immensely instructive — far more than I could have anticipated. A lot of the enjoyment I got from working on this thesis came from all the amazing people in the distributed graph algorithm community, who made every conference or workshop an event to look forward to. I wish to also thank Sepehr Assadi for being one of the author of the palette sparsification theorem, on top of which builds one of my favourite parts of this thesis, and for inviting me at Waterloo University in January 2024. As far as my academic journey goes, I also wish to thank some of my earlier mentors — Danupon Nanongkai, Yann Régis-Gianas, Daniel Juteau and Muriel Livernet — for introducing me, in one way or another, to the wonders of research (although they might not remember how).

Thank you Yayoi and Magnús for inviting me to dinner several times and for being such a welcoming hosts.

Last but not least, I would like to express my heartfelt gratitude to my friends and familly: without you, those three years in Iceland would not have been bearable. Thank you Embla, Ioana, and Shalini — the friends I made on this lonely rock — for your support in difficult times, as well as for the many challenging and/or funny conversations we had the chance to share. Merci Alexandre, Félix, Louis, Marie, Pierre et Tristan d'avoir fait l'effort de rester en contact pendant ces trois années et d'avoir continué à me supporter, à me rassurer ou à me distraire à travers mes nombreuses crises existentielles. Je conclurai en m'excusant auprès de ma famille de m'en être allé aussi loin — qui plus est pour accomplir une besogne aussi ésotérique. À Basile tout particulièrement: désolé de t'avoir laissé tout seul, coincé avec les vieux.

Contents

CHAPTER 1

Introduction

Today, large distributed networks like the internet are ubiquitous. Beyond computer networks, the scale of social networks (e.g., social media, research communities, etc.), industrial networks (e.g., power grids, transportation networks, etc.), and other complex networks (e.g., brain neural networks) has grown at an unprecedented rate; see, e.g., [Bar16] and references therein for a comprehensive survey on networks. At the same time, the amount of data generated by these networks is growing exponentially. Thus, efficient algorithms for large-scale computer networks or massive datasets are now a crucial component of almost every computer system. For instance, information is often dispersed in space; either as part of a widely decentralized network (e.g., in the case of networks of sensors monitoring meteorological or seismic activity) or because the amounts of data are too large to fit into a single machine (e.g., the graph of the world wide web). As a result, latency — that is, the time necessary for data to move in the network — is a core challenge when performing computations on these networks. In practice, many factors contribute to the latency observed by the users behind their computer, but we can sort them into three constraints: distances between machines, limitations stemming from the communication channels and limitations of the machines themselves.

To stress how fundamental *locality* — that is, the physical proximity — is in distributed networks, recall that information cannot travel faster than the speed of light. While light appears to travel instantaneously from a human perspective, computation on modern processing units is essentially free compared to the cost of communicating even a single bit of information. Indeed, as light travels one meter, a modern computer can perform about a dozen operations. So even at the scale of a country, let alone of the earth, transmitting information is costlier than computation by a couple of orders

of magnitude. As a result, models for distributed computing tend to focus on the cost of communication rather than that of local computations.

On top of locality come constraints from the network infrastructure, since the rate at which information can be transmitted is limited by the bandwidth of the cables used to connect computers. Bandwidth constraints are most pregnant when large amounts of data need to be transmitted from a few sources, as is the case when broadcasting a movie or sporting event. Generally speaking, *congestion* issues arise when the amounts of data to transmit exceed the bandwidth. In wireless networks, congestion issues are heightened, for machines can only broadcast small messages (as opposed to having the ability to send different messages to each indivial machine) and interferences disrupt the transmissions if more than one machine emits at the same time. So, in many cases, transmitting large amounts of data is prohibitively expensive and we need to be selective about what each machine needs to learn.

Even if all the necessary data has been gathered in one location, it might not fit into the memory of a single processing unit. To convey a sense of the size of such datasets, consider that, already in 2008, Google was processing more than twenty petabytes of data per day [DG08] and that the Facebook graph has trillions of edges [CEKLM15]. To process this data, companies build massive data centers where machines are near each other and supplied with high bandwidth cable connections to communicate with each other. Hundreds of machines are necessary to store the whole dataset and solving computational tasks therefore requires them to communicate through message passing without ever accessing the entire input. It is crucial to emphasize that, notwithstanding the small locality and congestion in data centers, communication between machines remains notably more time consuming than local computations.

To summarize, in distributed systems, no individual actor has a complete knowledge of the network. This limitation leads to the following fundamental question

> *How can one solve tasks efficiently with only a partial knowledge of the input?*

This is a very broad question that can be formalized from many different angles, to the degree that, over the last two decades, increasing attention has been given to algorithms for new models of computation designed to capture the aforementioned limits more accurately. In this context, communication is often used to model how much knowledge is necessary to perform certain tasks. For instance, if Alice whishes to know if Bob knows about a certain book, she needs only to ask one question with the name of the book. However, if Alice and Bob want to check if there is a book that they both like, either one of them will essentially have to ask about of all the books they like (in the worst case). Thus, solving the former task requires much less knowledge about Alice's and Bob's tastes than solving the latter.

This thesis is also a story about communication, but one where communication occurs over large networks of computers. Broadly speaking, we focus on the distributed graph algorithm setting, which models networks as graphs where vertices represent machines that communicate with their neighbors in order to solve a computational task. This formalism captures the challenges mentioned earlier: distances in the graph model how far apart machines are in the real world, links may have limited bandwidth, and the machines themselves might have limited memory compared to the size of the network.

1.1. Distributed Graph Algorithms

Distributed graph algorithms is a broad field, and we make no attempt to provide an exhaustive overview of the plethora of models studied in this context. Instead, we view these models as a spectrum to measure the effects of locality or bandwidth constraints for specific problems. In particular, this thesis studies the complexity of a fundamental coloring problem accross this spectrum.

Let us present the distributed models of most relevance to us.

Locality. In the LOCAL model, introduced by Linial [Lin92], the input graph is seen as an idealized network of computers where edges of the graph represent links over which machines can communicate. Communication occurs in synchronous rounds, without any loss or corruption of messages, and

the size of messages is not constrained. The goal is to solve a graph problem (e.g., coloring, maximal matching, etc.) in as few rounds as possible. The smallest number of rounds needed by a LOCAL algorithm to solve a problem is called the *locality* of the problem. Since the size of messages is not constrained in LOCAL, in T rounds, a machine can always learn the entire graph within distance T. Conversely, in a T round algorithm, machines at distance $2T+1$ from each other do not interact at all. So locality measures how far entities of a network need to synchronize to solve a problem. If a problem requires as many rounds as the diameter of the graph, then it is inherently global as it requires at least one vertex to learn essentially the whole graph. On the other hand, some problems (e.g., $(\Delta+1)$-coloring, maximal independent set, etc.) can be solved in considerably fewer rounds than the diameter of the graph.

Symmetry Breaking. A fundamental and remarkably challenging issue for distributed networks of computers is to *break symmetry*. Imagine, as a case in point, two cars on either side of a narrow bridge and wishing to cross it: they can either proceed to cross the bridge or wait for the other car to do so. If the two cars keep behaving the same, neither will ever cross the bridge. To resolve this impasse, they need to behave asymmetrically, for instance, by letting the one with the smaller plate number go first or by tossing a coin. While this example may appear silly, locality by itself causes intricate issues of that sort because one has to break symmetry without any knowledge of how it may affect distant places.

Coming back to the LOCAL model, the challenge is to ensure that vertices with the same local view of the network eventually behave differently. Since this is not possible without some additional assumptions, even in a graph consisting of a single edge, we provide vertices with unique identifiers (akin to plate numbers) or with a source of randomness (local random coins).

Congestion with Locality. Formally introduced by Peleg [Pel00], the CONGEST model is identical to the LOCAL model, except that the size of the messages sent between vertices is bounded by $O(\log n)$ bits, where n is the number of machines in the network. With this restriction, it is no

longer true that in T rounds vertices learn all the information in their T-hop neighborhood. At best, they receive $O(T \log n)$ bits of information from each neighbor and, in general, efficiently routing messages between distant nodes becomes a challenge. Informally speaking, congestion forces vertices to be selective about the information they need to learn from their T-hop neighborhood. While congestion has long been known to create bottlenecks for global problems [SHKKNPPW12], many LOCAL algorithms for local problems do not take advantage of the unbounded message size.

In wireless communications, machines can only broadcast messages and simultaneous emission of messages creates interference. Various models have been introduced to model communication over such networks: beeping [CK10], energy-focused models (see e.g., [CKPWZ19] and references therein) or physical models such as SINR. Ignoring interference caused by collisions, the CONGEST model where vertices are limited to broadcast only a single $O(\log n)$-bit message each round significantly weakens the model, for it reduces the throughput of a vertex by a factor proportional to its degree (its number of incident edges). We will henceforth refer to this variant as to the BCONGEST model.

Congestion without Locality. In some cases, the communication network is seen as a black box that offers all-to-all communication (e.g., in data centers or on the internet through TCP/IP). The focus is then on the limitations incurred by the bandwidth of the network. To model such cases, Lotker, Patt-Shamir, Pavlov, and Peleg [LPPP05] introduced the CONGESTED-CLIQUE model. In this model, the size of messages is limited to $O(\log n)$ bits but all-to-all communication is allowed, removing entirely the locality constraint to focus on the congestion. To more accurately model peer-to-peer networks, for which it is infeasible to assume that a node can send/receive more than $O(\log n)$ messages per round, Augustine, Ghaffari, Gmyr, Hinnenthal, Scheideler, Kuhn, and Li [AGGHSKL19] introduced the *node-capacitated clique* (or NCC) model.

In this thesis. We focus on constrained variants of the CONGEST model. This allows us to explore models like BCONGEST and NCC as well as settings with constrained memory or partial knowledge of the input graph.

1.2. Distributed $(\Delta+1)$-Coloring

A q-vertex-coloring (or simply q-coloring) of a graph G is a mapping from the vertices of G to the set $\{1,\dots,q\}$ such that adjacent vertices receive different colors. Coloring problems are fundamental to many areas of computer science and this thesis focuses on the $(\Delta+1)$-coloring problem — that is, given a graph G with maximum degree Δ, we want to find a $(\Delta+1)$-coloring of G. In the classical setting, this problem can be solved by a simple linear-time greedy algorithm that colors vertices in any order. In distributed settings, however, $(\Delta+1)$-coloring is emblematic of the challenge of breaking symmetry. As such, the $(\Delta+1)$-coloring problem (as well as other coloring problems) has been intensively studied in the field of distributed graph algorithms since the birth of the research area. For instance, in their seminal paper, Cole and Vishkin showed that 3-coloring n-cycles can be done in $O(\log^* n)$ rounds [CV86]. Quickly thereafter, Linial showed that any deterministic distributed algorithm to 3-color n-cycles requires $\Omega(\log^* n)$ rounds [Lin92] and this lower bound was later extended to randomized algorithms by Naor and Stockmeyer [NS95]. To this day, this remains the best known lower bound for the $(\Delta+1)$-coloring problem in LOCAL.

For a long time, the best upper bound for $(\Delta+1)$-coloring general graphs was the $O(\log n)$-round algorithm by Johansson [Joh99]. In this algorithm, at each round, uncolored vertices compute their *palette*: the set of colors not used by colored neighbors. Then, they sample a random color from that palette and, if no uncolored neighbor sampled the same color, the vertex colors itself — we say that the vertex *adopts* its color. Otherwise, the vertex remains uncolored. One can demonstrate that a vertex remains uncolored with constant probability, thereby establishing that the algorithm ends after $O(\log n)$ rounds with high probability. Despite its simplicity and efficiency, Johansson's algorithm is very far from the $\Omega(\log^* n)$ lower bound. It remains nonetheless relevant today for random color trials are the basis of most $(\Delta+1)$-coloring algorithms.

The first significant step towards a faster algorithm was made by Barenboim, Elkin, Pettie, and Schneider [BEPS16] when, inpired by the work of Beck [Bec91] on the Lovász Local Lemma (see, e.g., Chapter 6.7 in [MU05]),

they introduced a *shattering* technique to improve the round complexity to

$$O(\log \Delta) + \text{Det}(\text{poly}(\log n)) \ . \tag{1.1}$$

Here $\text{Det}(N)$ denotes the runtime of the best deterministic LOCAL algorithm for $(\deg+1)$-list-coloring N-vertex graphs. In that coloring problem, vertices are given a list of one more color than their degree and must pick a color from their list. Simplifying slightly, the authors showed that after $O(\log \Delta)$ random color trials, the uncolored vertices are gathered in $\text{poly}(\log n)$-sized connected components. Eq (1.1) is then obtained by running the best deterministic algorithm for extending the coloring to those small components.

The second breakthrough came from Harris, Schneider, and Su [HSS18] who, inspired by a result by Reed [Ree98], realized that they could take advantage of a decomposition of the graph into sparse and dense vertices. It was known since [MR14; EPS15] that sparse vertices were colorable with random color trials. This is because, after a single random color trial, many of their neighbors get colored the same, which provides them with a lot of slack: as we extend the coloring, they will always have $\Omega(\Delta)$ colors to choose from. In fact, a technique of Schneider and Wattenhofer [SW10] colors the sparse vertices (or shatters the graph) in $O(\log^* n)$ rounds.

So the core challenge of [HSS18] was to color the dense vertices faster than Eq (1.1). They showed that dense vertices were clustered into dense components called *almost-cliques*. Roughly speaking, those are $(\Delta+1)$-cliques[1] in which every vertex might be missing a few edges to the clique and have a few extra edges to the rest of the graph. The key insight of [HSS18] was that vertices of the same almost-clique are so densely connected to each other that they can *synchronize their color trials*. This idea allowed them to color the graph in $O(\sqrt{\log n}) + \text{Det}(\text{poly}(\log n))$ time. Shortly after, Chang, Li, and Pettie [CLP20] improved this to

$$O(\log^* n) + \text{Det}(\text{poly}(\log n)) \ , \tag{1.2}$$

which stands as the state-of-the-art for $(\Delta+1)$-coloring in the LOCAL model. Interestingly, when Δ is larger than some $\text{poly}(\log n)$ threshold, the success probability of [CLP20] becomes high enough that shattering

[1] a set of $\Delta+1$ vertices where each vertex is adjacent to all others

is not necessary anymore. For such graphs, Eq (1.2) therefore reduces to $O(\log^* n)$ rounds.

Concurrently, a breakthrough of Rozhoň and Ghaffari [RG20] on deterministic algorithms for network decompositions in the LOCAL model provided the first poly$(\log n)$-round deterministic algorithm for $(\deg+1)$-list-coloring (and maximal independent set), thereby solving a long-standing open problem. Several works have since improved the round complexity of this algorithm (e.g., [GGR21; GK21; GGHIR23; GG24] and references therein). Combined with Eq (1.2), at the time this thesis is written, the randomized complexity of $(\Delta+1)$-coloring in LOCAL is known to be somewhere between $\Omega(\log^* n)$ and $\widetilde{O}(\log^{5/3}\log n)$ rounds[2].

Model	Complexity	Reference
LOCAL det	$O(\sqrt{\Delta\log\Delta}+\log^* n)$	[FHK16]
	$O(\log^5 n)$	[RG20; GGR21]
	$\widetilde{O}(\log^{5/3} n)$	[GG24]
CONGEST det	$O(\log^2\Delta\cdot\log n)$	[GK21]
LOCAL rand	$O(\log n)$	[Joh99]
	$O(\log\Delta+\mathrm{Det}(\log n))$	[BEPS16]
	$O(\sqrt{\log n}+\mathrm{Det}(\log n))$	[HSS18]
	$O(\log^* n+\mathrm{Det}(\log n))$	[CLP20]
CONGEST rand	$O(\log^* n+\mathrm{Det}(\log n))$	[HKMT21; HNT22]
BCONGEST rand	$O(\log\Delta+\mathrm{Det}(\log n))$	[BEPS16]
	$O(\log\log n+\mathrm{Det}(\log n))$	**This Thesis**
CONGESTED-CLIQUE	$O(1)$ det	[CFGUZ19; CDP21b]
NCC	$O(\log^5 n)$	**This Thesis**

Table 1. State of the art for $(\Delta+1)$-coloring in classical distributed models. $\mathrm{Det}(N)$ represents the round complexity of the fastest deterministic algorithm for $(\deg+1)$-list-coloring on N-vertices graphs in the corresponding model.

[2]We use $\widetilde{O}(f)$ to hide polylogarithmic terms in f

This Thesis. While the LOCAL model is arguably the most important model of distributed graph algorithms — for it is the only one with unconditional super-constant lower bounds for coloring problems (e.g., [BFHKLRSU16; BBKO22] and references therein) which pertain to other models [GKU19; CDP24] — it is not a very realistic model: it assumes that nodes have unbounded computational power and that messages can be of unbounded size. In fact, the $\widetilde{O}(\log^{5/3} \log n)$-round algorithm of [CLP20; GG24] takes advantage of the full power of the LOCAL model ([CLP20] uses large messages and [GG24] has both large messages and exponential local computations).

Meanwhile, $(\Delta + 1)$-coloring has been studied in a wide variety of more realistic settings. Surprisingly perhaps, the authors of [CFGUZ19; CDP21b] demonstrated that congestion does not represent a significant barrier for $(\Delta + 1)$-coloring by solving it in $O(1)$ rounds in the congested clique model. Shortly after, the authors of [HKMT21; HNT22] showed that $(\Delta + 1)$-coloring could be solved in CONGEST essentially as fast as in LOCAL. And, perhaps the most surprising of all, the authors of [ACK19] showed that to $(\Delta + 1)$-color a graph, it was not necessary to store all its edges in memory. This streak of results inspired us to search for the weakest possible distributed model in which $(\Delta + 1)$-coloring could be solved efficiently. The caveat to push all these results further is that *they heavily rely on a form of centralization*: in [CFGUZ19; CDP21b; ACK19], one machine eventually learns $\widetilde{O}(n)$ edges; while in [HKMT21; HNT22], each dense cluster features a single machine that determines the color for nearly all other vertices in the cluster. So these results all break down when the bandwidth or the locality is more constrained.

This thesis aims at better understanding the complexity of $(\Delta + 1)$-coloring, by asking the following question:

> *What features of the computing & communication model make fast $(\Delta + 1)$-coloring feasible?*

The meaning of *fast* is ill-defined but, based on the recent advances on $(\Delta+1)$-coloring (as well as other problems) in LOCAL, we shall consider it within the context of the following complexity ranges:

- (Linear) $\Theta(n)$ or $\Theta(\Delta)$ rounds,

- (Logarithmic) $\text{poly}(\log n)$ rounds,
- (Doubly-Logarithmic) $\text{poly}(\log \log n)$ rounds, and
- (Ultrafast) $\Theta(\log^* n)$ rounds.

A model for which the round complexity is linear indicates that breaking symmetry has become intrinsically hard. Asking for a logarithmic runtime is asking if we can solve $(\Delta + 1)$-coloring as fast as the simple random color trial algorithm does in LOCAL. If we can reach a doubly-logarithmic runtime, this means that we are essentially as fast as the state-of-the-art LOCAL algorithm. Finally, a log-star runtime is the best one can hope for on general graphs and almost optimal given the exceedingly slow growth rate of the function. Moreover, these cutoffs corresponds to the possible complexity of LOCAL algorithms on trees [CP19], which makes them natural reference points even in cases where the local hierarchy does not apply.

We remark that obtaining the log-star runtime for $(\Delta + 1)$-coloring when Δ is large requires significantly more work than obtaining a doubly-logarithmic runtime. Whenever possible, we make this effort, for it ensures that we almost reached the optimal round complexity in high-degree graphs. Low-degree graphs behave somewhat differently for $(\Delta+1)$-coloring and, in almost all cases, it is easy to adapt the CONGEST algorithm to our needs. *Most of this thesis therefore focuses on high-degree graphs*, i.e., such that Δ is asymptotically larger than some $\text{poly}(\log n)$.

1.3. Main Contributions of this Thesis

In this thesis, we make progress on three metrics for distributed $(\Delta+1)$-coloring: the bandwidth usage, the local memory usage, and the knowledge of the graph.

Part I: Reducing the Bandwidth. While the work of [HSS18; CLP20] reduced the round complexity of $(\Delta+1)$-coloring to a sublogarithmic number, their algorithm uses large messages. Shortly after, [HKMT21; HNT22] provided equally fast algorithms with small messages (i.e., $O(\log n)$ bits), however they still require that each vertex communicates individually to each neighbor. Thus each vertex may send up to $\Theta(n \log n)$ bits in each

round. Our first research question asks how fast can one can compute a coloring if we constrain the set of outgoing messages. Specifically,

> *Can we compute a* $(\Delta+1)$*-coloring as fast as in the* CONGEST *model if, in each round, each node must transmit the same* $O(\log n)$*-bit message to all its neighbors?*

We answer this question in the affirmative by providing a *broadcast congest* $(\Delta+1)$-coloring algorithm that runs as fast as the state-of-the-art algorithm in the CONGEST model.

> RESULT 1. *There is a distributed* $(\Delta+1)$*-coloring algorithm that runs in* $O(\log^3 \log n)$ *rounds and where each vertex broadcasts one* $O(\log n)$*-bit message in each round. If* $\Delta \geqslant \text{poly}(\log n)$*, the algorithm runs in* $O(\log^* n)$ *rounds.*

This result is particularly interesting as the synchronized color trial, essential to all sublogarithmic algorithms, was heavily reliant on centralization in [HKMT21; HNT22] and could not be implemented efficiently in the broadcast congest model even with $o(\Delta)$-sized messages. As such, the only prior algorithm for $(\Delta+1)$-coloring general graph in broadcast congest was the $O(\log n)$-round algorithm of [Joh99].

Since the overarching goal in this work is not tied to any particular model, we aim to develop a distributed algorithm that assumes the least power provided by theoretical model. Towards this end, our broadcast congest algorithm introduces technical ideas (e.g., a distributed algorithm for colorful matching, setting aside reserved colors, using random groups) that are widely used throughout the thesis.

We point out that our algorithm is basic enough to be implemented even with limited memory per node, with only small additional changes. Notice that a node may receive many messages from its neighbors, up to $\Omega(n \log n)$ bits overall in one round. In general, receiving so many bits would necessitate significant memory for the node, and it also can complicate the task of simulating this algorithm in virtual graphs (see Part III). We show that our algorithm can be adapted to work with the same round complexity

when each node processes its incoming messages in a streaming fashion, using only $\text{poly}(\log n)$ memory (see Chapter 5).

Part II: Distributed Palette Sparsification. Distributed graph algorithms generally assume that the vertices are able, at each round, to exchange messages with all of their neighbors in the network. Even in the broadcast congest algorithm introduced in Part I, despite the fact that, in each round, vertices send a unique $O(\log n)$-bit message to each of their neighbors and that incoming messages can be processed in a stream, it is essential that every vertex receives all messages emitted by its neighbors in the same round.

While this assumption may be reasonable for distributed computations in data centers — or more generally, between highly connected servers — it is unrealistic in highly decentralized networks. Typically, in peer-to-peer networks or networks of sensors, vertices are small devices with limited computational power and memory. In such scenarios, we would prefer distributed algorithms in which vertices send and receive at most $\text{poly}(\log n)$ bits per round, even when Δ is very large. Our second research question therefore asks whether such algorithms exists:

> *How fast can we color a graph in a distributed model where vertices can only send and receive* $\text{poly}(\log n)$ *bits per round?*

To go even further, one can argue that having vertices store all their incident edges is not feasible in such scenarios. So a variation of this research question asks:

> *How fast can we color a graph in a distributed model where vertices can only store* $\text{poly}(\log n)$ *bits?*

The question of whether a graph could be $(\Delta+1)$-colored by using only $\text{poly}(\log n)$ bits of memory per vertex was already studied in the context of *semi-streaming algorithms* (see, e.g., [FKMSZ05; AGM12]). In this model, edges of the graph are processed in a stream and the goal is to compute a coloring of the graph using only $n \cdot \text{poly}(\log n)$ bits of memory. In a way, we ask for a distributed version of this semi-streaming algorithm: while the semi-streaming model requires that the average memory per vertex

is $\text{poly}(\log n)$, we require that every vertex uses only $\text{poly}(\log n)$ bits of memory.

That such a thing is possible, even in the semi-streaming setting, is surprising: how can we give two vertices a color if we do not know if they are adjacent? The answer lies in a combinatorial result of Assadi, Chen and Khanna [ACK19] known as the *Palette Sparsification Theorem*. This theorem states that, if we sample lists of $O(\log n)$ random colors in $\{1, 2, \ldots, \Delta + 1\}$ for each vertex, then, with high probability, there exists a coloring of the graph in which every vertex has a color from its list. From this theorem, they derive a semi-streaming algorithm by observing that, to color the graph, one needs only to store edges for which the two endpoints have intersecting lists. Since the lists are made of $O(\log n)$ random colors, this amounts to $O(n \log^2 n)$ edges with high probability. The graph retaining only those edges is called the *sparsified graph*.

The result of [ACK19] gives rise to the hope that there might be an ultimately scalable distributed solution for $(\Delta + 1)$-coloring, where each node needs to interact and coordinate with only $\text{poly}(\log n)$ of its neighbors. However, all known applications of the palette sparsification theorem require gathering the sparsified subgraph in one location, and solving the resulting list-coloring problem in a centralized fashion. This is prohibitively expensive in distributed models with limited communication.

The second major contribution of this thesis is to give a distributed version of the palette sparsification theorem. We show that there is a fast distributed algorithm for coloring the sparsified subgraph, which relies solely on communication within the sparsified graph.

RESULT 2. *If every vertex has a list of $O(\log^2 n)$ random colors in $\{1, 2, \ldots, \Delta + 1\}$, then there is a distributed algorithm that colors the sparsified graph in $O(\log^2 \Delta + \log^3 \log n)$ rounds and using $O(\log n)$ bit messages. In particular, each node needs to communicate with only $O(\log^4 n)$ different neighbors.*

That result answers our research question positively since it allows us to design the first polylogarithmic-round algorithms for $(\Delta+1)$-coloring in NCC, where vertices can communicate with only $O(\log n)$ neighbors per round, and in a distributed variant of the semi-streaming model, which we term Dist-Stream, where vertices have only $\text{poly}(\log n)$ memory.

Interestingly, the distributed palette sparsification theorem also allows us to design the first polylogarithmic-round algorithms for coloring virtual graphs. Informally speaking, this captures situations where the knowledge of the graph is itself distributed. We detail this more in the third part of this thesis.

Conversely, we show that algorithms communicating only on the sparsified graph cannot color the graph in truly sublogarithmic time.

> RESULT 3. *Any distributed* $(\Delta+1)$*-coloring algorithm based on palette sparsification with* $\text{poly}(\log n)$*-sized lists requires* $\Omega\left(\frac{\log \Delta}{\log \log n}\right)$ *rounds.*

This lower bound is particularly interesting in constrast to coloring algorithms in LOCAL, CONGEST and even BCONGEST, which can color a graph in $\text{poly}(\log \log n)$ rounds, and even in $O(\log^* n)$ rounds when Δ is sufficiently large.

Part III: Coloring Virtual Graphs. The common assumption in distributed graph algorithms is that the input graph is the communication network. However, in many cases, we run a distributed graph algorithm on a graph that is *not* the communication network.

For a real-life example, consider researchers scheduling conferences. Each community (e.g., distributed, dynamic, parametrized complexity, etc) has its own conference; however, some researchers belong to multiple communities. Scheduling these conferences so that researchers can attend all relevant conferences is equivalent to coloring the graph where communities are vertices, and edges connect communities that share at least one researcher. But communities do not communicate directly with one another. Instead, researchers communicate with their acquaintances within

their own communities. Therefore, the graph to be colored differs from the social network on which communication occurs.

In the literature on distributed graph algorithms, such scenarios occur, for instance, when the algorithm contracts edges, as seen in [GK13; GH16; GKKLP18; RG20; GGR21; FGLPSY21; GZ22; RGHZL22; GGHIR23]. It is also common to solve a problem on the square graph[3] of the input graph — for example, in [FG17; FGGKR23]. For coloring, this means that instead of merely requiring adjacent vertices to choose different colors, we ask that pairs of vertices at distance at most 2 from each other choose different colors. As such, the graph to be colored has edges between vertices that are not adjacent in the communication graph.

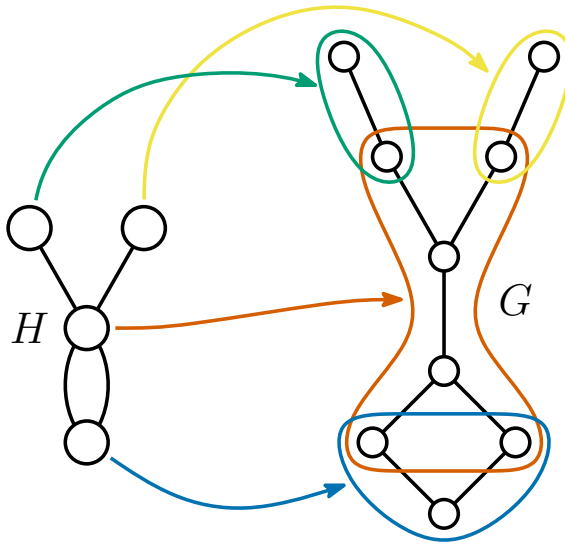

Figure 1. An example of an embedded virtual graph: on the left the graph to be colored H; on the right, the communication network G. The dilation and congestion of the embedding are three and one respectively.

[3] the square graph of a graph G, denoted G^2, is the graph on the same vertex set as G where $u \neq v$ are adjacent iff they are at distance at most two from each other in G. By extension, the k-th power of G, denoted G^k, is the graph where $u \neq v$ are adjacent iff they are at distance at most k from each other in G

In LOCAL, where the message size is unbounded, we can simulate every round of communication on the square graph with two rounds of communication on the input graph. In CONGEST, however, we encounter congestion issues, especially for graphs with a high degree. This is a recurrent issue in designing distributed algorithms, which leads us to our third research question of this thesis:

> *How bandwidth constraints affect distributed algorithms solving problems on graphs whose description is itself distributed on a communication network?*

First and foremost, we formalize this question with the notion of *virtual graphs*. Vertices of a virtual graph are sets of machines on the communication network such that adjacent vertices share a machine. For instance, the square of a graph can be modeled as a virtual graph where each vertex maps to its closed neighborhood.

Two key parameters of this formalism are the *dilation* and *congestion*. The dilation measures the diameter of each set of machines and the congestion measures the number of different sets that use the same link on the communication network.

Our first result is to show that those parameters are limiting factors for any algorithm solving $(\Delta+1)$-coloring on virtual graphs.

> RESULT 4. *Any algorithm for $(\Delta+1)$-coloring virtual graphs with dilation d, congestion c and b-bit messages requires $\Omega(\mathsf{c}/\mathsf{b}+\mathsf{d}\log^* n)$ rounds.*

Conversely, we provide fast algorithms for coloring virtual graphs. In Part II, we observe that the Distributed Palette Sparsification Theorem can be used to color virtual graphs in polylogarithmic time. However, as shown in Result 3, such algorithms cannot achieve truly sublogarithmic round complexity. We show that virtual graphs can be colored asymptotically as fast in LOCAL, up to a cd multiplicative factor.

RESULT 5. *There exists a randomized algorithm that $(\Delta+1)$-colors any virtual graph with dilation d and congestion c in $O(\mathsf{cd}\cdot\text{poly}(\log\log n))$ rounds with high probability. When $\Delta \geqslant \text{poly}(\log n)$, it ends in $O(\mathsf{cd}\cdot\log^* n)$ rounds with high probability.*

This results has a number of interesting corollaries as virtual graphs can be used to model many distributed computing problems. For instance, it implies that we can compute a distance-2 (Δ^2+1)-coloring in $\text{poly}(\log\log n)$ rounds, and even in $O(\log^* n)$ rounds when Δ is large enough. It also implies that cluster graphs, which are a special case of virtual graphs, can be colored in $\text{poly}(\log\log n)$ rounds.

Key Takeaways. In closing this introduction, we provide a high-level overview of the conclusion of this thesis. The technical core of this work primarily addresses the use of randomized color trials, tailored to the constraints of the underlying model. In Part II, we demonstrate that models where oblivious colors trials are efficient — as is the case, e.g., in NCC or cluster graphs — allow for coloring in logarithmic round. On the other hand, when the model allows for an efficient synchronized color trial — as it the case in Parts I and III — then it is possible to color in doubly-logarithmic many rounds. In fact, when the degree is at least polylogarithmic, and multicolor trials can be implemented efficiently, the round complexity further improves to log-star. We elaborate on this in Chapter 14.

1.4. Roadmap

This thesis is organized in three parts, following the three main contributions. Prior to that, we provide the necessary notation and background:

- Chapter 2 describes the notation we use throughout the thesis. It also gives formal definitions of LOCAL, CONGEST, NCC, and others. We routinely use concentration inequalities, not all of in their most common form, and refer readers to Section 2.3 for details.

- Chapter 3 summarizes the state of the art randomized LOCAL algorithm by [BEPS16; HSS18; CLP20; HKNT22] for $(\Delta + 1)$-coloring. It contains all proofs, some of which more general than strictly necessary as they are used in later chapters. It begins with a high-level overview of the algorithm and describes important concepts for this thesis such as slack, slack generation, multi-color trials, almost-clique decomposition, synchronized color trial and put-aside sets.

Part I: Reducing the Bandwidth.

- Chapter 4 provides a broadcast congest algorithm for $(\Delta + 1)$-coloring (Result 1). On the way, it introduces several useful techniques like a bandwidth efficient algorithm for computing the almost-clique decomposition, a version of the multicolor trial algorithm for broadcast congest, a simple distributed algorithm for computing a colorful matching and a new scheme for distributively sampling a random permutation of almost-cliques.
- Chapter 5 modifies the broadcast congest algorithm for the setting where incoming messages are processed in a streaming fashion.

Part II: Distributed Palette Sparsification.

- Chapter 6 introduces the Distributed Palette Sparsification Theorem (Result 2). It contains an overview of the argument and explains how it applies to various settings, from local streaming to peer-to-peer networks.
- Chapter 7 contains the full description and analysis of the algorithm behind the Distributed Palette Sparsification Theorem.
- Chapter 8 proves that no algorithm based on palette sparsification can truly be sub-logarithmic (Result 3).

Part III: Coloring Virtual Graphs.

- Chapter 9 introduces, motivates and formally defines our notion of virtual graphs. As such, it states with more details the results of Part III. It also gives an detailed overview of the technical ideas of Part III.

- Chapter 10 proves that congestion and dilation are bottlenecks for coloring algorithms on virtual graphs (Result 4).
- Chapter 11 introduces a general set of tools to work on virtual graphs. That includes rounting mechanisms, color trials and approximate counting techniques. Some of which might be of independent interest.
- Chapter 12 provides the $(\Delta + 1)$-coloring algorithm for virtual graphs *of high degree* (Result 5). It has the same general structure as the algorithm for broadcast CONGEST presented in Chapter 4, except that most steps require additional care. Especially in the densest almost-cliques, termed cabals, where computing the colorful matching and coloring the put-aside sets requires novel approaches.
- Chapter 13 completes Chapter 12 with an algorithm for $(\Delta + 1)$-coloring graphs in $\text{poly}(\log \log n)$ rounds when $\Delta \leqslant \text{poly}(\log n)$ (Result 5). The main technical ingredient of this chapter is an adaptation of the Ghaffari-Kuhn algorithm using approximate counting techniques from Chapter 11.

1.5. Publications Constituting This Thesis

This thesis is based on a subset of my publications. I am a main author in all of these publications. However, as is customary in theoretical computer science, authors are listed in alphabetical order. Since there is very significant overlap between each of the publications, this thesis reorganizes their content with consistent notation in an effort to tie the whole together. Chapter 3 is based on the work of [SW10; BEPS16; HSS18; ACK19; CLP20; HNT21; HKNT22].

Part I is based on

- Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. "Coloring Fast with Broadcasts". In: *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2023, Orlando, FL, USA, June 17-19, 2023*. ACM, 2023, pp. 455–465. DOI: 10.1145/3558481.3591095

Part II is based on

- Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. "A Distributed Palette Sparsification Theorem". In: *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*. SIAM, 2024. DOI: 10.1137/1.9781611977912.142

Part III is based on

- Maxime Flin, Magnús M. Halldórsson, and Alexandre Nolin. "Fast Coloring Despite Congested Relays". In: *37th International Symposium on Distributed Computing, DISC 2023, October 10-12, 2023, L'Aquila, Italy*. Vol. 281. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 19:1–19:24. DOI: 10.4230/LIPICS.DISC.2023.19
- Maxime Flin, Magnús M. Halldórsson, and Alexandre Nolin. "Decentralized Distributed Graph Coloring: Cluster Graphs". In: *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2025, Hotel Las Brisas Huatulco, Huatulco, Mexico, June 16-20, 2025*. Ed. by Alkida Balliu and Fabian Kuhn. ACM, 2025, pp. 394–405. DOI: 10.1145/3732772.3733549
- Maxime Flin, Magnús M. Halldórsson, and Alexandre Nolin. "Decentralized Distributed Graph Coloring II: Degree+1-Coloring Virtual Graphs". In: *38th International Symposium on Distributed Computing, DISC 2024, October 28 to November 1, 2024, Madrid, Spain*. Ed. by Dan Alistarh. Vol. 319. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 24:1–24:22. DOI: 10.4230/LIPICS.DISC.2024.24

The following results are *not* part of this thesis:

- Maxime Flin and Parth Mittal. "$(\Delta + 1)$ vertex coloring in O(n) communication". In: *Distributed Comput.* 38.1 (2025), pp. 19–29. DOI: 10.1007/S00446-024-00475-3 **Invited to the Special Issue of PODC 2024.**
- Keren Censor-Hillel, Tomer Even, Maxime Flin, and Magnús M. Halldórsson. "When MIS and Maximal Matching are Easy in the

Congested Clique". In: *Structural Information and Communication Complexity - 32nd International Colloquium, SIROCCO 2025, Delphi, Greece, June 2-4, 2025, Proceedings*. Ed. by Ulrich Schmid and Roman Kuznets. Vol. 15671. Lecture Notes in Computer Science. Springer, 2025, pp. 194–210. DOI: 10.1007/978-3-031-91736-3_12

- Maxime Flin and Magnús M. Halldórsson. "Faster Dynamic (Δ+1)-Coloring Against Adaptive Adversaries". In: *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, July 8-11, 2025, Aarhus, Denmark*. Ed. by Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis. Vol. 334. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025, 79:1–79:21. DOI: 10.4230/LIPICS.ICALP.2025.79

CHAPTER 2

Background

2.1. Notation

We introduce here general notations and conventions that we *try* to use consistently throughout this manuscript.

Letters $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{R}$ denote the set of integers, non-negative integers, and real numbers. Occasionally, we use $\mathbb{N}_{\geqslant 0}$, $\mathbb{N}_{\geqslant 1}$ or $\mathbb{R}_{\geqslant 1}$ to restrict the set to its members above some minimal value. For an integer $t \geqslant 1$, let $[t] \stackrel{\text{def}}{=} \{1, 2, \ldots, t\}$. Logarithms in base 2 and e are respectively denoted by log and ln. The log-star function is recursively defined as

$$\log^* : x \in \mathbb{R} \mapsto \begin{cases} 0 & \text{if } x \leqslant 1 \\ 1 + \log^*(\log x) & \text{otherwise} \end{cases}$$

The converse operator, tetration, is denoted using the Knuth's up arrow notation $a \uparrow\uparrow n = a^{a\uparrow\uparrow(n-1)}$ and $a \uparrow\uparrow 0 = 1$.

Sets and Functions. For any set X, we use 2^X as shorthand for the set of subsets of X. For a function $f : \mathcal{X} \to \mathcal{Y}$, the image of a set $X \subseteq \mathcal{X}$ is $f(X) \stackrel{\text{def}}{=} \{f(x) : x \in X\}$; conversely, the pre-image of $Y \subseteq \mathcal{Y}$ is $f^{-1}(Y) \stackrel{\text{def}}{=} \{x \in X : f(x) \in Y\}$. When Y is a singleton, we abuse notation and write $f^{-1}(y) \stackrel{\text{def}}{=} f^{-1}(\{y\})$. For a tuple $x = (x_1, x_2, \ldots, x_n)$ and $i \in [n]$, we write $x_{\leqslant i} = (x_1, \ldots, x_i)$ (and similarly for $x_{<i}$, $x_{>i}$ and $x_{\geqslant i}$) and $x_{\neg i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. When there are multiple indices $x_{i,j}$ where $i \in [n]$ and $j \in [m]$, we write $x_{i,*} = (x_{i,1}, x_{i,2}, \ldots, x_{i,m})$ and $x_{*,j} = (x_{1,j}, x_{2,j}, \ldots, x_{n,j})$.

Asymptotic Notation. Usually, letters γ, c and C denote absolute constants. Their values are not necessarily consistent across sections. When we mean to use constants set earlier, we append a subscript to its definition. We write $f \ll g$ or $f = O(g)$ if an absolute constant $c > 0$ exists for which

$f < cg$. Conversely, we write $f \gg g$ for $g \ll f$. If c depends on a parameter ε, we may explicit the dependency by writing $f \ll_\varepsilon g$ or $f = O_\varepsilon(g)$. We also write $\mathrm{poly}(n)$ for n^c for some absolute constant $c > 0$. We occasionally use $\widetilde{O}(f)$ to hide $\mathrm{poly}(\log f)$ factors and $\widetilde{\Omega}(f)$ to hide $1/\mathrm{poly}(\log f)$ factors.

Graphs. For a graph $G = (V_G, E_G)$, the set of neighbors of v in G are $N_G(v) \stackrel{\text{def}}{=} \{u \in V_G : \{u, v\} \in E_G\}$. For a set $S \subseteq V_G$, the subgraph induced by S, denoted $G[S]$, has vertex set S and edges set $\{\{u, v\} \in E_G : u, v \in S\}$. The degree of v in G is $\deg(v; G) \stackrel{\text{def}}{=} |N_G(v)|$ its number of neighbors. The closed neighborhood is $N_G[v] = \{v\} \cup N_G(v)$. The distance-$k$ neighborhood is $N_G^k(v)$ the set of vertices at distance at most k (and at least one) from v in G — and the closed distance-k neighborhood includes v. When G is clear from context, we drop the subscript and write $N(v) = N_G(v)$ and $\deg(v) = \deg(v; G)$. We call *anti-edge* or *non-edge* an unordered pair of V_G such that $\{u, v\} \notin E_G$.

Colorings. For any integer $q \geqslant 1$, a *partial q-coloring* of a graph G is a function $\varphi : V_G \to [q] \cup \{\perp\}$ where $\perp$ means "not colored". The domain $\operatorname{dom} \varphi \stackrel{\text{def}}{=} \{v \in V_G : \varphi(v) \neq \perp\}$ of φ is the set of colored nodes. A coloring φ is *total* when all nodes are colored, i.e., $\operatorname{dom} \varphi = V_G$; and we say it is *proper* if $\perp \in \varphi(\{u, v\})$ or $\varphi(v) \neq \varphi(u)$ whenever $\{u, v\} \in E_G$. We write that $\psi \succeq \varphi$ when a partial coloring ψ *extends* φ: for all $v \in \operatorname{dom} \varphi$, we have $\psi(v) = \varphi(v)$. The *uncolored degree* $\deg_\varphi(v) \stackrel{\text{def}}{=} |N(v) \setminus \operatorname{dom} \varphi|$ of v with respect to φ is the number of uncolored neighbors of v.

For lists $L : V_G \to 2^{\mathbb{N}}$, an L-list-coloring is a coloring φ such that $\varphi(v) \in L(v)$ for every vertex. For instance, a q-coloring is a list-coloring with respect to lists $L(v) = [q]$ for every vertex. The *palette* of v with respect to a L-list-coloring φ is $L_\varphi(v) = L(v) \setminus \varphi(N(v))$: the set of colors available to extend φ at v. Most of this thesis is concerned with $(\Delta + 1)$-coloring, so we often implicitly assume $L(v) = [\Delta + 1]$. Occasionally, the algorithms will use different lists which we shall call List to emphasize the distinction. Lastly, the $(\deg +1)$-list-coloring problem refers to the setting where each vertex v has a list of $\deg(v) + 1$ many colors. Observe that a partially $(\Delta + 1)$-colored graph induces a instance of the $(\deg +1)$-list-coloring problem.

Probability. Random variables are represented by bold letters such as $\mathbf{X}$, $\mathbf{Y}$ or $\boldsymbol{\chi}$. The support of $\mathbf{X}$ is the set of outcomes x such that $\mathbb{P}[\mathbf{X} = x] > 0$. The indicator random variable $\mathbf{X}$ for an event $\mathcal{E}$ is the variable equal to one when $\mathcal{E}$ holds and to zero otherwise. For brevity, we sometimes say that $\mathbf{X}$ indicates $\mathcal{E}$.

The terms "w.p.", "w.h.p." and "w.e.h.p." abridge "with probability", "with high probability" and "with exponentially high probability". High probability means with a probability close to one up to an additive term that is polynomially small in the size of the problem. For graph problems, "w.h.p." typically means with probability $1-1/n^c$ where n is the number of vertices in the graph and c a desirably large constant. Exponentially high probability in some parameter m means with probability $1-\exp(-\Omega(m))$.

2.2. Distributed Graph Algorithms

The LOCAL Model. In the LOCAL model, we are given a graph G which we see as a network. Each vertex is a computer running the same algorithm. Initially, vertices know the number of vertices n and their neighbors in the graph. They communicate on their adjacent edges in synchronous rounds, without message loss or corruption, and transmission occurs simultaneously everywhere in the graph. In a randomized LOCAL algorithms, each vertex is given an infinite string of random bit. In a deterministic algorithm, vertices are given unique identifiers, usually of $O(\log n)$ bits, where n is the number of vertices in G. Observe that a randomized algorithm can generate unique $\Theta(\log n)$-bit identifiers w.h.p. in n without any communication (by sampling random identifiers). In this thesis, we always implicitly assume that vertices have access to random bits. We denote the unique identifiers of v by $\mathsf{ID}(v)$ and

$$N_<(v) = \{u \in N(v) : \mathsf{ID}(u) < \mathsf{ID}(v)\} ,$$

and similarly for $N_<[v]$ with the closed neighborhood and $N_>(v)$ with larger identifiers. When n is not known exactly, replacing it by a polynomial upper bound on the size of the graph increases the size of identifiers by a constant factors. As is customary, in this thesis, we assume that vertices know Δ,

the maximum degree of the graph, although it is a global parameter like n. See [HS20] for a more detailed discussion of the LOCAL model.

The CONGEST Model. The CONGEST model is the LOCAL model with bandwidth constraints: all messages contain only $O(\log n)$ bits. Still, during the same round, vertices can send different $O(\log n)$-bit sized messages to different neighbors. The *broadcast* CONGEST model, denoted by BCONGEST, is a variant of the CONGEST model in which vertices send the same $O(\log n)$-bit message to all their neighbors at each round. See [HS20, Chapter 5] for a more detailed discussion.

Deterministic $(\Delta+1)$-Coloring. The current best deterministic $(\Delta+1)$-coloring algorithm, in CONGEST and even BCONGEST, is an algorithm of Ghaffari and Kuhn with $O(\log^2 \Delta \log n)$-round complexity. In LOCAL, the algorithm has been improved further by [FGGKR23; GG23; GG24] all the way to $\widetilde{O}(\log^{5/3} n)$. Since we use it multiple time in this thesis, let us state the result of Ghaffari and Kuhn.

THEOREM 2.1 ([GK21, Theorem 4.2]). *There is a deterministic* BCONGEST *algorithm that solves the* $(\deg+1)$*-list-coloring problem in* $O(\log^2 |\mathscr{C}| \cdot \log n)$ *rounds, using messages of* $O(\log |\mathscr{C}|)$ *bits. Here,* $\mathscr{C}$ *denotes a set of colors known to all vertices such that every vertex has a list* $L(v) \subseteq \mathscr{C}$ *of at least* $\deg(v)+1$ *colors.*

Models with Global Communication. The CONGESTED-CLIQUE model is the same as the CONGEST model except that, in each round, a vertex can receive/send a $O(\log n)$ bit message from/to any other one — even if they are not adjacent. In this model, $(\Delta+1)$-coloring can be solved deterministically in $O(1)$ rounds [CDP21b, Theorem 1].

In the Massively Parallel Computation, or MPC, model, the input is partitioned between machines with S words (or $O(S \log n)$ bits) of local space each. The number of machines is guaranteed to be large enough to contain the whole input, e.g., if the graph has n vertices and m edges, there are at least $(n+m)/S$ machines. Machine can perform local computation and each round they receive/send up to S bits from/to any other machine. When machines have linear local space, i.e., $S = \Omega(n)$, they can solve

$(\Delta+1)$-coloring deterministically in $O(1)$ rounds [CDP21b, Theorem 2]. In the low-space regime, when $S = n^\delta$ for some $\delta \in (0,1)$, the $(\Delta+1)$-coloring problem can be solved in $O(\log\log\log n)$ rounds [CDP21a].

In the Node-Capacitated Clique, or NCC, model, the n vertices initially know about all edges incident to them and may receive/send $O(\log n)$-bit messages from/to any $O(\log n)$ vertices in the graph. The only known coloring result in this model (prior to this thesis) was $O(a)$-coloring, with time linear in a, where a is the arboricity of the graph [AGGHSKL19]. The question of efficient $(\Delta+1)$-coloring has remained open.

Other Applications. Since its introduction by Linial, the interest in the LOCAL model has expanded beyond the original aim to study locality of graph algorithms. Bridges were created between the complexity landscape of LOCAL algorithms [CP19] and the study of descriptive combinatorics [Ber23] and finitary factors [GR23]. We refer readers to [Roz24] for a survey of recent progress in the LOCAL model and its various applications.

2.3. Concentration Inequalities

Since almost all of the algorithms presented here are randomized, the use of concentration bounds to analyze their success probability is ubiquitous. When variables are independent, we use the standard form of Chernoff Bound. We refer readers to [DP09] or [Doe20, Section 1.10] for more details.

Lemma 2.2 (Chernoff bounds)**.** *Let $\mathbf{X}_1, \dots, \mathbf{X}_n$ be a family of independent random variables with values in $[0,1]$, and let $\mathbf{X} = \sum_{i\in[n]} \mathbf{X}_i$. Suppose $\mu_\mathsf{L} \leqslant \mathbb{E}[\mathbf{X}] \leqslant \mu_\mathsf{H}$, then*

$$\forall \delta \geqslant 0, \qquad \mathbb{P}[\mathbf{X} > (1+\delta)\mu_\mathsf{H}] \leqslant \exp\left(-\frac{\delta^2}{2+\delta}\mu_\mathsf{H}\right). \tag{2.1}$$

$$\forall \delta \in (0,1), \qquad \mathbb{P}[\mathbf{X} < (1-\delta)\mu_\mathsf{L}] \leqslant \exp\left(-\frac{\delta^2}{2}\mu_\mathsf{L}\right). \tag{2.2}$$

$$\forall t > 0, \qquad \mathbb{P}[\mathbf{X} < \mu_\mathsf{L} - t],\ \mathbb{P}[\mathbf{X} > \mu_\mathsf{H} + t] \leqslant \exp\left(-2t^2/n\right). \tag{2.3}$$

Often, random variables are not independent. However, if they obey a certain form of stochastic domination, Chernoff-like Bounds still hold. Since this form of the inequality is relatively unusual, we provide a quick proof. We remark that similar results have been used previously; see, e.g.,

[Doe20, Section 1.8.3] or [KQ21] and references therein. We refer readers to [DP09, Sections 5.1 and 5.2] for definition and properties of the conditional expectation.

Lemma 2.3 (Chernoff bounds with stochastic domination)**.** *Let $\mathbf{X}_1, \dots, \mathbf{X}_n$ be random variables and $\mathbf{Y}_i = f_i(\mathbf{X}_1, \dots, \mathbf{X}_i) \in [0,1]$ be any function of the first i variables with value in $[0,1]$. Consider $\mathbf{Y} = \sum_{i=1}^n \mathbf{Y}_i$.*

If there exists $q_1, \dots, q_n \in [0,1]$ such that $\mathbb{E}[\mathbf{Y}_i \mid \mathbf{X}_1, \dots, \mathbf{X}_{i-1}] \leqslant q_i$ for every $i \in [n]$, then for any $\delta > 0$,

$$\mathbb{P}[\mathbf{Y} \geqslant (1+\delta)\mu] \leqslant \exp\left(-\frac{\delta^2}{2+\delta}\mu\right) \quad \text{where} \quad \mu \geqslant \sum_{i=1}^n q_i \ . \tag{2.4}$$

Conversely, if there exists $q_1, \dots, q_n \in [0,1]$ such that $\mathbb{E}[\mathbf{Y}_i \mid \mathbf{X}_1, \dots, \mathbf{X}_{i-1}] \geqslant q_i$ for every $i \in [n]$, then for any $\delta \in (0,1)$,

$$\mathbb{P}[\mathbf{Y} \leqslant (1-\delta)\mu] \leqslant \exp\left(-\frac{\delta^2}{2}\mu\right) \quad \text{where} \quad \mu \leqslant \sum_{i=1}^n q_i \ . \tag{2.5}$$

Proof. We make the argument only for Eq (2.4); the proof of Eq (2.5) is essentially the same. Following the classical Chernoff bound proof, we introduce a real parameter $t > 0$

$$\mathbb{P}[\mathbf{Y} \geqslant m] \leqslant e^{-tm}\,\mathbb{E}\left[\prod_{i=1}^n e^{t\mathbf{Y}_i}\right] = e^{-tm}\,\mathbb{E}\left[\mathbb{E}\left[e^{t\mathbf{Y}_n} \mid \mathbf{X}_1, \dots, \mathbf{X}_{n-1}\right] \prod_{i=1}^{n-1} e^{t\mathbf{Y}_i}\right]$$

using the law of total expectation and that every $\mathbf{Y}_i$ except $\mathbf{Y}_n$ is a function of $\mathbf{X}_1, \dots, \mathbf{X}_{n-1}$. Since $\mathbf{Y}_n \in [0,1]$ and $\mathbb{E}[\mathbf{Y}_n \mid \mathbf{X}_1, \dots, \mathbf{X}_{n-1}] \leqslant q_n$, a convexity argument implies that $\mathbb{E}\left[e^{t\mathbf{Y}_n} \mid \mathbf{X}_1, \dots, \mathbf{X}_{n-1}\right] \leqslant q_n e^t + 1 - q_n$ (see [DP09, Section 1.5] for instance). Repeating the argument above thus yields

$$\mathbb{P}[\mathbf{Y} \geqslant m] \leqslant e^{-tm} \prod_{i=1}^n (q_i e^t + 1 - q_i)$$

which then imply Eq (2.4) following the usual steps of the Chernoff bound's proof (see, e.g., [Doe20, Section 1.10.1.7]). ∎

Note that the order on the variables $\mathbf{X}_1, \dots, \mathbf{X}_n$ in Lemma 2.3 is important because $\mathbf{Y}_i$ can depend only on the first i random variables. In

some cases, this is not necessary and can be a technical nuisance. A different form of domination, called Bernoulli domination, may be used to apply Eq (2.4).

Lemma 2.4 (Chernoff with Ber(p)-domination, [PS97, Theorem 3.4])**.** *Let $p \in (0,1]$ and $\mathbf{X}_1, \dots, \mathbf{X}_n$ be binary random variables such that for any $S \subseteq [n]$, we have that,*

$$\mathbb{E}\left[\prod_{i \in S} \mathbf{X}_i\right] \leqslant p^{|S|} \ .$$

Then Eq (2.4) applies to $\mathbf{Y} = \sum_{i=1}^{n} \mathbf{X}_i$ and $q_i = p$ for all $i \in [n]$.

We briefly remark that while all aforementionned versions of the Chernoff Bound apply to random variables with values in $[0,1]$, by a simple rescaling trick they all extend to sums of variables with values in $[0,c]$ except that μ is replaced by μ/c on the right-hand side, i.e., Eq (2.4) becomes $\exp\left(-\frac{\delta^2 \mu}{(2+\delta)c}\right)$ and Eq (2.5) $\exp\left(-\delta^2\mu/(2c)\right)$.

When we sample vertices at random, Janson's inequality is useful to guarantee the existence of some structures in the subgraph induced by the sampled vertices.

Lemma 2.5 (Janson's Inequality, [DP09, Section 3.3])**.** *Let Ω be a ground set and let $\mathbf{A}$ be a subset obtained by sampling each $x \in \Omega$ independently with probability $p_x \in [0,1]$. Let $\mathcal{S}$ be a family of subsets of Ω and for each $S \in \mathcal{S}$ let $\mathbf{X}_S$ indicate if $S \subseteq \mathbf{A}$. Define*

$$\mathbf{X} = \sum_{S \in \mathcal{S}} \mathbf{X}_S, \quad \mu = \mathbb{E}[\mathbf{X}] \quad \text{and} \quad K = \frac{1}{2} \sum_{S \cap S' \neq \emptyset} \mathbb{E}[\mathbf{X}_S \mathbf{X}_{S'}] \ .$$

Then, for any $0 \leqslant t \leqslant \mu$, we have

$$\mathbb{P}[\mathbf{X} < \mu - t] \leqslant \exp\left(-\frac{t^2}{2\mu + K}\right) \ .$$

Let $\mathbf{Y}_1, \dots, \mathbf{Y}_n$ be boolean random variables (with values in $\{0,1\}$). We say they form a *read-k family* if they can be expressed as a function of independent random variables $\mathbf{X}_1, \dots, \mathbf{X}_m$ such that each $\mathbf{X}_j$ influences k variables Y_i at most. More formally, there are sets $P_i \subseteq [m]$ for each $i \in [n]$ such that: (1) for each $i \in [n]$, the variable $\mathbf{Y}_i$ is a function of $\{\mathbf{X}_j : j \in P_i\}$ and (2) $|\{i : j \in P_i\}| \leqslant k$ for each $j \in [m]$.

Lemma 2.6 (read-k bound, [GLSS15])**.** *Let $\mathbf{Y}_1, \dots, \mathbf{Y}_n$ be a read-k family of boolean variables, and let $\mathbf{Y}$ be their sum. Then, for any $\delta > 0$,*

$$\mathbb{P}\left[\left|\mathbf{Y} - \mathbb{E}[\mathbf{Y}]\right| > \delta n\right] \leqslant 2\exp\left(-\frac{2\delta^2 n}{k}\right).$$

Let f be any real-valued function on an n-dimensional product space. For $c \in \mathbb{R}_{\geqslant 0}$, the function f is *c-Lipschitz* if and only if for every $x = (x_1, \dots, x_n)$, changing any single x_i affects the value of $f(x)$ by at most c in absolute value. For $r \in \mathbb{N}$, the function f is *r-certifiable* if for every x there exists a set $I(x) \subseteq [n]$ of $r \cdot f(x)$ indices such that for all x' such that $x'_i = x_i$ for all $i \in I(x)$, we have $f(x') \geqslant f(x)$. In words, fixing the values of indices in $I(x)$ certifies that f is at least $f(x)$.

Lemma 2.7 (Talagrand's inequality [DP09, Section 11.3])**.** *Let $f(x_1, \dots, x_n)$ be a c-Lipschitz r-certifiable function. Let $\mathbf{X}_1, \dots, \mathbf{X}_n$ be independent random variables and $f(\mathbf{X}) = f(\mathbf{X}_1, \dots, \mathbf{X}_n)$. Then for $t \geqslant 1$,*

$$\mathbb{P}\left[\left|f(\mathbf{X}) - \mathbb{E}[f(\mathbf{X})]\right| > t + 30c\sqrt{r \cdot \mathbb{E}[f(\mathbf{X})]}\right] \leqslant 4 \cdot \exp\left(-\frac{t^2}{8c^2 r\, \mathbb{E}[f(\mathbf{X})]}\right).$$

2.4. Families of Hash Functions

It will be occasionally useful to represent elements (e.g., vertices or colors) using very few bits. Hashing is a common technique to compress communication so we list here the types of explicit families of hash functions we will need in this thesis.

Definition 2.8 (k-Wise Independence)**.** Let $N, M, k \geqslant 1$ such that $k \leqslant N$. A family of functions $\mathscr{H}$ from $[N]$ to $[M]$ is said to be k-wise independent if for every $x_1, x_2, \dots, x_k \in [N]$ the random variables $h(x_1), \dots, h(x_k)$ are independent and uniformly distributed in $[N]$ when h is sampled uniformly in $\mathscr{H}$.

See [Vad12, Section 3.5.5] for a proof of Theorem 2.9.

Theorem 2.9. *For any $N, M, k > 0$ with $k \leqslant N$, there exists an explicit family $\mathscr{H}$ of pairwise independent functions $[N] \to [M]$ such that describing $h \in \mathscr{H}$ requires $O(k(\log N + \log M))$ bits.*

By weakening the uniformity of the hashes, we can improve the dependency on N.

Definition 2.10 (Almost Pairwise Independence)**.** Let $N, M \in \mathbb{N}$ and $\varepsilon > 0$. A family of functions $\mathscr{H}$ from $[N]$ to $[M]$ is said to be ε-almost-pairwise independent if for every $x_1 \neq x_2 \in [N]$ and $y_1, y_2 \in [M]$, we have

$$\mathbb{P}_{h \in \mathscr{H}}[h(x_1) = y_1 \text{ and } h(x_2) = y_2] \leqslant \frac{1+\varepsilon}{M^2} \ .$$

We refer readers to [Vad12, Problem 3.4] for a proof of the following theorem.

THEOREM 2.11. *For any $N, M \in \mathbb{N}$ and $\varepsilon > 0$, there exists an explicit ε-almost-pairwise independent family $\mathscr{H}$ of functions $[N] \to [M]$ such that describing $h \in \mathscr{H}$ requires $O(\log\log N + \log M + \log 1/\varepsilon)$ bits.*

Definition 2.12 (Min-Wise Independence)**.** A family of functions $\mathscr{H}$ from $[n]$ to $[n]$ is said to be (ε, s)-min-wise independent if for any $X \subseteq [n]$, $|X| \leqslant s$ and $x \in [n] \setminus X$ we have:

$$\left| \mathbb{P}_{h \in \mathscr{H}}[h(x) < \min h(X)] - \frac{1}{|X|+1} \right| \leqslant \frac{\varepsilon}{|X|+1} \ .$$

Lemma 2.13 ([Ind01])**.** *There exists some constant $C_{2.13} > 0$ such that for any $N \geqslant 2$, $\varepsilon > 0$ and $s \leqslant \varepsilon N / C_{2.13}$ any $O(\log 1/\varepsilon)$-wise independent family of functions $[N] \to [N]$ is (ε, s)-min-wise independent. In particular, a function in such a family can be described using $O(\log N \cdot \log 1/\varepsilon)$ bits.*

2.5. Information Theory

Let $\mathbf{A}, \mathbf{B}, \mathbf{C}$ be three random variables of respective support $\mathscr{A}, \mathscr{B}, \mathscr{C}$. Let us slightly abuse notation by taking the convention that $0 \cdot \log(1/0) = 0$, $0 \cdot \log(0/0) = 0$ and $c \cdot \log(1/0) = \mathrm{sign}(c)\infty$ for $c \neq 0$.

Entropy: The *entropy* $H(\mathbf{A})$ of $\mathbf{A}$ is defined as

$$H(\mathbf{A}) \stackrel{\text{def}}{=} \sum_{a \in \mathscr{A}} \mathbb{P}[\mathbf{A} = a] \cdot \log\left(\frac{1}{\mathbb{P}[\mathbf{A} = a]}\right) .$$

Conditional entropy: The *entropy of* $\mathbf{A}$ *conditioned on* $\mathbf{B}$ is defined as

$$H(\mathbf{A} \mid \mathbf{B}) \stackrel{\text{def}}{=} \sum_{b \in \mathscr{B}} \mathbb{P}[\mathbf{B} = b] \cdot \sum_{a \in \mathscr{A}} \mathbb{P}[\mathbf{A} = a \mid \mathbf{B} = b] \cdot \log\left(\frac{1}{\mathbb{P}[\mathbf{A} = a \mid \mathbf{B} = b]}\right) .$$

Joint distribution: The *joint distribution* $\mathbf{AB}$ of $\mathbf{A}$ and $\mathbf{B}$ is the distribution over $\mathscr{A} \times \mathscr{B}$ s.t. $\forall (a,b) \in \mathscr{A} \times \mathscr{B}, \mathbb{P}[\mathbf{AB} = (a,b)] = \mathbb{P}[\mathbf{A} = a \wedge \mathbf{B} = b]$.

As immediate properties, we have $H(\mathbf{AB}) \leqslant H(\mathbf{A}) + H(\mathbf{B})$, with equality when $\mathbf{A}$ and $\mathbf{B}$ are independent, and $H(\mathbf{A} \mid \mathbf{B}) = H(\mathbf{AB}) - H(\mathbf{B})$.

Mutual information: The *mutual information* between $\mathbf{A}$ and $\mathbf{B}$ is defined as

$$I(\mathbf{A} : \mathbf{B}) \stackrel{\text{def}}{=} H(\mathbf{A}) + H(\mathbf{B}) - H(\mathbf{AB}) \ .$$

As with entropy, we define mutual information conditioned on a third variable $\mathbf{C}$ as

$$I(\mathbf{A} : \mathbf{B} \mid \mathbf{C}) \stackrel{\text{def}}{=} H(\mathbf{A} \mid \mathbf{C}) + H(\mathbf{B} \mid \mathbf{C}) - H(\mathbf{AB} \mid \mathbf{C}) \ .$$

As immediate property, we have $I(\mathbf{A} : \mathbf{B} \mid \mathbf{C}) + H(\mathbf{C}) = I(\mathbf{AC} : \mathbf{BC})$.

Kullback-Leibler divergence: The *(Kullback-Leibler) divergence* between two distributions $p(x)$ and $q(x)$ over a common support $\mathcal{X}$ is

$$\mathsf{D}(p \parallel q) \stackrel{\text{def}}{=} \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} = \mathop{\mathbb{E}}_{p(x)} \left[\log \frac{p(x)}{q(x)} \right] \ .$$

Note that the divergence is $+\infty$ when for some element x we have $q(x) = 0$ but $p(x) \neq 0$.

Total variation distance: The *total variation distance* between two distributions $p(x)$ and $q(x)$ over a common support $\mathcal{X}$ is

$$\|p - q\|_1 = \sum_{x \in \mathcal{X}} |p(x) - q(x)| = 2 \max_{S \subseteq \mathcal{X}} (p(S) - q(S)) \ .$$

The notation $\|.\|_1$ highlights that this distance is equivalent to the ℓ_1-distance between p and q, defined as the ℓ_1-norm of the vector $p - q$.

Lemma 2.14 (Pinsker's inequality)**.**

$$\|p - q\|_1 \leqslant \sqrt{\frac{\ln(2)}{2} \cdot \mathsf{D}(p \parallel q)} \ .$$

Corollary 2.15. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two random variables of distributions* $p_{\mathbf{A}}$ *and* $p_{\mathbf{B}}$*, with* $p_{\mathbf{A}|\mathbf{B}=b}$ *the distribution of* $\mathbf{A}$ *conditioned on* $\mathbf{B} = b$*. We have*

$$\mathop{\mathbb{E}}_{b\sim\mathbf{B}}\Big[\big\|p_{\mathbf{A}} - p_{\mathbf{A}|\mathbf{B}=b}\big\|_1\Big] \leqslant \sqrt{\frac{\ln(2)}{2}\cdot I(\mathbf{A}:\mathbf{B})}\ .$$

The corollary follows from Pinsker's inequality, together with $I(A : B) = \mathbb{E}_{b\sim B}\big[\mathsf{D}(p_{A|B=b} \parallel p_A)\big]$, the concavity of $x \to \sqrt{x}$, and Jensen's inequality.

CHAPTER 3

$(\Delta+1)$-Coloring in Local

Despite the LOCAL model being much stronger than most of the other models considered in this thesis, fast $(\Delta+1)$-coloring in this model results from a long line of research. As this thesis extends this line of research, we find it useful to first review the state-of-the-art algorithm for $(\Delta+1)$-coloring in LOCAL.

THEOREM 3.1 ([CLP20; GG24]). *There is a randomized* LOCAL *algorithm that $(\Delta+1)$-colors any n-vertex graph with maximum degree Δ in $\widetilde{O}(\log^{5/3}\log n)$ with high probability. When $\Delta \geqslant (c\log n)^{3.1}$, it ends after $O(\log^* n)$ rounds.*

This chapter aims at providing a self-contained proof of Theorem 3.1 *when Δ is large*. Along the way, we formally introduce key concepts for the remainder of this thesis (slack, multi-color trial, slack generation, almost-clique decomposition, synchronized color trial). Our focus is on the high-degree case because it avoids delicate technical details that are not salient for the remainder of this thesis. We refer interested readers to Section 3.7 for more details on this issue.

The results of this chapter are aggregated from many earlier publications [SW10; HSS18; ACK19; CLP20; AW22; HKMT21; HKNT22] and the less novel parts of [FHN23] by the author. In an attempt to make this thesis self-contained, and since Theorem 3.1 (when Δ is large) is the starting point of this thesis work, we provide complete proofs for each statement. Beyond setting the stage for future chapters, we hope this chapter will be a useful synthesis for any reader interested in learning about randomized $(\Delta+1)$-coloring in LOCAL.

Organization of this Chapter. Each section from Section 3.2 to Section 3.5 corresponds to one of the main steps of the algorithm. Section 3.6

puts all pieces together and gives a proof of Theorem 3.1 when Δ is large. Each section begins with a discussion of the main ideas along with definitions of key concepts and general formal statements. Proofs appear in the later part of each section and are often stated in a more general form than is needed in LOCAL so that they can be used in future chapters.

3.1. Overview

We begin with a streamlined presentation of the $(\Delta+1)$-coloring meta-algorithm, based on the prior work of [SW10; HSS18; ACK19; CLP20; HKNT22], that is underlying all our work. We emphasize that aspects of this presentation are oversimplified for the sake of intuition.

Ultrafast Coloring with Slack (Section 3.4). The slack of a vertex is the difference between the number of its neighbors that attempt to get colored and the number of colors it has available. What makes slack essential to randomized coloring is an idea of Schneider and Wattenhofer [SW10] (reformulated many times, e.g., in [EPS15; HSS18; CLP20]) that colors the graph in $O(\log^* n)$ rounds when vertices have slack linear in their uncolored degree. Suppose there exists some integer x such that every vertex has $(1+2x)\deg(v)$ colors available and $\deg(v)$ uncolored neighbors. If every vertex tries x available colors, the neighbors of v block at most $x\deg(v)$ colors, leaving at least half of the colors free. So v remains uncolored with probability at most 2^{-x}. For nodes that remain, the degree decreased by a 2^x factor while the slack does not decrease. For the next iteration, we can therefore increase x to 2^x. After $O(\log^* n)$ iteration, $x = \Theta(\log n)$ and every vertex gets colored with high probability.

For the $(\Delta+1)$-coloring problem, vertices only begin with one unit of slack, which does not suffice for ultrafast coloring. Nonetheless, it suffices that the slack be linear in the uncolored degree, for a randomized color trial reduces uncolored degrees by constant factor. The key challenge for sublogarithmic $(\Delta+1)$-coloring is to generate such slack.

Slack Generation (Section 3.3). The main mean of providing a vertex with slack is by coloring pairs of neighbors with the same color. Of course, this is only possible if the vertex has pairs of neighbors that are not adjacent.

This motivates the notion of sparsity (also sometimes called local-sparsity), which counts the number of edges missing from the neighborhood of a vertex: a vertex has sparsity ζ if its neighborhood contains at most $\binom{\Delta}{2} - \zeta\Delta$ edges. The essential observation (already made in [MR02, Chapter 10]), is that a single random color trial provides vertices with slack linear in their sparsity. In particular, vertices with $\Omega(\Delta)$ sparsity can be colored in $O(\log^* n)$ rounds. More challenging will be to color the denser vertices for which $\Omega(\Delta)$ is not achievable, for instance, in the case of a $(\Delta+1)$-clique.

The Sparse-Dense Decomposition (Section 3.2). The structural decomposition of [HSS18; ACK19] (inspired by [Ree98]) classifies vertices according to their sparsity. A sparse vertex is one with sparsity $\Omega(\Delta)$ and every other vertex is dense. Dense vertices are organized into *ε-almost-cliques* (Definition 3.5): a cluster of $(1 \pm \Theta(\varepsilon))\Delta$ vertices in which every vertex is adjacent to at least $(1-\varepsilon)\Delta$ other vertices in its cluster, where ε is a small constant. The key insight is that almost-cliques are structured around their densest vertices. As such, no vertex in an almost-clique is much denser than the average density of the almost-clique (Corollary 3.9). And the denser the almost-clique, the less dependencies it has with the rest of the graph. For instance, a $(\Delta+1)$-clique, which is the densest possible almost-clique, is trivial to color in two rounds: one vertex can distribute a different color to every other vertex of the clique. As the sparsity of the almost-clique increases, random color trials become more effective (e.g., slack generation provides more slack), while contention with the rest of the graph becomes more important.

Synchronized Color Trial (Section 3.5). Simple color trials are ineffective for dense vertices, for conflicts are too likely to arise between vertices of the same almost-clique. The key insight of [HSS18] was to take advantage of the high density of almost-cliques to synchronize color trials on their inside. All sub-logarithmic $(\Delta+1)$-coloring algorithms have some form of synchronized color trial. We follow the approach of [HKNT22; FGHKN23] in which vertices are shuffled uniformly at random and try the unused color corresponding to its random rank. This precludes all conflicts between vertices of the same almost-clique. As such, the number of uncolored vertices

in each almost-clique is linear in its average external degree, i.e., the average number of edges its vertices have going out of the almost-clique. As it turns out, vertices of an almost-clique have a sparsity linear in this amount; because each connection to the outside yields some anti-edges between the outside and the inside. Dense vertices can therefore be colored by the multicolor trial algorithm after one random color trial (to generate slack) and one synchronized color trial.

Put-Aside Sets (Section 3.5.2). The multicolor trial then colors all remaining dense vertices, *except for those in the densest almost-cliques* where probabilistic arguments fail to apply with high probability. We treat separately almost-cliques (called full in Chapter 4 and cabals in Part III) where the average node has fewer than $(c \log n)^{1+\kappa}$ external neighbors, where κ is a small constant, e.g., $\kappa = 1/30$.

To color those extremely dense almost-cliques, the idea is to *put aside* some of their vertices. Indeed, by keeping $(c \log n)^{1+\kappa}$ vertices uncolored, we provide the other vertices in the cabal enough slack to be colored ultrafast. Since vertices have so few outgoing edges, we can ensure that no edge connects put-aside sets from different almost-cliques (see Proposition 3.24). In particular, put-aside sets can be colored last with simple information gathering — at least in LOCAL.

3.2. The Almost-Clique Decomposition

In this section, we present a structural graph decomposition that classifies vertices according to their ability to get colored fast by random color trials. Loosely speaking, the sparsity (sometimes also called *local* sparsity) of a vertex measures the number of edges missing in its neighborhood. The higher the sparsity, the more efficient random color trials will be.

Definition 3.2. Vertex v is ζ-sparse if $G[N(v)]$ contains at most $\binom{\Delta}{2} - \zeta\Delta$ edges.

Let us begin with simple facts about the sparsity. First, observe that $G[N(v)]$ cannot contain many edges if v has a small degree. Quantitatively, Fact 3.3 follows from simple computation (see Section A.1).

Fact 3.3. *A non-isolated vertex with degree at most $\Delta - x$ is $(x/2)$-sparse.*

More interestingly, the local sparsity of a high-degree vertex captures the number of anti-edges its induced neighborhood contains. Indeed, the existence of $x\Delta$ anti-edges in $G[N(v)]$ means that $G[N(v)]$ contains at most $\binom{\deg(v)}{2} - x\Delta \leqslant \binom{\Delta}{2} - x\Delta$ edges.

Fact 3.4. *If $G[N(v)]$ contains $x\Delta$ anti-edges, then v is x-sparse.*

As we shall see in future chapters, $\Omega(\Delta)$-sparse vertices get colored efficiently by (variations of) random color trials. On the other hand, vertices that are not $\Omega(\Delta)$-sparse are clustered into sets resembling cliques.

Definition 3.5. For $\varepsilon \in (0, 1)$. A set of vertices K is an ε-almost-clique if

(1) $|K| \leqslant (1 + \varepsilon)\Delta$, and
(2) $|N(v) \cap K| \geqslant (1 - \varepsilon)\Delta$ for all $v \in K$.

The parameter ε intuitively quantifies how close K is to an actual $(\Delta+1)$-clique. For the entirety of this thesis, ε will be a small constant independent of n and Δ. Observe that an almost-clique has (strong-)diameter at most two because every pair of vertices in K share at least one neighbor in K, for otherwise $2(1 - \varepsilon)\Delta \leqslant |K| \geqslant (1 + \varepsilon)\Delta$ which is impossible when $\varepsilon < 1/3$.

Let K be a set of vertices and $v \in K$. We call neighbors outside of K *external* and vertices in K to which v is not adjacent the *anti-neighbors* of v. As external- and anti-neighbors will play a key role, we introduce notations for them: $E(v) = N(v) \setminus K$, $A(v) = K \setminus N[v]$, $e(v) = |E(v)|$, and $a(v) = |A(v)|$. We respectively call $e(v)$ and $a(v)$ the external- and anti-degree of v. Note that, from the definition of ε-almost-clique, we have $e(v) \leqslant \varepsilon\Delta$ and $a(v) \leqslant 2\varepsilon\Delta$. It should be clear that for every almost-clique K and vertex $v \in K$, we have that,

$$\deg(v) + 1 = |K| + e(v) - a(v) \ . \tag{3.1}$$

The main result of this section is that a constant round LOCAL algorithm partitions vertices of any graph between $\Omega(\varepsilon^2\Delta)$-sparse vertices and ε-almost-cliques. The earliest occurence of such a decomposition comes from [Ree98]. It was later adapted by [HSS18; ACK19] for $(\Delta+1)$-coloring.

Proposition 3.6. *Suppose $\varepsilon \in (0, 11/61)$ and $\Delta \gg_\varepsilon 1$. For every graph G with maximum degree Δ, there exists a bipartition V_{sparse}, V_{dense} of its vertices for which*

(1) every $v \in V_{sparse}$ is $(\varepsilon^2/2^{12})\Delta$-sparse;
(2) V_{dense} is partitioned into ε-almost-cliques;
(3) every $v \in V_{dense}$ is $(\varepsilon/2^{12})e(v)$-sparse.

The decomposition is computed deterministically in $O(1)$ rounds of LOCAL.

Throughout this thesis, we informally refer to a decomposition V_{sparse}, V_{dense} verifying Parts 1 to 3 as to an *ε-almost-clique decomposition*. We often drop the ε and simply write almost-clique and almost-clique decomposition. Henceforth, we refer to vertices of V_{sparse} and V_{dense} as sparse and dense vertices respectively. Note that a vertex in V_{dense} can be $\Omega(\Delta)$-sparse. For each dense vertex v, we denote by K_v the almost-clique that contains v.

Remark 3.7. It is worth noting that in [HSS18; CLP20; HKMT21] the external degree only counts edges to external neighbors in other almost-cliques. For this definition, it is direct that the sparsity of a dense vertex is proportional to its external degree. It is not the case here, because the external degree includes sparse vertices. To obtain Part 3, our algorithm has one extra step compared to that of [HSS18]. Namely, every (sparse) vertex with too many edges to an almost-clique is added to that almost-clique. We remark that, as observed in [HNT21, Lemma 3], doing so is not needed at the cost of a slightly worse dependency in ε in Part 3. We provide details in Section A.1 for completeness.

For the purpose of this chapter, we extend Part 3 to also include the anti-degree.

Lemma 3.8. *Every $v \in V_{dense}$ is $(\varepsilon/2^{14})(e(v) + a(v))$-sparse.*

The proof is a relatively direct counting argument which we defer to Section A.1 because we will not need it in future chapters. An interesting corollary of Lemma 3.8 is that a dense vertex is at least as sparse as the "average vertex" of its almost-clique. To make this precise, let us introduce

the following notations for average external- and anti-degrees:

$$e(K) = \frac{1}{|K|}\sum_{v\in K} e(v) \quad \text{and} \quad a(K) = \frac{1}{|K|}\sum_{v\in K} a(v)\ .$$

Corollary 3.9. *Every $v \in V_{dense}$ is $(\varepsilon/2^{19})(e(K_v) + a(K_v))$-sparse.*

Proof. Let $K = K_v$ and $\zeta = e(K) + a(K)$. We may assume that $e(v) \leqslant \zeta/8$, $a(v) \leqslant \zeta/24$ since Lemma 3.8 imply the corollary if either of those inequalities are false.

If $e(K) \geqslant 2a(K)$, then $e(v) \leqslant \zeta/8 \leqslant e(K)/4$. Because Eq (3.1) holds for all vertices in K, it also holds on average and we get that $\Delta + 1 \geqslant |K| + e(K) - a(K) \geqslant |K| + e(K)/2$. Using Eq (3.1) again at v, we get $\deg(v) \leqslant |K| - 1 + e(v) \leqslant \Delta - (e(K)/2 - e(v)) \leqslant \Delta - e(K)/4$. And thus, by Fact 3.3, v is $e(K)/8$-sparse, which is $\zeta/16$-sparse.

If $e(K) \leqslant 2a(K)$, then $a(v) \leqslant \zeta/24 \leqslant a(K)/8$. Observe that K contains $a(K)|K|/2$ anti-edges, of which at most $a(v) \cdot 2\varepsilon\Delta$ may have an endpoint out of $N(v) \cap K$. So $G[N(v) \cap K]$ contains at least $a(K)|K|/2 - 2a(v)\varepsilon\Delta \geqslant a(K)\Delta/8$ anti-edges for $\varepsilon < 3/20$. So v is $a(K)/8$-sparse which is $\zeta/16$-sparse. ∎

3.2.1. Proof of Proposition 3.6. In this section, we use a small constant $\delta = \Theta(\varepsilon) \in (0, 1)$ that we set precisely in Algorithm 1. A dense vertex is one that has essentially the same neighborhood as almost all of its neighbors. To capture this, we call an edge friendly when its endpoints share many neighbors:

Definition 3.10. The edge $\{u, v\}$ is δ-friendly if $|N(v) \cap N(u)| \geqslant (1-\delta)\Delta$.

By extension, let us define for each v its δ-friendly neighbors $F_\delta(v)$ as the set of $u \in N(v)$ such that $\{u, v\}$ is δ-friendly. Observe that in LOCAL, vertices can compute $F_\delta(v)$ in one round (vertices send their list of neighbors). A vertex with many friends is called popular:

Definition 3.11. A vertex is δ-popular if $|F_\delta(v)| \geqslant (1 - \delta)\Delta$.

A vertex with few friends, i.e., which is *not* δ-popular, must be sparse.

Lemma 3.12. *For every $\delta \in (0,1)$, a vertex such that $|F_\delta(v)| \leqslant (1-\delta)\Delta$ is $(\delta^2/8)\Delta$-sparse.*

PROOF. We may assume that $\deg(v) \geqslant (1-\delta/2)\Delta$ since v is otherwise $(\delta/4)\Delta$-sparse (Fact 3.3). Then, each not δ-friendly neighbor is incident to at least $\deg(v) - (1-\delta)\Delta \geqslant (\delta/2)\Delta$ anti-edges in $G[N(v)]$. By the hand-shaking lemma over all $\deg(v) - |F_\delta(v)| \geqslant (\delta/2)\Delta$ not δ-friendly neighbors of v, we obtain that $G[N(v)]$ contains at least $(\delta^2/8)\Delta$ anti-edges (note the division by two as we count anti-edges twice). ∎

Conversely, popular vertices induce almost-cliques. In some sense, every almost-clique is defined in term of its densest vertex. The condition in Lemma 3.13 that every D_i contains a high-degree vertex in H ensures that almost-cliques are large enough. The lemma is more general than needed in LOCAL so we can use it in later chapters.

Lemma 3.13. *Let $\delta \in (0, 1/8)$ and $\Delta \geqslant 1/(1-8\delta)$. Suppose V_δ is a set of δ-popular vertices and E_δ be a set of edges that contains all δ-friendly and only 2δ-friendly edges. Let $D_1, \ldots, D_t$ be the connected components of $H = (V_\delta, E_\delta)$ that* contain at least one vertex with $(1-\delta)\Delta$ neighbors in H. *Then, every D_i is a 5δ-almost-clique.*

PROOF. Consider $D = D_i$. Let us begin by showing that D has diameter two in H. Consider two vertices $v, u \in D$ connected by a path of length 3 in H. Since all edges of H are at least 2δ-friendly, one can show that $|N_G(v) \cap N_G(u)| \geqslant (1-6\delta)\Delta$. Furthermore, since u and v are δ-popular, they each have at most $\delta\Delta$ neighbors that are not δ-friendly. That is neighbors in G that are not neighbors in H. In particular, we have that $|N_H(u) \cap N_H(v)| \geqslant |N_G(u) \cap N_G(v)| - 2\delta\Delta \geqslant (1-8\delta)\Delta$. For $\delta < 1/8$ and $\Delta \geqslant 1/(1-8\delta)$, we get that u and v have at least one shared neighbor in H; hence are two hops apart.

(Part 1.) To bound the size of D, we bound the anti-degree $a(v) = |D \setminus N[v]|$ of $v \in D$. Since D has diameter at most 2, every $u \in A(v)$ shares at least one 2δ-friendly neighbor with v, hence $|N_G(u) \cap N_G(v)| \geqslant (1-4\delta)\Delta$. So, the number of two-hops paths of the form vxu where x is any vertex in G and $u \in A(v)$ is at least $(1-4\delta)a(v)\Delta$. On the other hand, there can be only $2\delta\Delta^2$ such two-hops paths: the at most Δ many δ-friendly

neighbors of v give at most $\delta\Delta$ two-hops paths (as they would otherwise not be δ-friendly) and the at most $\delta\Delta$ not δ-friendly neighbors of v — since v is δ-popular — cannot give more than Δ two-hops paths each. Together, we obtain $a(v) \leqslant 2\delta\Delta/(1-4\delta) \leqslant 5\delta\Delta$ for $\delta < 3/20$. Finally, $|D| \leqslant |N(v)| + a(v) \leqslant (1+5\delta)\Delta$.

(Part 2.) By definition, D contains a vertex v with $(1-\delta)\Delta$ neighbors in D. A neighbor $u \in N(v) \cap D$ must be 2δ-friendly with v, so has at least $|N(v) \cap D| - 2\delta\Delta \geqslant (1-3\delta)\Delta$ neighbors in D. Since D has strong diameter at most two, every $w \in D$ not adjacent to v is adjacent to — thus 2δ-friendly with — some $u \in N(v) \cap D$ and hence has $|N(v) \cap N(u) \cap D| - 2\delta\Delta \geqslant (1-5\delta)\Delta$ neighbors in D. ∎

Lemmas 3.12 and 3.13 give Parts 1 and 2 in Proposition 3.6. To ensure Part 3, we may add a few $\Omega(\varepsilon^2\Delta)$-sparse vertices to some of the D_is described in Lemma 3.13.

ALGORITHM 1. Computing the almost-clique decomposition.

Input: a graph G and a parameter $\varepsilon \in (0, 11/61)$.
Output: V_{sparse}, V_{dense} as described by Proposition 3.6.

Let $\delta = \varepsilon/11$.

(1) Let V_δ be the δ-popular vertices and E_δ the δ-friendly edges.
(2) Compute the sets $D_1, \ldots, D_t$ described in Lemma 3.13.
(3) Define

$$K_i = D_i \cup \{v \in V_G \setminus D_i : |N(v) \cap D_i| \geqslant (1-6\delta)\Delta\} .$$

(4) Output $V_{dense} = K_1, \ldots, K_t$ and $V_{sparse} = V_G \setminus V_{dense}$.

PROOF OF PROPOSITION 3.6. Let δ, V_δ and E_δ be as described in Algorithm 1. Note that they fit the conditions of Lemma 3.13. Compute $H = (V_\delta, E_\delta)$ and, in two rounds, vertices determine if they belong to one of the connected components $D_1, \ldots, D_t$ described by Lemma 3.13 — since they have diameter at most two. Those are 5δ-almost-cliques. Note that for our choice of $\delta < 1/12$, the K_i are disjoint.

(Part 1.) Let $U = V_G \setminus (D_1 \cup \ldots \cup D_t)$ be the remaining vertices. We claim that $v \in U$ is *not* $(\delta/2)$-popular. Toward a contraduction, let us assume that v is $(\delta/2)$-popular. Then, every pair $u, w \in F_{\delta/2}(v)$ of adjacent neighbors have $|N(v) \cap N(u) \cap N(w)| \geqslant (1-\delta)\Delta$ shared neighbors; hence, are δ-friendly. Since v is $(\delta/2)$-popular, it follows that every $u \in F_{\delta/2}(v)$ is δ-popular because it has $|F_{\delta/2}(v)| - (\delta/2)\Delta \geqslant (1-\delta)\Delta$ neighbors in $F_{\delta/2}(v)$. As such, $F_{\delta/2}(v) \subseteq V_\delta$ and v has $(1-\delta/2)\Delta$ neighbors in H, which contradict $v \in U$. Lemma 3.12 implies that every $v \in U$ is $(\delta^2/32)\Delta$-sparse. In particular, every vertex in V_{sparse} belongs to U so is $(\varepsilon^2/2^{12})\Delta$-sparse.

(Part 2.) Consider a set K_i. We show it is an 11δ-almost-clique, which implies it is an ε-almost-clique. By construction, each $v \in K_i$ has at least $(1-6\delta)\Delta \geqslant (1-\varepsilon)\Delta$ neighbors in $D_i \subseteq K_i$. So it suffices to upper bound $|K_i \setminus D_i|$. Since every vertex in D_i has at most $5\delta\Delta$ neighbors outside of D_i, we have that,

$$|K_i \setminus D_i| \leqslant \frac{5\delta\Delta \cdot |D_i|}{(1-6\delta)\Delta} \leqslant \frac{5\delta(1+5\delta)}{1-6\delta}\Delta \leqslant 6\delta\Delta \ .$$

(using that $5(1+5\delta)/(1-6\delta) \leqslant 6$ for $\delta \leqslant 1/61$)

This part follows because $|K_i| \leqslant |D_i| + |K_i \setminus D_i| \leqslant (1+11\delta)\Delta$.

(Part 3.) We may assume that $v \in D_i$ for some i since we showed that every vertex in U is $(\varepsilon^2/2^{12})\Delta$-sparse, which means that $v \in K_i \setminus D_i$ is $(\varepsilon/2^{12})e(v)$-sparse since $e(v) \leqslant \varepsilon\Delta$. So consider some $v \in D_i$ and recall that $|N(v) \cap D_i| \geqslant (1-5\delta)\Delta$. We show that every external neighbor $u \in E(v)$ has at least $\delta\Delta$ anti-edges to $N(v) \cap K$. If $u \in D_j$ for $j \neq i$, since D_j is a 5δ-almost-clique, we have that $|N(v) \cap K_i \setminus N(u)| \geqslant (1-5\delta)\Delta - e(u) = (1-10\delta)\Delta \geqslant \delta\Delta$ for $\delta \leqslant 1/11$. Otherwise, if $u \in U \setminus K_i$, it has at most $(1-6\delta)\Delta$ edges to D_i, and so it must have $|N(v) \cap D_i \setminus N(u)| \geqslant (1-5\delta)\Delta - (1-6\delta)\Delta \geqslant \delta\Delta$ anti-neighbors in $N(v) \cap D_i$. Summing overall $u \in E(v)$, we count at least $\delta e(v)\Delta$ anti-edges between $E(v)$ and $N(v) \cap D_i$, which implies that v is $(\varepsilon/11)e(v)$-sparse. ∎

3.3. Slack Generation

In randomized distributed graph coloring, the *slack of a vertex* v refers to the difference between the number of colors available to v and its uncolored degree[1]. We explain in the next chapter how vertices with slack proportional to their degree (i.e., $\delta \deg(v)$ for a constant $\delta > 0$) can be colored in $O(\log^* n)$ rounds. In this section, we analyze a simple algorithm that generates slack — as initially vertices have only one unit of slack.

To create excess colors for v, the main technique is to color multiple (at least two) neighbors using only one color. We call such colors *repeated* or *redundant*. For each redundant color in $N(v)$, vertex v gains slack as it loses only one color while the uncolored degree decreases by two. Recall that $L_\varphi(v) = [\Delta + 1] \setminus \varphi(N(v))$ denotes palette of v, the set of colors that can be used to extend φ at v.

Fact 3.14. *For any (possibly partial) coloring φ such that $N(v)$ contains at least s redundant colors, we have that $|L_\varphi(v)| \geqslant \deg_\varphi(v) + s$.*

However, not all vertices have equal capabilities to receive slack this way: while a vertex included in no triangles may receive up to Δ slack (if all neighbors use the same color), a vertex in a $(\Delta+1)$-clique cannot receive any. The notion of sparsity introduced in Definition 3.2 captures the ability of v to receive slack. Let us introduce the following notation,

$$\zeta(v) = \frac{1}{\Delta}\left(\binom{\Delta}{2} - \#\text{edges in } G[N(v)]\right). \tag{3.2}$$

so that, according to Definition 3.2, a vertex v is ζ-sparse if $\zeta(v) \geqslant \zeta$. We shall see that the color trial described in Algorithm 2 generates $\Omega(\zeta(v))$ redundant colors in $N(v)$.

Algorithm 2. SlackGeneration

Input: a graph G, a parameter $p \in [8/\Delta, 1]$ and $r \in \mathbb{N}_{\geqslant 0}$.
Output: a partial coloring φ.

[1]Slack could be defined formally, like in, e.g., [HKNT22]. In this thesis, we make the choice to use it as an informal notion and, formally, refer to the source of vertices' slack instead.

Each vertex joins **A** (becomes active) independently w.p. p
Each active vertex does:

(1) Sample $\boldsymbol{\chi}(v)$ uniformly in $(r, \Delta + 1]$
(2) If $\boldsymbol{\chi}(v) \notin \boldsymbol{\chi}(N(v) \cap \mathbf{A})$, then $\varphi(v) = \boldsymbol{\chi}(v)$, otherwise $\varphi(v) = \perp$.

Before we state the slack generation result, a few remarks about Algorithm 2 are in order. First, we introduce an activation probability p so we can easily ensure in later parts of the algorithm that we did not color too many vertices. Second, the parameter r (when non-zero) ensures that we do not use colors $\{1, 2, \ldots, r\}$, which we will need in later parts of this thesis. Choosing a constant activation probability (independent of Δ) and $r = (1-\Theta(1))\Delta$ many reserved colors affect the slack generated by constant factors.

Proposition 3.15. *Suppose G is a graph of maximal degree $\Delta \geqslant 4$, $p \in [8/\Delta, 1]$ and $r \leqslant \Delta/2$. If φ is the coloring produced by Algorithm 2, then for every v with $\zeta(v) \gg 1/p^2$, we have that,*

$$|L_\varphi(v)| > \deg_\varphi(v) + 2^{-16} \cdot p^2 \zeta(v)$$

with probability at least $1 - \exp\big(-\Omega(p^2\zeta(v))\big)$.

We emphasize that the failure probability in Proposition 3.15 is exponentially small in the sparsity of the vertex we consider, rather than, say, the maximum degree. This will turn out to be an important technical detail as, in the densest regions of the graph, Algorithm 2 will not be able to produce slack with high-enough probability.

3.3.1. Proof of Proposition 3.15. For succinctness, in this section, we use $\zeta = \zeta(v)$. Let us assume without loss of generality that v has $\deg(v) \geqslant \Delta + 1 - \zeta$. Otherwise, notice that $|L_\varphi(v)| \geqslant \deg_\varphi(v) + \zeta$ for all colorings. It is easy to verify $G[N(v)]$ contains at least $\zeta/2$ anti-edges when v has degree $\deg(v) \geqslant \Delta + 1 - \zeta$. See Section A.1 for a short proof.

Fact 3.16. *If v has degree at least $\Delta - x \geqslant 1$, then $G[N(v)]$ contains at least $(\zeta(v) - x/2)\Delta$ anti-edges.*

As in Algorithm 2 vertices get colored only if activated, we need that the sparsity is (roughly) preserved in the graph induced by activated vertices. While this should be clear in expectation, Lemma 3.17 shows that it holds with sufficiently high probability. The following lemma and its proof are adapted from [HMP24, Lemma 8.3].

Lemma 3.17. *Let S be a set of vertices such that $G[S]$ contains at least $\overline{m}$ anti-edges. Suppose every vertex joins $\mathbf{A}$ independently with probability $p \in [8/|S|, 1]$. Then, $G[S \cap \mathbf{A}]$ contains at least $p^2\overline{m}/2$ anti-edges with probability at least $1 - \exp(-p\overline{m}/5|S|)$.*

Proof. For each anti-edge f in $G[S]$, let $\mathbf{X}_f$ indicate if both endpoints joined $\mathbf{A}$. We are interested in lower bounding the probability that $\mathbf{X} = \sum_f \mathbf{X}_f$ is small. As $\mu = \mathbb{E}[\mathbf{X}] = p^2\overline{m}$, it suffices to show concentration. It is obtained by an application of Janson's inequality (Lemma 2.5). As vertices are sampled independently, it should be clear that

$$C \stackrel{\text{def}}{=} \frac{1}{2} \sum_{f \cap f' \neq \emptyset} \mathbb{E}[\mathbf{X}_f \mathbf{X}_{f'}] \leqslant \overline{m}|S|p^3$$

As $C/\mu \leqslant p|S|$, we get the bound

$$\mathbb{P}[\mathbf{X} < \mu/2] \leqslant \exp\left(-\frac{\mu}{8 + 4C/\mu}\right) \leqslant \exp\left(-\frac{p^2\overline{m}}{8 + 4p|S|}\right)$$

and the lemma follows from the assumption $p|S| \geqslant 8$. ∎

We turn now to the bulk of this proof. In Lemma 3.18, we show that for a fixed set of activated vertices A, the number redundant colors to appear in $N(v) \cap A$ is proportional to its sparsity — the number of activated anti-edges $N(v)$ contains. Here again, it is relatively straightforward to obtain the argument on average, as each anti-edge becomes monochromatic and retains its color with probability $\approx 1/\Delta$. To obtain concentration, we follow the approach of [MR14; AA20; HKMT21] which uses Talagrand Inequality.

Lemma 3.18. *Let G be a graph of maximal degree $\Delta \geqslant 4$ and $r \in [0, \Delta/2]$. Let v be a vertex and A a set of vertices such that $G[N(v) \cap A]$ contains at least $\overline{m} \geqslant 2 \cdot 10^{12}\Delta$ anti-edges. Suppose every vertex in A independently samples a uniform color in $\boldsymbol{\chi}(v) \in (r, \Delta + 1]$. Denote by $\mathbf{X}$ the random variable counting the number of pairs $u, w \in N(v)$ to sample the same*

color, retain their color and are the only vertices in $N(v)$ to sample that color. Then

$$\mathbb{P}\big[\mathbf{X} \geqslant 2^{-14}\overline{m}/\Delta\big] \geqslant 1 - 8\exp\left(-\frac{\overline{m}/\Delta}{10^{12}}\right) .$$

Proof. Let $q = \Delta + 1 - r$ be the size of the sample space. For an anti-edge $f = \{u, w\}$ in $G[N(v) \cap A]$, let $\mathbf{X}_f$ indicate if $\boldsymbol{\chi}(u) = \boldsymbol{\chi}(w)$ and $\boldsymbol{\chi}(u), \boldsymbol{\chi}(w) \notin \boldsymbol{\chi}(N(v) \cup N(u) \cup N(w) \setminus \{u, w\})$. Call these events respectively $\mathcal{E}_f^1$ and $\mathcal{E}_f^2$ (i.e., $\mathbf{X}_f$ indicates $\mathcal{E}_f^1 \cap \mathcal{E}_f^2$). We claim that $\mathbb{E}[\mathbf{X}_f] \geqslant \Omega(1/\Delta)$, which then implies that $\mathbf{X} = \sum_f \mathbf{X}_f$ has expectation at least $\Omega(\overline{m}/\Delta)$. As colors are sampled independently and $u, w \in A$, it should be clear that $\mathbb{P}[\mathcal{E}_f^1] = 1/q$. To lowerbound the probability of $\mathcal{E}_f^2 \mid \mathcal{E}_f^1$, we fix the color sampled by endpoints of f and use that each vertex in $N(f) \stackrel{\text{def}}{=} N(v) \cup N(u) \cup N(w) \setminus \{u, v\}$ sampled its color independently and that $|N(f)| \leqslant 3\Delta \leqslant 6q$, we have that

$$\mathbb{P}[\mathcal{E}_f^2 \mid \mathcal{E}_f^1] = \sum_{\chi \in (r, \Delta+1]} \frac{1}{q}\left(1 - \frac{1}{q}\right)^{|N(f)|} \geqslant 2^{-2\frac{|N(f)|}{q}} \geqslant 2^{-12} .$$

(using that $q \geqslant 2$ and $1 - x \geqslant 2^{-2x}$ for all $x \in (0, 1/2)$)

And so

$$\mathbb{E}[\mathbf{X}] = \sum_f \mathbb{P}[\mathcal{E}_f^2 \mid \mathcal{E}_f^1] \cdot \mathbb{P}[\mathcal{E}_f^1] \geqslant 2^{-12} \cdot \overline{m}/q .$$

To show that $\mathbf{X}$ is concentrated, we show concentration of two auxiliary variables:

- $\mathbf{Y}$ is the number of colors sampled by at least two non-adjacent vertices in $N(v)$,
- $\mathbf{Z}$ is the number of colors sampled by at least two non-adjacent vertices in $N(v)$ and sampled by one of their neighbors or at least one other vertex in $N(v)$.

By definition, we have $\mathbf{X} = \mathbf{Y} - \mathbf{Z}$. Firstly, observe that $\mathbf{Y}$ and $\mathbf{Z}$ are functions of $\boldsymbol{\chi}(v), v \in A$, which are independent random variables. Then both $\mathbf{Y}$ and $\mathbf{Z}$ are 2-Lipschitz functions of $\boldsymbol{\chi}$ as changing one $\boldsymbol{\chi}(v)$ affects at most two colors. Finally, there are respectively 2- and 3-certifiable: to certify $\mathbf{Y}$, point at one monochromatic anti-edge for each color, and to certify $\mathbf{Z}$, point at one monochromatic anti-edge f plus one neighbor or

other vertex in $N(f)$ that also sampled that color. Hence, we can use Talagrand's Inequality (with $c = 2$ and $r = 3$) to show concentration for both variables.

First, we need to upperbound $\mathbb{E}[\mathbf{Y}]$. For each color, by union bound over anti-edges, the probability that at least anti-edge samples that color is at most $\overline{m}/q^2$. Thus, $\mathbb{E}[\mathbf{Y}] \leqslant \overline{m}/q \leqslant 2^{12}\,\mathbb{E}[\mathbf{X}]$ and we have that

$$\mathbb{E}[\mathbf{X}]/4 - 60\sqrt{3\,\mathbb{E}[\mathbf{Y}]} \geqslant \mathbb{E}[\mathbf{X}]/4 - 60\cdot 2^6\sqrt{3\,\mathbb{E}[\mathbf{X}]} \geqslant \mathbb{E}[\mathbf{X}]/8\ ,$$

when $\mathbb{E}[\mathbf{X}]$ is large enough, e.g., for $\overline{m}/q \geqslant 10^{12}$. Hence, Talagrand's inequality gives

$$\mathbb{P}[|\mathbf{Y} - \mathbb{E}[\mathbf{Y}]| \geqslant \mathbb{E}[\mathbf{X}]/4] \leqslant 4\exp\left(-\frac{(\mathbb{E}[\mathbf{X}]/8)^2}{100\,\mathbb{E}[\mathbf{Y}]}\right) \leqslant 4\exp\left(-\frac{\overline{m}/q}{10^{12}}\right)\ .$$

Clearly, $\mathbb{E}[\mathbf{Z}] \leqslant \mathbb{E}[\mathbf{Y}]$ so we obtain the same concentration for $\mathbf{Z}$. By union bound, w.p. $1 - 8e^{-\frac{\overline{m}/q}{10^{12}}}$, we have that

$$\begin{aligned}\mathbf{X} = \mathbf{Y} - \mathbf{Z} &\geqslant \mathbb{E}[\mathbf{Y}] - \mathbb{E}[\mathbf{Z}] - \mathbb{E}[\mathbf{X}]/2 \geqslant \mathbb{E}[\mathbf{X}]/2\\ &\geqslant 2^{-13}\cdot\overline{m}/q \geqslant 2^{-14}\cdot\overline{m}/\Delta\end{aligned}$$

since $q \leqslant 2\Delta$. ∎

Putting everything together, we obtain Proposition 3.15.

Proof of Proposition 3.15. As observed earlier, we can assume that $\deg(v) \geqslant \Delta + 1 - \zeta$ and hence obtain that $G[N(v)]$ contains at least $\overline{m} \stackrel{\text{def}}{=} \zeta\Delta/2$ anti-edges (Fact 3.16). As vertices get activated independently with probability $p \geqslant 8/\Delta$, by Lemma 3.17, w.p. $1 - e^{-\Omega(p\zeta)}$, the graph induced by activated neighbors $G[N(v) \cap A]$ contains at least $\overline{m}_A \geqslant p^2\overline{m}/2$ activated anti-edges. Finally, by Lemma 3.18 on the activated subgraph, w.p. $1 - e^{-\Omega(p^2\zeta)}$, we have that $N(v)$ contains $2^{-14}\overline{m}_A/\Delta \geqslant 2^{-16}\cdot p^2\zeta$ repeated colors. Fact 3.14 then gives the result. ∎

3.4. Ultrafast Coloring with Slack

In this section, we shall see that when vertices have slack linear in their uncolored degrees, they can be colored in $O(\log^* n)$ rounds. We state Proposition 3.19 in terms of arbitrary lists $\text{List}(v) \in 2^{\mathbb{N}}$ for vertices of an

uncolored graph G. Note that it can be used to extend a partial $(\Delta+1)$-coloring φ by setting $\text{List}(v) = [\Delta+1] \setminus \varphi(N(v))$ for every uncolored vertex (provided those lists are large enough).

Proposition 3.19. *Let $\delta > 0$ be a real number and G be an n-vertex graph in which every vertex has a list $\text{List}(v)$ of at least $\deg(v) + s(v)$ colors where $s(v) \geqslant \delta \deg(v)$. If there exists $\kappa \in [1/n, 1]$ and $c > 0$ (both globally known) such that $s(v) \geqslant (2c \ln n)^{1+\kappa}$ for every vertex, then there exists a $O(\log^* n + 1/\delta^2 \log 1/\delta + 1/\kappa)$-round randomized* LOCAL *algorithm called* SlackColor *to* List*-color G with high probability.*

In the literature, SlackColor is often referred to as the MultiColorTrial algorithm for its main subroutine is an eponym algorithm (Algorithm 3) in which vertices try exponentially increasing numbers of colors. Various versions of Proposition 3.19 have been used [SW10; HSS18; CLP20; HKNT22] and we follow here the formalism of [HKNT22].

While Proposition 3.19 does not directly apply to $(\Delta+1)$-coloring, because vertices initially have $s(v) = 1$, SlackColor is the backbone of the coloring algorithm. Indeed, the previous section showed how a simple random color trial provides $s(v) \geqslant \Omega(\Delta)$ slack to every sparse vertex with high probability (Proposition 3.15). The coloring can then be extended to all sparse vertices using Proposition 3.19. Coloring dense vertices requires additional techniques to decrease uncolored degrees — which are the subject of the next chapter — so Proposition 3.19 applies to those as well.

Finally, we emphasize that to ensure a high probability of success, Proposition 3.19 needs all vertices to have slack asymptotically slightly larger than $\log n$. It seems necessary to ensure that the very last vertices can be colored by SlackColor.

3.4.1. Proof of Proposition 3.19. We introduce the MultiColorTrial algorithm in which vertices can try multiple colors at the same time. We emphasize the number of colors they try must be chosen carefully for this algorithm to be effective as trying too few merely colors vertices with a constant probability, while trying too many hinders chances of getting colored. Lemma 3.20 shows that MultiColorTrial is effective when the

number of colors tried is chosen as half of the ratio between the number of available colors and uncolored neighbors.

ALGORITHM 3. MULTICOLORTRIAL

Input: a graph G, lists List, a partial proper coloring φ and $x \in \mathbb{N}$.
Output: a proper extension of φ.

Repeat 3 times: each uncolored vertex v does,

(1) Sample a set $\mathbf{S}(v)$ of x colors in $\mathrm{List}_\varphi(v)$ with replacement[a].
(2) If $\chi \in \mathbf{S}(v) \setminus \mathbf{S}(N(v))$, then extend φ with χ at v. Otherwise, vertex v remains uncolored.

[a] for colored nodes, set $\mathbf{S}(v) = \emptyset$

Lemma 3.20. *Let $G, \mathrm{List}, \varphi, x$ be as described in Algorithm 3. Let $S \subseteq V_G \setminus \mathrm{dom}\,\varphi$ be a set of uncolored vertices for which $2x \deg_\varphi(v) \leqslant \mathrm{List}_\varphi(v)$. Then Algorithm 3 fails to colors all vertices of S with probability at most $2^{-3|S|x}$.*

PROOF. Focus on some vertex $v \in S$. It remains uncolored if all x colors in $\mathbf{S}(v)$ hit $\mathbf{S}(N(v))$. Since only uncolored vertices sample colors, the number of colors tried by neighbors is at most $x \deg_\varphi(v) \leqslant |\mathrm{List}_\varphi(v)|/2$. Hence, each sample in $\mathbf{S}(v)$ hits $\mathbf{S}(N(v))$ with probability at most $1/2$, and since all samples are independent, they all hit with probability $1/2^x$. Observe that this argument relies only on the randomness of $\mathbf{S}(v)$, hence the probability that all vertices in S fail 3 times multiply to $2^{-3|S|x}$. ∎

Observe that Lemma 3.20 only applies once the uncolored degrees are less than half the number of available colors. In particular, it does not apply to the initial lists given by Proposition 3.19 because they are only guaranteed to be larger than $(1+\delta)\deg(v)$ for a (possibly very small) constant $\delta > 0$. To remedy this, it suffices to observe that trying $x = 1$ color decreases uncolored degrees by a constant factor with high probability.

Lemma 3.21. *Let G, List, δ be as in Proposition 3.19. Let φ be the coloring obtained after repeating Algorithm 3 with $x = 1$ for $t = \lceil 1/\delta^2 \ln(2/\delta) \rceil$ times. Then, w.h.p., every v has $\deg_\varphi(v) \leqslant s(v)/2$.*

PROOF. Consider vertex v and the i-th iteration of Steps (1) and (2) in Algorithm 3 (out of $3t$ iterations in total). Call $u_1, u_2, \dots, u_d$ the uncolored neighbors of v at the beginning of this iteration. Since colored vertices conserve their color, we may assume that $d > s(v)/2$. Let $\mathbf{X}_i$ indicate if u_i remains uncolored at this iteration. The $\mathbf{X}_i$ are not necessarily independent but are stochastically dominated: for any set $S \subseteq [d]$, we have that

$$\mathbb{E}\left[\prod_{i\in S} \mathbf{X}_i\right] \leqslant (1-\delta)^{|S|}$$

because whatever the coloring might be at this step each neighbor of u_i blocks (by using or sampling) at most one color while its lists contain $(1+\delta)\deg(u_i)$ many colors. So the Chernoff Bound with Bernoulli-domination, (Lemma 2.4 with $p = 1-\delta$) implies that the probability that the uncolored degree of v does not decrease by a $(1-\delta^2)$ factor is

$$\begin{aligned}\mathbb{P}\left[\sum_{i=1}^{d} \mathbf{X}_i > (1-\delta^2)d\right] &= \mathbb{P}\left[\sum_{i=1}^{d} \mathbf{X}_i > (1+\delta)(1-\delta)d\right] \\ &\leqslant \exp\left(-\frac{\delta^2(1-\delta)d}{2+\delta}\right) \leqslant n^{-\Theta(c)} ,\end{aligned}$$

where the last inequality uses that $d \geqslant s(v)/2 \gg \delta^{-2}\log n$. By union bound over every vertex and iterations, w.h.p., uncolored degrees of every vertex decreases by a $(1-\delta^2)$ factor at each iteration. So, after $t = \lceil 1/\delta^2 \ln(2/\delta)\rceil$ iterations, the uncolored degrees are

$$\deg_\varphi(v) \leqslant (1-\delta^2)^t \deg(v) \leqslant e^{-\delta^2 t}\deg(v) \leqslant \delta/2 \cdot \deg(v) \leqslant s(v)/2 . \qquad \blacksquare$$

After trying one color $O(1/\delta^2 \log(1/\delta))$ times, SLACKCOLOR repeatedly applies MULTICOLORTRIAL $O(\log^* n)$ times with increasing values of x until $x \geqslant c\log n$, at which point Lemma 3.20 ensures that the remaining vertices get colored with high probability. Algorithm 4 has three Phases: Phase (1) reduces the uncolored applies Lemma 3.21; in Phase (2), the number of colors tried increases exponentially each iteration; and Phase (3) increases the color more slowly. We have Phase (3) because, at the very end, we cannot ensure uncolored degree decrease as aggressively as in Phase (2).

ALGORITHM 4. SLACKCOLOR

Input: a graph G, lists $\mathrm{List}(v)$, s and κ globally known
Ouput: a proper List-coloring of G

Let $t = \lceil 1/\delta^2 \ln(2/\delta) \rceil$.
Let i^* be the largest integer such that $2 \uparrow\uparrow i^\star \leqslant s^{\kappa/(1+\kappa)}$.
(1) for $i = 1, 2, \ldots, t$, run MULTICOLORTRIAL with $x = 1$;
(2) for $i = 1, 2, \ldots, i^*$, run MULTICOLORTRIAL with $x = 2 \uparrow\uparrow (i-1)$;
(3) for $i = 1, 2, \ldots, 1/\kappa$, run MULTICOLORTRIAL with $x = s^{i\kappa/(1+\kappa)}$.

We state Lemma 3.22 in a form more general than needed here so we can apply it in more constrained settings if provided with alternative implementations for MULTICOLORTRIAL.

Lemma 3.22. *Suppose there exist an algorithm called* MULTICOLORTRIAL *that verifies Lemmas 3.20 and 3.21. Then, if* $G, \mathrm{List}, s, \delta$ *and* κ *are as in Proposition 3.19, there exists an algorithm that computes a total proper* List*-coloring of* G *with high probability by executing* MULTICOLORTRIAL $\log^* n + \lceil 1/\delta^2 \ln(2/\delta) \rceil + \lceil 1/\kappa \rceil$ *times.*

PROOF. Let $s = (2c \ln n)^{1+\kappa}$ (recall that $s(v) \geqslant s$ for every vertex). Note that the integer i^* described in Algorithm 4 verifies $i^* \leqslant \log^*(s^{\kappa/(1+\kappa)}) \leqslant \log^* n$. By Lemma 3.21, at the end of Phase (1), every vertex has $\deg_\varphi(v) \leqslant s(v)/2$ with high probability. We analyze Phases (2) and (3) of SLACKCOLOR separately. For each of them, let φ_0 be the coloring before the first iteration and, for every $i \geqslant 1$, let φ_i be the coloring after the i-th iteration. For succintness, we use notation $\deg_i(v) = \deg_{\varphi_i}(v)$ for the uncolored degrees with respect to coloring φ_i.

Phase (2). We show that for each $i \in \{0, 1, \ldots, i^*\}$ of this phase, Eq (3.3) holds with high probability:

$$\text{for every } v \in V_G, \quad \deg_i(v) \leqslant \frac{s(v)/2}{2 \uparrow\uparrow i} \,. \tag{3.3}$$

Before the first call to MULTICOLORTRIAL, when $i = 0$, Eq (3.3) holds as given at the end of Phase (1). Now, assume Eq (3.3) holds for i and let us upper bound the probability that it fails to hold at $i + 1$. Fix an

uncolored vertex v. Note that Eq (3.3) is the promise required by the $(i+1)$-th call to MultiColorTrial with $x = 2 \uparrow\uparrow i$ (recall that we always have $\mathrm{List}_i(v) \geqslant s(v)$). Let $u_1, \dots, u_{\deg_i(v)}$ be the uncolored neighbors of v after the i-th iteration. Let $\mathbf{X}_j$ indicate if u_j remains uncolored. The probabilistic assumption on MultiColorTrial means that variables $\mathbf{X}_1, \dots, \mathbf{X}_{\deg_i(v)}$ are $\mathrm{Ber}(p)$-dominated for $p = 2^{-3x} \leqslant \frac{1}{2 \cdot 2 \uparrow\uparrow (i+1)}$. When $i < i^*$, the Chernoff bound with $\mathrm{Ber}(p)$-domination (Lemma 2.4 with $\delta = 1$) shows that

$$\begin{aligned}\mathbb{P}\left[\deg_{i+1}(v) > \frac{s(v)/2}{2 \uparrow\uparrow (i+1)}\right] &\leqslant \exp\left(-\frac{s(v)/2}{3 \cdot 2 \uparrow\uparrow (i+1)}\right) \\ &\leqslant \exp\left(-s^{1/(1+\kappa)}/6\right) \leqslant n^{-c/6} .\end{aligned}$$

By union bound, Eq (3.3) holds for every $i < i^*$ with high probability. At the last iteration, the Chernoff bound argument shows that, w.h.p.,

$$\deg_{i^*+1}(v) \leqslant \max\left(\frac{s(v)/2}{2 \uparrow\uparrow (i^*+1)}, c \ln n\right) \leqslant \frac{s(v)/2}{s^{\kappa/(1+\kappa)}}$$

where the second inequality uses that $s(v) \geqslant s \geqslant (2c \ln n)^{1+\kappa}$.

Phase (3). In this phase, uncolored degrees decrease by a multiplicative factor $s^{\kappa/(1+\kappa)}$ at each iteration. Consequently, during the last iteration, we run MultiColorTrial with $x = s^{1/(1+\kappa)} \geqslant c \ln n$ and it colors every remaining uncolored vertex with high probability. We show that for all $i \in \{0, 1, \dots, \lceil 1/\kappa \rceil\}$, we have that $\deg_i(v) \leqslant s(v)/(2s^{(i+1)\kappa/(1+\kappa)})$. When $i = 0$, it is given by the last iteration of Phase (2). Let us now assume it holds for every vertex for i and show that it holds for $i+1$ with high probability. Observe that the induction hypothesis at i implies the promise of $(i+1)$-th call to MultiColorTrial with $x = s^{i\kappa/(1+\kappa)}$. Consider a vertex v. By union bound over k-sized subsets of $\deg_i(v) \leqslant s(v)$, the probability that $k = s(v)/(2s^{(i+2)\kappa/(1+\kappa)})$ neighbors remain uncolored is at most

$$\begin{aligned}\mathbb{P}[\deg_{i+1}(v) > k] &\leqslant \binom{s(v)}{k} 2^{-3kx} \leqslant \exp(k(\ln(s(v)) - 2x)) \\ &\leqslant \exp\left(-s^{1/(1+\kappa)}\right) \leqslant n^{-c} ,\end{aligned}$$

where the third inequality uses that $\ln(s(v)) \leqslant \ln(n)$ and the assumption that $s^{1/(1+\kappa)} > c \ln n$ for some large enough constant $c > 0$. As $\lceil 1/\kappa \rceil \leqslant n$, the induction holds for all $i \in [\lceil 1/\kappa \rceil]$ with high probability. ∎

3.5. Coloring Almost-Cliques

Coloring almost-cliques seems challenging as they contain many edges, thus a lot of contention between their nodes. And, indeed, trying uniform random colors from palettes, as algorithms in Sections 3.3 and 3.4 have done thus far, will need $\Omega(\log \Delta)$ rounds to color a $(\Delta+1)$-clique. Nonetheless, coloring a $(\Delta+1)$-clique takes only one round of LOCAL: learn all identifiers and give color i to the vertex with the i-th smallest identifier. Pushing this further, coloring *one* almost-clique in LOCAL is also easy as it has diameter 2. Things become challenging when we attempt to color many almost-cliques concurrently.

For an almost-clique K, we call *clique palette* the set of colors that are not yet used in K

$$L_\varphi(K) = [\Delta+1] \setminus \varphi(K) \ . \tag{3.4}$$

Using the clique palette as a proxy for the palette of dense vertices will be fundamental throughout this thesis. By doing so, we make two types of approximations: colors of external neighbors are retained in $L_\varphi(K)$ while available colors used by anti-neighbors are excluded from $L_\varphi(K)$. Roughly speaking, such approximations are affordable because both quantities are proportional to the sparsity of the vertex.

The first technical result on this section concerns the *synchronized color trial*, in which we distribute colors of $L_\varphi(K)$ randomly to the uncolored vertices of K so that they each receive a different color. This limits possibilities of color conflicts to external neighbors. Note that it could also be that $L_\varphi(K)$ does not contain as many colors as there are uncolored vertices, so some vertices remain uncolored for that reason as well. As such, the number of uncolored vertices will be proportional to the sum of the average external- and anti-degree in K. More formally, the guarantees of the synchronized color trial can be stated as follow:

Proposition 3.23. *Let V_{sparse}, V_{dense} be an ε-almost-clique decomposition of a graph G and φ a partial coloring such that at least $(3/4)|K_i|$ vertices are uncolored in every almost-clique K_i. There is an algorithm (Algorithm 5) that, w.h.p., extends φ such that every K_i contains at most $16e(K_i) + \min_{v\in K_i} a(v) + O(\log n)$ uncolored vertices.*

The second technical piece of this section introduces *put-aside sets*. In the densest almost-cliques, i.e., where $e(K) \leqslant \Theta(\log^{1+\kappa} n)$, slack generation does not provide sufficient slack for SlackColor. To remedy this, we keep a set of $\Theta(\log^{1+\kappa} n)$ uncolored vertices inactive to ensure every other vertex has this much slack (because a vertex always has as many colors as uncolored neighbors). The challenge is to ensure we can later color those put-aside sets efficiently. Since we need those only in the densest almost-clique, it is actually possible to choose them so that no edge connects put-aside sets from different almost-cliques. As such, they can be colored independently and in parallel at the very end.

Proposition 3.24. *Let $\alpha, \ell \geqslant 0$ be integers and $c > 0$ a large constant. Let V_{sparse}, V_{dense} be an ε-almost-clique decomposition of a graph G with maximum degree $\Delta \geqslant 300\alpha \cdot \ell^2 \cdot c \ln n$. There exists a constant round* LOCAL *algorithm that, w.p. $1 - n^{-\Theta(c)}$, computes for every K_i with $e(K_i) \leqslant \ell$ a sets $P_i \subseteq K_i$ of size $\alpha\ell$ such that no edge connects P_i to some P_j for $j \neq i$.*

The parameter α is used to fine-tune the size of the put-aside sets. We also observe that the lower bound on Δ can be improved to $(\log n)^{2+\kappa}$ by employing a more intricate algorithm, as detailed in [HKNT22, Appendix C].

3.5.1. Synchronized Color Trial. In this section, we prove Proposition 3.23. Algorithm 5 and Lemma 3.25 are stated in more general form than strictly needed in LOCAL so we can use them in future chapters. For instance, in LOCAL, the order σ_i on the vertices of K_i can simply be the one induced by the unique identifiers and the set of colors $\mathscr{C}_i$ will be $L_\varphi(K_i)$.

Algorithm 5. SynchronizedColorTrial

Input:

- a graph G with almost-cliques $K_1, \dots, K_k$, a partial coloring φ;
- a set of colors $\mathscr{C}_i \subseteq L_\varphi(K_i)$ for every $i \in [k]$;
- a set of vertices $S_i \subseteq K_i \setminus \operatorname{dom}\varphi$ for every $i \in [k]$ and
- orders[a] $\sigma_i : S_i \setminus \operatorname{dom}\varphi \xrightarrow{\sim} [|S_i|]$ so that $v \in S_i$ knows $\sigma_i(v)$;

Output: an extension of φ

Do in each almost-clique K_i:

(1) Sample $\boldsymbol{\pi}_i$ a uniform permutation of $[|S_i|]$
(2) Let $\boldsymbol{\chi}(v)$ be the $\boldsymbol{\pi}_i(\sigma_i(v))$-th color in $\mathscr{C}_i$ for every uncolored $v \in S_i$
(3) If $\boldsymbol{\chi}(v) \in L_\varphi(v) \setminus \boldsymbol{\chi}(N(v))$, then extend φ with $\boldsymbol{\chi}(v)$ at v. Otherwise, v remains uncolored.

[a] we see a bijection σ between S and $\{1, 2, \dots, |S|\}$ as the total order where x is the $\sigma(x)$-th smallest element

Lemma 3.25. *Let $c > 0$ and $\alpha \in (0, 1]$. Consider G, K_i, φ, σ_i and $\mathscr{C}_i$ as in Algorithm 5. If $\alpha|K_i| \leqslant |S_i| \leqslant |\mathscr{C}_i|$ for every $i \in [k]$, then Algorithm 5 computes ψ, an extension of φ, such that, with probability at least $1-n^{-c+2}$, the number of uncolored vertices in every S_i is*

$$|S_i \setminus \operatorname{dom} \psi| \leqslant 8\alpha^{-1}e(K_i) + 6c \log n \ .$$

Proof. Fix i. We prove that the claimed bounds hold for K_i with high probability, and it then holds for all almost-cliques by union bound. For succinctness, write $K = K_i$, $S = S_i$ and $\mathscr{C} = \mathscr{C}_i$. Observe that since $|S| \leqslant |\mathscr{C}|$, every vertex in S has one color to try. Let $\mathbf{X}_j$ indicate if the j-th vertex of S (according to σ_i) fails to get colored. Partition S into two sets of size roughly $|S|/2$ according to σ_i-indices $A = \{1, 2, \dots, \lfloor |S|/2 \rfloor\}$ and $B = [|S|] \setminus A$. Let $\mu = 4\alpha^{-1}e(K) + 3c \log n$. We claim that

$$\mathbb{P}\left[\sum_{j \in A} \mathbf{X}_i > \mu\right], \ \mathbb{P}\left[\sum_{j \in B} \mathbf{X}_i > \mu\right] \leqslant 1/n^c \ . \tag{3.5}$$

Eq (3.5) implies the result because, by union bound[2], w.p. $1 - 1/n^{c+1}$, the number of uncolored vertices in S is at most 2μ.

We show that for every $j \in A$, if v is the j-th vertex in S (according to σ_i), then

$$\mathbb{P}[\mathbf{X}_j = 1 \mid \mathbf{X}_1, \dots, \mathbf{X}_{j-1}] \leqslant \frac{e(v)}{|\mathscr{C}| - (j-1)} =: q_j \tag{3.6}$$

[2] We emphasize that the two bad events are not independent (because they depend on the same underlying permutation) however, as they are both low-probability events, we can both rule them out by union bound.

This holds because v remains uncolored only if $\boldsymbol{\chi}(v) \in \varphi(E(v)) \cup \boldsymbol{\chi}(E(v))$ since vertices from the same almost-clique receive *different* colors — all from the clique palette (recall that $\mathscr{C} \subseteq L_\varphi(K)$). For any conditioning on $\mathbf{X}_1, \dots, \mathbf{X}_{j-1}$, at least $|\mathscr{C}| - (j-1)$ colors in $\mathscr{C}$ remain unassigned, and they are each equally likely to be given to the j-th vertex. On the other hand, each external neighbor blocks at most one color. Eq (3.6) thus holds by union bound over external neighbors of v.

Similarly, for every $j = \lceil (|S|+1)/2 \rceil - 1 + j' \in B$, we have the same bound with an offset,

$$\mathbb{P}\left[\mathbf{X}_j = 1 \mid \mathbf{X}_{\lceil |S|/2 \rceil}, \dots, \mathbf{X}_{j-1}\right] \leqslant \frac{e(v)}{|\mathscr{C}| - (j'-1)} =: q_j \ .$$

Note that $1 \leqslant j' \leqslant |S|/2$ for every $j \in B$.

Using that $|\mathscr{C}| \geqslant |S|$, $|S|/2 \geqslant j$ (or j') and $|S| \geqslant \alpha|K|$, we bound probabilities in Eq (3.6) as

$$q_j \leqslant \frac{2\alpha^{-1} e(v)}{|K|} \quad \text{and} \quad \sum_{j \in A} q_j, \ \sum_{j \in B} q_j \leqslant 2\alpha^{-1} e(K) \ .$$

Since $\sum_{j \in A} q_j \leqslant \mu/2$, Chernoff with domination (with $\mathbf{Y}_i = \mathbf{X}_i$ and $\delta = 1$) implies Eq (3.5) because $1 - \exp(-\mu/3) \geqslant 1 - n^{-c}$, since $\mu \geqslant 3c \log n$. ∎

We can now prove Proposition 3.23 for the specific case of LOCAL.

Proof of Proposition 3.23. Let σ_i be the order on uncolored vertices of K_i induced by the identifiers. Since $v \in K_i$ can learn all identifiers of vertices in K_i in two rounds, it knows $\sigma_i(v)$. Then, let $\mathscr{C}_i$ be the clique-palette $L_\varphi(K_i)$ and let S_i be the first $|\mathscr{C}_i|$ uncolored vertices in K_i according to σ_i. By assumption φ is a partial coloring such that every almost-clique K_i has at most $|K_i|/4$ colored vertices. And so $|\mathscr{C}_i| = |S_i| \geqslant \Delta + 1 - |K_i|/4 \geqslant (3/4 - 2\varepsilon)|K_i| \geqslant |K_i|/2$ where we use that $\varepsilon \leqslant 1/8$ and $\Delta \geqslant |K_i|/(1+\varepsilon) \geqslant (1-2\varepsilon)|K_i|$. Every vertex of K_i can learn $L_\varphi(K_i)$ in two rounds of LOCAL because K_i has diameter two. So we can run Algorithm 5 in three rounds of LOCAL (two to broadcast $\boldsymbol{\pi}_i$ in K_i and one to try the color). And so, Lemma 3.25 (with $\alpha = 1/2$) implies that each S_i contains at most $8e(K_i) + 6c \log n$ uncolored vertices with high probability. We conclude the proof by bounding the number of uncolored vertices that do not have a color to try

in Algorithm 5, i.e., the number of vertices in $K_i \setminus S_i$. We claim that $|K_i \setminus \operatorname{dom}\varphi| - |L_\varphi(K_i)| \leqslant \min_{v \in K_i} a(v)$. Let $v \in K_i$ be the vertex with minimum anti-degree; observe that $|K| = 1 + |N(v) \cap K| + |A(v)| \leqslant \Delta + 1 + a(v)$. Now, the bound follows as

$$\begin{aligned} |K_i \setminus \operatorname{dom}\varphi| = |K| - |K \cap \operatorname{dom}\varphi| &\leqslant \Delta + 1 - |K \cap \operatorname{dom}\varphi| + a(v) \\ &\leqslant |L_\varphi(K_i)| + a(v) \ . \end{aligned}$$

Since $K_i \setminus S_i$ is a set of $\max(0, |K_i \setminus \operatorname{dom}\varphi| - |L_\varphi(K)|)$ vertices by definition, this concludes the proof. ∎

3.5.2. Computing Put-Aside Sets. Algorithm 6 is the technical heart of this section and selects (under the proper conditions) sets of vertices $P_i \subseteq S_i$ that are large enough and not connected to each other. Proposition 3.24 makes one call to Algorithm 6 with the right parameters to find put-aside sets. We state Lemma 3.26 this way for future use.

ALGORITHM 6. LOWDEGREESAMPLE

Input: $p \in (0, 1]$, a graph G and sets $S_1, \ldots, S_k \subseteq V_G$.
Output: sets $P_i \subseteq S_i$ for every $i \in [k]$.

Do in each S_i:

(1) Sample each v in $\mathbf{A}$ independently with probability p.
(2) Return $P_i = \{v \in S_i \cap \mathbf{A} : (N(v) \setminus S_i) \cap \mathbf{A} = \emptyset\}$.

Lemma 3.26. *Let $p \in (0, 1]$. Suppose that $|N(v) \cap \bigcup_{j \neq i} S_j| < 1/(10p)$ for every $v \in S_i$ and, for every $i \in [k]$, we have that $|S_i| \geqslant 8p^{-2}c \ln n$. Then, w.h.p., sets P_i returned by Algorithm 6 have the following properties:*

(1) each $P_i \subseteq S_i$ contains at least $|S_i|p/5$ vertices,
(2) no edge connects P_i and P_j when $i \neq j$.

PROOF. Since $P_i \subseteq S_i \cap \mathbf{A}$, it should be clear, from Step (2) in Algorithm 6 that Part 2 holds. So we focus on proving that Part 1 holds with high probability.

Fix $i \in [k]$. Let $\mathbf{X}_v$ indicate if $v \in \mathbf{A}$. Since each vertex joins $\mathbf{A}$ independently with probability p, the classic Chernoff Bound (Eq (2.2))

implies that

$$\mathbb{P}\left[\sum_{v\in S_i}\mathbf{X}_v < p|S_i|/2\right] \leqslant \exp(-p|S|/8) \leqslant n^{-c}$$

where the last inequality uses the assumption on the size of S_i. Let $S' \subseteq S_i$ be a set of at least $p|S_i|/2$ vertices; we henceforth condition activations in S_i so that $S_i \cap \mathbf{A} = S'$.

Let $\mathbf{Y}_v$ indicate if $(N(v) \setminus S_i) \cap \mathbf{A} = \emptyset$, so that $v \in P_i$ if $v \in S'$ and $\mathbf{Y}_v = 1$. Since only neighbors of $\bigcup_j S_j$ get activated (join $\mathbf{A}$), by our assumption on degrees, the probability that a vertex joins P_i is

$$\begin{aligned}\mathbb{P}[\mathbf{Y}_v = 1 \mid S_i \cap \mathbf{A} = S'] &= (1-p)^{|N(v)\cap\bigcup_{j\neq i} S_j|} \\ &\geqslant 1 - p|N(v) \cap \bigcup_{j\neq i} S_j| \geqslant 9/10\end{aligned}$$

(using that $(1+x)^y \geqslant 1+xy$ for $x \geqslant -1$ and $y \geqslant 1$)

So we expect P_i to have size $0.9|S'|$ by linearity of the expectation. Note that $\mathbf{Y}_v$ is a function of $\{\mathbf{X}_u, u \in N(v) \cap \bigcup_{j\neq i} S_j\}$ and that each $\mathbf{X}_u$ influences (at most) $k = 1/(10p)$ variables $\mathbf{Y}_v$. By the read-k bound (Lemma 2.6, with $n = |S'|$ and $\delta = 1/2$),

$$\mathbb{P}\left[\sum_{v\in S'}\mathbf{Y}_v < |S_i|p/5\right] \leqslant \exp\left(-\frac{p|S_i|}{4k}\right) \leqslant n^{-c}\ ,$$

where the last inequality uses the assumption on the size of $|S_i|$. By union bound, this holds for all i with high probability, which concludes the proof. ∎

Proof of Proposition 3.24. For every almost-clique K_i with $e(K_i) < \ell$, define S_i as the set of vertices with external degree at most 2ℓ. By Markov's inequality, at least half of the vertices in K_i belong to $|S_i|$. So S_i has size at least $\Delta/3$ since $|K_i| \geqslant (1-\varepsilon)\Delta$ for $\varepsilon < 1/3$. Let $p = 1/(20\ell)$. Assuming that $\Delta \geqslant 300\alpha\cdot\ell^2\cdot c\ln n$ for a sufficiently large constant c, the sets $|S_i|$ verify the size condition of Lemma 3.26 for our value of p (with a smaller constant $\Theta(c)$). So, w.p. $1-n^{-\Theta(c)}$, the sets $P_i \subseteq S_i$ returned by Algorithm 6 are not connected by any edges and have size $|S_i|p/5 \geqslant \Delta/(300\ell) \geqslant \alpha\ell$ by

assumption on Δ. In particular, we can return as put-aside sets for K_i any $\alpha\ell$-sized subset of P_i. ∎

3.6. The Full Algorithm

We now have all the necessary technical ingredients to prove the existence of a $O(\log^* n)$-round algorithm when Δ is large. Algorithm 7 lists all steps with the appropriate parameters. Observe that each step extends the coloring computed earlier. Call φ_i the coloring at the beginning of the i-th step of Algorithm 7; by extension, $L_i(v) = [\Delta + 1] \setminus \varphi_i(N(v))$ and $\deg_i(v) = \deg_{\varphi_i}(v)$.

ALGORITHM 7. $(\Delta+1)$-coloring in LOCAL

Input: a graph G with $\Delta \gg \log^{3.1} n$.
Output: a proper $(\Delta+1)$-coloring of G

Let $\varepsilon = 1/2^3$, $\kappa = 1/30$, $\alpha = 2^{37}$ and $\ell = (2c \ln n)^{1+\kappa}$.

(1) Compute the ε-almost-clique decomposition V_{sparse}, V_{dense}.
(2) SLACKGENERATION with $p = 1/8$ and $r = 0$.
(3) COMPUTEPUTASIDESETS of size $\alpha\ell$ in K_i where $e(K_i) < \ell$.
(4) SYNCHRONIZEDCOLORTRIAL in every K_i
(5) SLACKCOLOR on $H = G[V_G \setminus (\bigcup_i P_i \cup \operatorname{dom} \varphi_5)]$
(6) Color every put-aside sets P_i independently in parallel

PROOF OF THEOREM 3.1 WHEN $\Delta \gg \log^{3.1} n$. Consider the graph H induced by vertices uncolored after the synchronized color trial but not in a put-aside set. *If any put-aside vertex was colored earlier, we uncolor it.* We claim that, at the beginning of Step 5, w.h.p., every v in H has

$$|L_5(v)| \geqslant \deg_5(v, H) + \max\{2^{-49} \deg_5(v, H), \ell\} \ . \tag{3.7}$$

Assuming this holds, SLACKCOLOR extends the coloring w.h.p. to every vertex in H in $O(\log^* n)$ rounds by Proposition 3.19 (with $G = H$, List $= L_5$, $\delta = 2^{-49}$ and κ). The argument for Eq (3.7) is separate for sparse and dense vertices.

Sparse Vertices. (Step 2.) By Proposition 3.6-(1), vertices of V_{sparse} have $\zeta(v) \geqslant \varepsilon^2/2^{12}\Delta$. Hence, by Proposition 3.15, w.p. $1-\exp(-\Omega(p^2\zeta(v))) \geqslant 1-\exp(-\Omega(\Delta)) \geqslant 1-1/\text{poly}(n)$, every $v \in V_{sparse}$ has

$$|L_2(v)| \geqslant \deg_2(v) + 2^{-16}p^2\zeta(v) \geqslant \deg_2(v) + 2^{-40}\Delta$$

from Step 2. Extending the coloring maintains this inequality. Uncoloring put-aside sets vertices increases the uncolored degree in G, but not in H, so Eq (3.7) holds for every sparse vertex.

Dense Vertices. (Steps 2 to 4.) Recall that vertices get colored in SlackGeneration only if active, so w.p. at most $p = 1/8$. By Chernoff Bound, w.h.p., at most $2p|K_i| \leqslant |K_i|/4$ vertices get colored prior to Step 4 in every almost-clique. So, after SynchronizedColorTrial (Step 4), by Proposition 3.23, w.h.p., every almost-clique contains at most $\min_{v\in K_i} a(v) + 16e(K_i) + O(\log n)$ uncolored vertices. As such, the uncolored degree of every dense vertex after SynchronizedColorTrial is upper bounded as

$$\deg_5(v) \leqslant e(v) + a(v) + 16e(K_v) + O(\log n)\ . \tag{3.8}$$

Now, recall that, by Lemma 3.8 and Corollary 3.9, every dense vertex is $\zeta(v)$-sparse for

$$\zeta(v) \geqslant 2^{-22}\max\{e(v)+a(v), e(K_v)+a(K_v)\}\ .$$

Observe that Eq (3.8) implies that $\deg_5(v) \leqslant 18 \times 2^{22}\zeta(v) \leqslant 2^{27}\zeta(v)$.

If $\zeta(v) \geqslant 2^{22}\ell$, since $\zeta(v) \gg \log n$, after SlackGeneration (Step 2), w.h.p. by Proposition 3.15, we have that,

$$|L_2(v)| \geqslant \deg_2(v) + 2^{-16}p^2\zeta(v) \geqslant \deg_2(v) + 2^{-22}\zeta(v)\ . \tag{3.9}$$

By union bound, both Eq (3.8) and (3.9) hold with high probability at the beginning of Step 5. And so Eq (3.9) becomes $|L_2(v)| \geqslant \deg_2(v) + 2^{-49}\deg_2(v)$. And since we assumed that $2^{-22}\zeta(v) \geqslant \ell$, Eq (3.7) follows.

Conversely, if $\zeta(v) \leqslant 2^{22}\ell$, then $a(v) \leqslant 2^{22}\zeta(v) \leqslant 2^{42}\ell$. Vertex v has at least $|P_i| - a(v) \geqslant \ell$ put-aside neighbors, because we choose $|P_i| = \alpha\ell$ with $\alpha = 2^{43}$. Since put-aside vertices are not included in H (they remain uncolored), they provide sufficient slack to others because

$$|L_5(v)| \geqslant \deg_5(v) = \deg_5(v;H) + |N(v)\cap P_i| \geqslant \deg_5(v;H) + \ell\ .$$

Since $\ell \geqslant 2^{-22}\zeta(v) \geqslant 2^{-49}\deg_5(v)$, this implies Eq (3.7).

Coloring Put-Aside Sets. (Step 6.) After coloring H, the only vertices remaining to color are in put-aside sets. As put-aside sets are not connected to each other, we can color them independently. Put-aside vertices in P_i learn all edges in $G[P_i]$ and palettes $L_6(u)$ for every $u \in P_i$. This takes three rounds of LOCAL as put-aside sets are included in almost-cliques. Vertices then locally compute the first proper L_6-list-extension of φ in, say, the lexicographic ordering and this becomes the coloring of P_i. ∎

3.7. A Note on the Low-Degree Case

In this section, we briefly address the low-degree case, i.e., when $\Delta \leqslant O(\log^{3.1} n)$. When Δ is large, the *local* error probability is polynomially small in n which transfers to a polynomially small *global* error probability by union bound over vertices. When Δ becomes small, the local errors become too likely to be avoided completely with high probability in n. A technique pionnered by [Bec91] to deal with Lovász Local Lemma and introduced to the realm of distributed algorithm by [BEPS16] deals with that exact issue. It states that, if failures are local enough and negatively correlated from other distant-enough failures, then w.h.p. failed vertices are gathered in polylogarithmic-size connected components. This is usually refered to as *shattering*:

Proposition 3.27 (Shattering)**.** *Suppose $\mathscr{A}$ is a* LOCAL *algorithm that outputs a set $\mathbf{B}$ of vertices such that there exists a constant t for which the following holds: for every vertex v and set $S \subseteq V_G \setminus N^t[v]$, we have*

$$\mathbb{P}[v \in \mathbf{B} \mid S \subseteq \mathbf{B}] \leqslant \Delta^{-3t} .$$

Then, w.p. at least $1 - n^{-\Omega(c)}$, every connected component in $G[\mathbf{B}]$ contains at most $(c/t)\Delta^{2t}\log_\Delta n$ vertices.

This led to the *shattering framework* for randomized LOCAL algorithm: run a randomized algorithm with a local failure probability Δ^{-c}; then, extend the solution to the vertices $\mathbf{B}$ that failed using a deterministic algorithm. Provided the solution can be extended to $\mathbf{B}$, like it is the case for $(\Delta+1)$-coloring, Proposition 3.27 states that w.h.p., the deterministic algorithm runs on $\text{poly}(\log n)$-size graphs — the components of $G[\mathbf{B}]$ —

thus resulting in a speed-up. Using the current best deterministic LOCAL algorithm [GG24], this gives us Theorem 3.1.

Proof Sketch for Theorem 3.1 when $\Delta \leqslant O(\log^{3.1} n)$. Run Algorithm 7 except that we skip put-aside sets (Steps 3 and 6). Initially $\mathbf{B}$ is empty and as the algorithm goes on, we add vertices to it. Whenever we add a vertex to $\mathbf{B}$, we uncolor it and it remains inactive for the remainder of the execution. Importantly, a vertex that joined $\mathbf{B}$ does *not* count as an uncolored neighbor, in some sense, we remove it from the graph. Vertices can become bad (join $\mathbf{B}$) for the following reasons:

- if a sparse vertex receives less slack than expected in Step 2;
- every vertex in an almost-clique K join $\mathbf{B}$, if K contains one vertex that receives less slack than expected in Step 2 or K contains more uncolored vertices than expected at the end of Step 4;
- during Step 5, if the uncolored degree of v decreases less than expected after one of the calls to MultiColorTrial in Algorithm 4.

The failure of every single event can be detected in $O(1)$ rounds and occurs with probability at most $\exp(-\Delta^\delta)$ for some constant $\delta \in (0, 1]$. Following carefully the analysis of each step, one can verify that the probabilistic guarantee holds even after conditioning on $S \subseteq \mathbf{B}$ for any $S \subseteq V_G \setminus N^t[v]$ with $t = O(1)$. That part can get quite tricky and technical, which is why we omit it here. We refer interested readers to [CLP20; HKNT22] for details.

The shattering lemma (Proposition 3.27) thus implies that at the end of Algorithm 7, w.h.p., the uncolored vertices are gathered in connected components in $G[\mathbf{B}]$ of size at most $N = \Delta^{O(1)} \log n = (\log n)^{O(1)}$ each. We run the algorithm of [GG24] and complete the coloring in $\widetilde{O}(\log^{5/3} N) = \widetilde{O}(\log^{5/3} \log n)$. ∎

Interestingly, it appears that (at fixed Δ), shattering is crucial to the advantage that randomized LOCAL algorithms may hold over their deterministic counterparts. As shown by [CKP19], the gap between randomized and deterministic complexity for "local problems" is at most exponential. The reason being that by pretending that the graph is exponentially larger,

we improve the success probability of an algorithm exponentially — and a direct application of the probabilistic method thus yields existence of a deterministic algorithm.

THEOREM 3.28 ([CKP19]). *Let* Π *be an LCL*[3] *and* $\mathscr{G}_{n,\Delta}$ *be the set of* n*-vertices graphs with maximum degree* Δ*. Call* $\mathsf{Rand}_\Pi(n,\Delta)$ *the smallest round complexity of a randomized algorithm that solves* Π *on* $\mathscr{G}_{n,\Delta}$ with probability $1-1/n$ *and call* $\mathsf{Det}_\Pi(n,\Delta)$ *the smallest round complexity of a deterministic algorithm solving* Π *on graphs of* $\mathscr{G}_{n,\Delta}$*. Then,*

$$\mathsf{Det}_\Pi(n,\Delta) \leqslant \mathsf{Rand}_\Pi(2^{n^2},\Delta) \ .$$

For some problems, such as sinkless-orientation, an exponential gap between deterministic and randomized exists [BFHKLRSU16; GS17] while for some problems (e.g., for 3-coloring trees [Lin92; MR85]) the optimal round complexity is $\Theta(\log n)$ for both deterministic and randomized algorithms [BBEHMOS20].

[3]Informally speaking, an LCL is a problem where vertices are tasked with labelling vertices or edges such that they avoid local configuration given as part of Π. The problem Π is local in the sense that the forbidden local configurations can be detected by a $O(1)$-round LOCAL algorithm. See, e.g., [NS95] or [Roz24, Definition 2.29] for more details.

Part I

Reducing the Bandwidth

CHAPTER 4

Ultrafast Coloring in Broadcast Congest

Classic distributed algorithms for coloring [Lub86; Joh99] achieved complexity $O(\log n)$ in the CONGEST model. There has been exciting recent progress on sublogarithmic time algorithms [BEPS16; HSS18; CLP20; GGR21; HKMT21; GK21; HKNT22; HNT22], and the state of the art round complexity is $O(\log^3 \log n)$ rounds. This is also the best known in the more relaxed LOCAL model, which allows unbounded message sizes. However, unlike the earlier algorithm of [Joh99], these faster algorithms make some nodes send one different message to each of their neighbors. Thus, each node may send up to $\Theta(n \log n)$ bits in one round. The research question at the core of this chapter is to understand the extent to which one can compute a coloring fast if we constrain the set of outgoing messages. Specifically,

> *Can we compute a* $(\Delta+1)$*-coloring as fast as in the* CONGEST *model if, in each round, each node must transmit the same* $O(\log n)$*-bit message to all its neighbors?*

To the best of our knowledge, with this restriction, the best round complexity known in general graphs remains the classic $O(\log n)$ bound [Lub86; Joh99; BEPS16].

Our Results. We give a fast $\Delta + 1$-coloring algorithm in the *broadcast congest* model (or BCONGEST) where, per round, each node *broadcasts one* $O(\log n)$*-bit message* to all of its neighbors.

THEOREM 4.1. *Let G be any n-node graph with maximum degree at most Δ. There is a distributed $O(\log^3 \log n)$-round algorithm that $\Delta + 1$-colors G with high probability, where each node broadcasts one $O(\log n)$-bit*

message in each round. If $\Delta \geqslant \Omega(\log^3 n)$*, the algorithm runs in* $O(\log^* n)$ *rounds.*

As a side remark, we note that the $O(\log n)$ complexity was the best bound known for general graphs even in the much more relaxed *broadcast congested clique* model, in which each node can send a $O(\log n)$ bit message to all other nodes. To emphasize, in this model, the communication graph is a complete graph and every two nodes are neighbors. The coloring is still with respect to the input graph G. This model is also sometimes known as the *shared blackboard model* with simultaneous messages and the *distributed sketching* model [DKO14; AKO20; AKZ22]. Our $O(\log^3 \log n)$-round complexity improves nearly exponentially over existing algorithms in this model.[1]

Assumption on Δ. The main technical contribution is a $O(\log^* n)$-round algorithm for coloring graphs with $\Delta \geqslant \Omega(\log^3 n)$. For low-degree graphs, a $O(\log^3 \log n)$-round algorithm that works in BCONGEST is known [BEPS16; GK21]. In fact, by using [BEPS16; GK21], our algorithm has complexity $O(\log \log n) + \text{Det}(\text{poly} \log n)$, where $\text{Det}(N)$ is the round complexity of $(\deg+1)$-list-coloring in BCONGEST. So, we assume $\Delta \geqslant \Omega(\log^3 n)$ for the remainder of this chapter.

Organization of this Chapter. First, we review the main challenges toward a BCONGEST algorithm for $(\Delta+1)$-coloring and its key differences from work prior to this thesis [HN23; HKMT21; HNT22]. Sections 4.2 and 4.3 adapt respectively the almost-clique decomposition (Section 3.2) and the slack color algorithm (Section 3.4) to BCONGEST. Section 4.4 presents a routing trick based on partitioning almost-cliques into *random groups* that we employ repeatedly in this thesis. Section 4.6 introduces *colorful matchings*, which will be useful in later chapters as well. Section 4.5 presents two implementations of the synchronized color trial in this model. Finally, a proof of Theorem 4.1 with a complete description of the algorithm can be found in Section 4.8.

[1] If we increase the size of the message sent by each node in this BCC model from $O(\log n)$ to $O(\log^3 n)$ bits, then a celebrated work of Assadi, Khanna, and Chen [ACK19] provides a one round algorithm.

4.1. Overview

4.1.1. Review of the Challenges. To implement the algorithm of Chapter 3, one needs to compute the almost-clique decomposition. In CONGEST, computing the list of neighbors shared by endpoints of every edges is difficult. Assadi, Chen and Khanna [ACK19] showed that it sufficed to discriminate δ-friendly edges from those that are not 2δ-friendly (recall that $\{u, v\}$ is δ-friendly if $|N(u) \cap N(v)| \geqslant (1-\delta)\Delta$, Definition 3.10). However their sampling based approach requires $O(\log^2 n)$ bandwidth. The authors of [HKMT21; HNT22] devised a CONGEST algorithm for this task, however it requires to send individual responses.

> *Challenge 1: How can we discriminate between δ-friendly and not 2δ-friendly edges with $O(\log n)$-bit broadcast? The previous approaches [ACK19; HNT22] require either large messages or individual responses.*

A basic primitive in randomized coloring algorithms is a *random color trial*: each vertex selects a color from its *palette* (its set of available colors) uniformly at random and keeps the color if none of its neighbors picked the same. As expounded in Section 3.4, sufficient slack speeds up coloring dramatically: each vertex can try *multiple colors* in each round (recall MultiColorTrial, Algorithm 3), resulting in a $O(\log^* n)$-round coloring algorithm called SlackColor. As a color requires up to $O(\log n)$ bits to describe, trying more than a constant number of them is infeasible with $O(\log n)$ bandwidth. A solution by [HNT22] was to use pseudorandomness: say each v tries a set of colors X_v, then v broadcasts a hash function h_v which each neighbor u of v uses to reply $h_v(X_u)$. A color that collides under h_v with none of its neighbors is safe to adopt. However, this approach requires individual responses $h_v(X_u)$ from each neighbor u. Therefore it does not work with single-message broadcasts.

> *Challenge 2: How can we perform* MultiColorTrial *with $O(\log n)$-bit broadcasts? The previous approaches [SW10; HNT22] require either large messages or individual responses.*

Recall that the second key concept for fast coloring is to *synchronize* the colors tried within each almost-clique, in the following sense: the color suggested to each vertex should be random from the viewpoint of the vertices outside the almost-clique, but there should be no conflicts between vertices inside the almost-clique. The earlier version of synchronized color trial (SCT for short) involved gathering all the information of the almost-clique for centralized processing [HSS18; CLP20], requiring high bandwidth. A simpler form of SCT of [HKNT22] has a leader vertex permute its own palette and distribute the colors to the other vertices of the almost-clique. In Section 3.5.1, we presented the variant introduced in [FGHKN23] that requires vertices to permute the clique palette. This still requires different messages to be sent along the different edges from a leader, making it incompatible with BCONGEST.

> *Challenge 3: How can we synchronize color trials with $O(\log n)$-bit broadcasts? The previous approaches [HSS18; CLP20; HKNT22] require either centralization or a vertex sending up to $\Omega(\Delta)$ messages.*

Finally, SlackColor requires $\ell = \Omega(\log^{1+\Omega(1)} n)$ slack in order to fully color the graph with high probability. This is solved in Section 3.5.2 (based on [HKNT22]) by putting aside mutually non-adjacent sets of ℓ vertices in very dense cliques, to be colored at the very end. In LOCAL or CONGEST, one colors put-aside sets by gathering all their relevant information (list of uncolored neighbors and palette) and broadcasting the coloring from a leader vertex.

> *Challenge 4: How can we color the put-aside sets with $O(\log n)$-bit broadcasts? The previous approach [HKNT22] does not work as it requires full information gathering and dissemination.*

Observe that Challenges 1, 2 and 4 can easily be solved by increasing the bandwidth to a small $\text{poly}(\log n)$. On the other hand, Challenge 3 seems to require greater effort to implement with the broadcast constraint, even with $\text{poly}(\log n)$ bandwidth.

4.1.2. Our Algorithm. In this section, we give an overview of our solutions to each of the challenges described earlier.

Almost-Clique Decomposition. Inspired by [HNT22], we let vertices compute a $O(\log n)$-bit fingerprint of their neighborhood as follow: each vertex sampled a uniform integer in $\{1, 2, \ldots, O(\Delta/\varepsilon)\}$ and then collect the set of integers smaller than $O(\varepsilon^{-4} \log n)$ used by its neighbors. By counting the number of elements shared between their sets, u and v can determine if they are δ-friendly or not 2δ-friendly. Conveniently, those sets can be represented in one $O(\varepsilon^{-4} \log n)$-bitmap, resulting in an efficient BCONGEST algorithm.

Multi-Color Trial. A subset of a *known* universe can be sampled pseudo-randomly in BCONGEST [HN23]. The problem is that when MultiColorTrial is applied after SCT, each vertex has a different palette, which is unknown to its neighbors. We solve this by *reserving* a subset of the color space for use by MultiColorTrial. Namely, each dense vertex v reserves the subset $[x(v)] = \{1, 2, \ldots, x(v)\}$, where $x(v)$ is a function of v's almost-clique density. Both slack generation and the synchronized color trial within v's almost-clique are restricted to using colors outside $[x(v)]$. The key is then to show that: a) using the colors $[\Delta + 1] \setminus [x(v)]$ suffices for these steps, and b) enough colors in $[x(v)]$ remain unused (by neighbors of v) for MultiColorTrial to succeed.

Synchronized Color Trial. Like in Chapter 3, the synchronized color trial in almost-clique K uses the *clique palette*: the set of colors not used by vertices in K. We randomly permute this set, in a distributed manner, and assign each color to a single uncolored vertex of K. As explained in Section 3.5.1, this colors enough vertices in each almost-clique.

To learn the clique palette $L_\varphi(K)$ in an almost-clique K, we randomly assign vertices of K into groups such that: a) every vertex is adjacent to at least one vertex of each group, and b) each group is connected and has a low diameter. Each group is tasked with learning a part of the clique palette, which it teaches to the rest of the almost-clique K.

We also randomly assign vertices into groups to randomly permute K. The random assignment roughly positions each vertex within the output

permutation π. Each group, of much smaller size than K, then randomly permutes its members. The small size of each group, combined with relabeling its members with smaller IDs, makes the description of a permutation of its members fit within small bandwidth.

Colorful Matching. Crucially, to ensure the clique palette contains enough colors for the synchronized color trial, we must generate additional slack in every almost-clique. Using an idea from [ACK19], we compute a *colorful matching*: a matching of anti-edges whoose endpoints are colored the same. Akin to slack generation, this is done by trying random colors in $[\Delta + 1]$ except that we retain colors only if they are repeated in their almost-clique (and they do not conflict with a neighbor). After repeating this $O(1/\varepsilon)$ times, each almost-clique contains a colorful matching of $\Omega(a(K)/\varepsilon)$ anti-edges, which is enough for our needs.

Coloring Put-Aside Sets. The put-aside set P_K of an almost-clique K has no edges to the put-aside sets in other almost-cliques. As such, coloring P_K can be done purely within K. Our algorithm first reduces the size of each P_K to sublogarithmic. Then, it gathers information about what remains of each P_K. One randomized color trial reduces $|P_K|$ by a constant factor with probability $1 - e^{-\Theta(|P_K|)}$. We compress the equivalent of $O(\log \log n)$ iterations of this process into $O(1)$ rounds by sampling the colors of all iterations *in advance* and sending them all at once. To reach sublogarithmic size *with high probability*, we run $O(\log \log n)$ independent iterations in parallel. We avoid congestion issues by using few colors per iteration and by representing colors with few bits.

4.2. Almost-Clique Decomposition

Proposition 4.2. *Let $\varepsilon \in (0, 11/61)$ and G be a graph with maximum degree Δ. There exists a $O(\varepsilon^{-4})$-round* BCONGEST *algorithm to a decomposition V_{sparse}, V_{dense} for which*

(1) every $v \in V_{sparse}$ is $(\varepsilon^2/2^{14})\Delta$-sparse;
(2) V_{dense} is partitioned into ε-almost-cliques;
(3) every $v \in V_{dense}$ is $(\varepsilon/2^{14})e(v)$-sparse.

Most of the work is on determining which edges are δ-friendly (recall Definition 3.10). While this is trivial in LOCAL, discriminating between δ-friendly and not δ-friendly edges with $O(\log n)$-bit messages is hard. Fortunately, as [ACK19] observed, it suffices to compute those approximately. Namely, to discriminate between δ-friendly and not 2δ-friendly edges. To do this, vertices compute a "fingerprint" of their neighborhoods such that similarity of fingerprints indicates similarity of neighborhoods. The probabilistic statement is a standard balls-in-bins argument; the trick for the algorithm to be bandwidth efficient is to use as fingerprint the number of non-empty bins amongst the first $O(\log n)$ of them and representing this using a bitmap.

ALGORITHM 8. Algorithm detecting friendly edges.

Input: a graph G with n vertices and maximum degree Δ
Output: a set $E_\delta \subseteq E_G$

(1) Each vertex samples $\mathbf{X}(v) \in [\alpha\Delta]$ uniformly at random where $\alpha = \lceil 16/\delta \rceil$
(2) Let $F(v) = \mathbf{X}(N(v)) \cap [b]$ where $b = \Theta(\delta^{-4} \log n)$.
(3) Add in E_δ every edge $\{u, v\}$ such that $|F(u) \cap F(v)| \geqslant (1 - 0.75\delta)b/\alpha$.

Lemma 4.3. *Fix $\delta \in (0, 1)$ and suppose that $n \gg 1$. Given a graph G with maximum degree Δ, Algorithm 8 outputs a set E_δ of edges such that, w.h.p. in n, every $(\delta/2)$-friendly edge belongs to E_δ and every edge in E_δ is at least δ-friendly.*

PROOF. Consider some edge $\{u, v\}$ and let $f = |N(u) \cap N(v)|$. First, we count the contribution of shared neighbors to $|F(u) \cap F(v)|$. Define $\mathbf{Y}_i$ indicating if a vertex $w \in N(u) \cap N(v)$ has $\mathbf{X}(w) = i$. Let $\mathbf{Y} = \sum_{i=1}^{b} \mathbf{Y}_i$. Observe that $\mathbf{Y}$ is 1-Lipschitz (changing $\mathbf{X}(w)$ affects at most two variables $\mathbf{Y}_i$ by one) and 1-certifiable (each $\mathbf{Y}_i = 1$ is certified by one $w \in N(u) \cap N(v)$ with $\mathbf{X}(w) = i$). Using that $\mathbb{E}[\mathbf{Y}] \leqslant b$, Talagrand's inequality implies

concentration for n large enough as

$$\mathbb{P}\Big[|\,\mathbb{E}[\mathbf{Y}]-\mathbf{Y}|>\delta b/8\alpha\Big]\leqslant 4\exp\left(-\frac{(\delta b/8\alpha-30\sqrt{\mathbb{E}[\mathbf{Y}]})^2}{8b}\right)$$
$$\leqslant \exp\big(-\Omega(\delta^4 b)\big)\leqslant n^{-c}\,.$$

(recall $\alpha=\Theta(\delta^{-1})$ and $b=\Theta(\delta^{-4}\log n)$)

In particular, when $\{u,v\}$ is $(\delta/2)$-friendly, i.e., when $f\geqslant(1-\delta/2)\Delta$, Algorithm 8 selects the edge with high probability in n. Indeed, $\mathbf{Y}_i=0$ if and only if none of the f shared neighbors of u and v sampled i. Since the $\mathbf{X}$-variables are independent, we have that

$$\mathbb{E}[\mathbf{Y}]=b\left(1-\left(1-\frac{1}{\alpha\Delta}\right)^f\right)\geqslant\left(\frac{f}{\alpha\Delta}-\left(\frac{f}{\alpha\Delta}\right)^2\right)b$$

(using $1-x\leqslant e^{-x}$ for all x and $e^{-x}\leqslant 1-x+x^2$ for $x\leqslant 1$)

$$\geqslant\left(\frac{f}{\alpha\Delta}-\frac{\delta/8}{\alpha}\right)b\geqslant\frac{1-\delta/2-\delta/8}{\alpha}b\,.$$

(using $(1-\delta/2)\Delta\leqslant f\leqslant\Delta$ and $8/\delta\leqslant\alpha$)

And so, when $\{u,v\}$ is $(\delta/2)$-friendly, w.h.p., $F(u)\cap F(v)$ contains at least $\mathbf{Y}\geqslant\mathbb{E}[\mathbf{Y}]-\delta b/8\alpha\geqslant(1-0.75\delta)b/\alpha$; hence, Algorithm 8 selects the edge. By union bound, every $(\delta/2)$-friendly edge is selected w.h.p. in n.

Conversely, when the edge is not δ-friendly, i.e., when $f<(1-\delta)\Delta$, w.h.p., shared neighbors of u and v will contribute to $|F(u)\cap F(v)|$ at most

$$\mathbf{Y}\leqslant\mathbb{E}[\mathbf{Y}]+\frac{\delta/8}{\alpha}b=b\left(1-\left(1-\frac{1}{\alpha\Delta}\right)^f\right)+\frac{\delta/8}{\alpha}b$$
$$\leqslant\left(\frac{f}{\alpha\Delta}+\frac{\delta/8}{\alpha}\right)b<\frac{1-7\delta/8}{\alpha}b\,.$$

(where the second inequality uses that $(1-1/\alpha\Delta)^f\geqslant 1-f/\alpha\Delta$.)

It remains to bound the contribution of $N(v)\triangle N(u)=(N(v)\setminus N(u))\cup(N(u)\setminus N(v))$ to $|F(u)\cap F(v)|$. Observe that when $w\in N(v)\triangle N(u)$ samples the same $\mathbf{X}$-value as a vertex in $N(v)\cap N(u)$, it does not increase

$|F(u)\cap F(v)|$. So the only way for a $w \in N(v)\setminus N(u)$ to increase $|F(u)\cap F(v)|$ is when there also exists a vertex $w' \in N(u) \setminus N(v)$ such that $\mathbf{X}(w) = \mathbf{X}(w') = i$. Define $\mathbf{Z}_i$ to indicate if there exists $w \in N(v) \setminus N(u)$ and $w' \in N(u) \setminus N(v)$ such that $\mathbf{X}(w) = \mathbf{X}(w')$. Since every vertex picks its $\mathbf{X}(w)$ independently and $|N(v) \setminus N(u)|, |N(u) \setminus N(v)| \leqslant \Delta$, we have that

$$\mathbb{E}[\mathbf{Z}] \leqslant b\left(1 - \left(1 - \frac{1}{\alpha\Delta}\right)^{\Delta}\right)^2 \leqslant \left(\frac{1}{\alpha}\right)^2 b \leqslant \frac{\delta/16}{\alpha} b\ .$$

(using $\alpha \geqslant 16/\delta$)

Similarly to $\mathbf{Y}$, the variable $\mathbf{Z}$ is 1-Lipschitz and 2-certifiable (each $\mathbf{Z}_i = 1$ is certified by two vertices w, w' with $\mathbf{X}(w) = \mathbf{X}(w') = i$). So Talagrand's inequality applies and w.e.h.p. in $\delta^4 b$, we get that $\mathbf{Z} \leqslant \mathbb{E}[\mathbf{Z}] + \delta b/16\alpha \leqslant \delta b/8\alpha$. Putting everything together, when $\{u, v\}$ is not δ-friendly, w.e.h.p. in $\delta^4 b$, we have

$$|F(u) \cap F(v)| \leqslant \mathbf{Y} + \mathbf{Z} < \frac{1 - 0.75\delta}{\alpha} b$$

and Algorithm 8 does not select the edge. By our choice of $b = \Theta(\delta^{-4} \log n)$, this holds for the edge $\{u, v\}$ w.h.p. in n. By union bound over edges, this holds for every edge w.h.p. in n. ∎

Remark 4.4. The use of Talagrand's inequality forces n to be greater than a very large constant. It can be avoided with a more ingenous analysis; e.g., see [HNT22, Claim 1] for different a proof with better constants.

Proof of Proposition 4.2. We run Algorithm 1 as in Section 3.2 with $\delta = \varepsilon/11$ except that E_δ is the outcome of Algorithm 8 and V_δ as the set of vertices with at least $(1 - \delta)\Delta$ incident edges in E_δ. By Lemma 4.3, w.h.p., every edge in E_δ is δ-friendly, so every vertex in V_δ is δ-popular. Let $D_1, \ldots, D_t$ be the connected components of $H = (V_\delta, E_\delta)$ that contain one vertex with at least $(1 - \delta)\Delta$ neighbors in H. Lemma 3.13 applies and sets D_i are 5δ-almost-cliques. Then, we let K_i be D_i plus all the vertices with at least $(1-6\delta)\Delta$ neighbors in D_i. Proof for Part 2 and Part 3 are the same as in Section 3.2. So we only explain why vertices of $U = V_G \setminus (D_1 \cup \ldots \cup D_t)$ are $(\varepsilon^2/2^{14})$-sparse.

A vertex $v \in U$ is not $(\delta/4)$-popular because every pair $u, w \in F_{\delta/4}(v)$ of adjacent neighbors are $(\delta/2)$-friendly, thus every $u \in F_{\delta/4}(v)$ is $(\delta/2)$-popular. Since Algorithm 8 selects every $(\delta/2)$-friendly edge w.h.p., every $(\delta/2)$-popular vertex joins V_δ; hence v and every vertex in $F_{\delta/4}(v)$ are in V_δ. As such, v would have $|F_{\delta/4}(v)| \geqslant (1-\delta/4)\Delta$ neighbors in H, which contradict $v \in U$. So Lemma 3.12 implies that every $v \in V_{sparse}$ is $(\delta^2/128)$-sparse, hence $(\varepsilon^2/2^{14})$-sparse.

To run Algorithm 8 in BCONGEST, vertices broadcast $\mathbf{X}(v)$ in one round and then describe the set $F(v)$ using a b-bit bitmap. So it runs in $O(\delta^{-4}) = O(\varepsilon^{-4})$ rounds of BCONGEST. Then the algorithm performs two rounds of BFS to identify the component $D_1, \dots, D_t$. Each vertex computes its degree in D_i in one round, hence the algorithm takes $O(\delta^{-4})$ rounds overall. ∎

4.3. Slack Color with Public Randomness

As explained in Section 4.1, the MultiColorTrial algorithm (Algorithm 3) uses up to $O(\log^2 n)$ bandwidth. In [HNT22], the authors propose an alternative which uses $O(\log n)$ bandwidth but requires sending individual messages; hence does not apply to BCONGEST. By requiring that vertices known which set of colors their neighbors are sampling from (Part i)), we can use a trick of [New91; HN23] to implement MultiColorTrial.

Proposition 4.5. *Let $\delta \in (0,1)$ be a real number and G be an n-vertex graph in which every vertex has a list $\mathrm{List}(v)$ and φ a partial coloring of G such that*

i) $\mathrm{List}(v)$ is known to every vertex in $N[v]$,
ii) $|\mathrm{List}_\varphi(v)| \geqslant \deg_\varphi(v) + s(v)$ colors where $s(v) \geqslant \delta|\mathrm{List}(v)|$.

If there exists $\kappa \in [1/n, 1]$ and $c > 0$ (both globally knwon) such that $s(v) \geqslant (2c\log n)^{1+\kappa}$ for every vertex, then there exists a $O(1/\delta^2(\log^* n + \log 1/\delta + 1/\kappa))$-round randomized BCONGEST algorithm called SlackColor to List-list-colors G with high probability.

The main idea is to try colors from a $O(\log n)$-sized pseudorandom subset of $\mathrm{List}(v)$. Part i) allows to describe the pseudorandom sets using

$O(\log n)$ bits. The authors of [HN23] formalizes this with the notion of representative sets.

Definition 4.6 (Representative Sets [HN23])**.** Let $\mathscr{U}$ be a universe of size k. A familly $\mathscr{F} = \{S_1, \dots, S_t\}$ of s-sized set is a (α, δ, ν)-representative familly if and only if

$$\forall T \subseteq \mathscr{U}, |T| \geqslant \delta k : \qquad \mathbb{P}_{i\in[t]}\left[\left|\frac{|S_i \cap T|}{|S_i|} - \frac{|T|}{k}\right| \leqslant \alpha\frac{|T|}{k}\right] \geqslant 1 - \nu \ . \tag{4.1}$$

The definition of representative sets in [HN23] is slightly stronger than ours. We focus here on Eq (4.1) because it suffices for our use. A classic application of the probabilistic method with Chernoff Bounds proves the existence of small families of representative sets:

Proposition 4.7 ([HN23, Lemma 3.2])**.** *Let $\mathscr{U}$ be a universe of size k. For any $\alpha, \delta, \nu > 0$, there exists an (α, δ, ν)-representative family of $t = \Theta(k/\nu + k\log(k))$ subsets, each of size $s = \Theta(\alpha^{-2}\delta^{-1}\log(1/\nu))$.*

In [HN23], the authors used this to solve coloring problems in which every vertex has $\Omega(\Delta)$ slack — so essentially the case where every vertex is sparse. We adapt their idea in Algorithm 9 to handle vertices of various sparsity.

ALGORITHM 9. MULTICOLORTRIAL in BCONGEST.

Input: a graph G, lists List, a partial coloring φ and $x \in \mathbb{N}$.
Output: an extension of φ.

Repeat $T = \lceil 12/\delta \rceil$ times: for every uncolored vertex v,

(1) Let $\mathscr{F}(v) = \{S_1(v), \dots, S_t(v)\}$ be an $(1/2, \delta/2, n^{-c})$-representative familly over the universe $\mathscr{U} = \text{List}(v)$ where each $S_i(v)$ has size $O(\delta^{-1}\log n)$.
(2) Let $\mathbf{S}(v)$ be a uniform random set from $\mathscr{F}(v)$.
(3) Sample a set $\mathbf{S}'(v)$ of x colors in $\text{List}_\varphi(v) \cap \mathbf{S}(v)$ with replacement.
(4) If $\chi \in \mathbf{S}'(v) \setminus \mathbf{S}'(N(v))$, then extend φ with χ at v.
Otherwise, vertex v remains uncolored.

Lemma 4.8. *Let $G, \mathrm{List}, \varphi, x$ be as described in Algorithm 9 and such that Part ii) holds. There is an event $\mathcal{E}$ of probability at least $1-\delta^{-1}n^{-\Theta(c)}$ such that when conditionned on $\mathcal{E}$ the following holds. For any set of uncolored vertices $S \subseteq V_G \setminus \operatorname{dom}\varphi$ for which $2x\deg_\varphi(v) \leqslant |\mathrm{List}_\varphi(v)|$, the probability that all vertices of S remain uncolored after Algorithm 9 is at most $2^{-3|S|x}$.*

Proof. Consider some fixed iteration of Step 1 to Step 4 at the beginning of which $v \in S$ is uncolored. Since Part ii) holds initially and remains true as we extend the coloring, it holds during this iteration. In particular, as we extend φ, we maintain $|\mathrm{List}_\varphi(v)| \geqslant s(v) \geqslant \delta|\mathrm{List}(v)|$. Fix the colors $B = \mathbf{S}'(N(v))$ tried by neighbors arbitrarily. They represent at most $|B| = x\deg_\varphi(v) \leqslant |\mathrm{List}_\varphi(v)|/2$ colors by assumption. As such, Eq (4.1) with $T = \mathrm{List}_\varphi(v) \setminus B$ implies — note that $|\mathrm{List}_\varphi(v) \setminus B| \geqslant |\mathrm{List}_\varphi(v)|/2 \geqslant \delta|\mathrm{List}(v)|/2$ so Eq (4.1) applies — that w.h.p. over the choice of $\mathbf{S}(v) \in \mathscr{F}(v)$, we have

$$\begin{aligned}\frac{|\mathbf{S}(v) \cap (\mathrm{List}_\varphi(v) \setminus B)|}{|\mathbf{S}(v)|} &\geqslant (1/2)\frac{|\mathrm{List}_\varphi(v) \setminus B|}{|\mathrm{List}(v)|} \\ &\geqslant (1/4)\frac{|\mathrm{List}_\varphi(v)|}{|\mathrm{List}(v)|} \geqslant \delta/4\end{aligned} \tag{4.2}$$

where the last inequality uses Part ii). Call $\mathcal{E}_i$ the event that Eq (4.2) holds for every $v \in S$. Assuming $\mathcal{E}_i$, the probability that none of the colors in $\mathbf{S}(v) \cap (\mathrm{List}_\varphi(v) \setminus B)$ gets sampled in $\mathbf{S}'(v)$ is at most $(1-\delta/4)^x \leqslant 2^{-\delta x/4}$. Note that after conditioning on $\mathcal{E}_i$, this upper bound depends only on the randomness of $\mathbf{S}'(v)$. So, assuming $\mathcal{E}_i$ and that every vertex of S is uncolored the beginning of the i-th iteration, the probability that Algorithm 9 fails to color every vertex in S during this iterations is at most $2^{-\delta|S|x/4}$.

Let $\mathcal{E} = \bigcap_{i\in[T]} \mathcal{E}_i$ be the event from the statement. By union bound, the event $\mathcal{E}$ holds w.h.p. in n. Since each iteration runs independently from the previous ones, probabilities multiply and the probability that none of the vertices in S gets colored in any of the iterations is at most $2^{-\delta T|S|x/4} \leqslant 2^{-3|S|x}$ as claimed. ∎

The proof of Lemma 4.8 follows from running SlackColor (Algorithm 4) and applying the same inductive analysis as in Lemma 3.22.

PROOF OF PROPOSITION 4.5. In BCONGEST, vertices can compute the set $\text{List}_\varphi(v)$ in one round (e.g., each colored vertex broadcasts its color). So Algorithm 3 with $x = 1$, in which vertices try one color in $\text{List}_\varphi(v)$, is a BCONGEST algorithm and we run it $O(1/\delta^2 \log(1/\delta))$ times with that same round complexity. By Lemma 3.21, w.h.p., every vertex in has $\deg_\varphi(v) \leqslant s(v)/2$.

Observe that each iteartion of Step 1 to Step 4 in Algorithm 9 takes $O(1/\delta)$ rounds to implement. Indeed, Steps 1 to 3 take $O(1)$ rounds since v needs only to know $\text{List}_\varphi(v)$. Step 4 takes $O(1/\delta)$ rounds to implement because v needs to know $\mathbf{S}'(u)$ for every neighbor. The set can be represented in $O(\log|\mathscr{F}(u)| + s)$ bits because, by assumption i), v knows $\text{List}(u)$ and the parameters for representative sets (i.e., δ, c and n) are globally known. So the set $\mathbf{S}'(u)$ can be described by u by the index of $\mathbf{S}(u)$ in $|\mathscr{F}(u)|$ and an s-bitmap which u broadcasts. Overall, Algorithm 9 uses $T \cdot O(1/\delta) = O(1/\delta^2)$ rounds of BCONGEST.

Note that the argument of Lemma 3.22 focuses on a vertex v and then consists of $O(\log^* n)$ uses of Lemma 4.8. By union bound, the high probability event $\mathcal{E}$ holds for each of the uses of Lemma 4.8 with high probability in n. And since $s, \kappa, \deg_\varphi(v) \leqslant s(v)/2$ and Algorithm 9 (with the conditioning on $\mathcal{E}$) verifies the condition of Lemma 3.22, vertices get List-list colored in $O(1/\delta^2 \log^* n + 1/(\delta^2\kappa))$ rounds of BCONGEST with high probability. ∎

4.4. Random Groups & Many-to-All Broadcast

The subsequent sections focuse on almost-cliques. We introduce here a simple albeit key trick for routing messages in almost-cliques. Using that almost-cliques are very densely connected, we observe that random groups of vertices preserve good connectivity with high probability.

Lemma 4.9 (Random Groups)**.** *Let K be an almost-clique and an integer $k \leqslant \Delta/(c \ln n)$ for some large enough $c > 0$. Suppose each $v \in K$ samples $\boldsymbol{t}(v) \in [k]$ uniformly at random. Then, w.p. $1 - n^{-\Theta(c)}$, for each $i \in [k]$, the set $T_i = \{v \in K : \boldsymbol{t}(v) = i\}$ satisfies that for any $u, w \in K$, $|T_i \cap N(u) \cap N(w)| \geqslant (c/4) \log n$.*

We say that T_i *2-hop connects* K in that each pair of nodes in K has a common neighbor in T_i. Note that since $T_i \subseteq K$, each T_i also 2-hop connects itself, thus has diameter 2. The proof is a direct application of the Chernoff Bound.

PROOF. Fix an index $i \in [k]$. Each node joins T_i w.p. $1/k$ independently from other nodes. For each pair $u, w \in K$, in expectation, $T_i \cap N(u) \cap N(w)$ has size $\mu = |N(u) \cap N(v)|/k \geqslant (1-2\varepsilon)\Delta/k \geqslant \Delta/(2k)$. The classic Chernoff bound (Eq (2.2)), implies that $\mathbb{P}[|T_i \cap N(u) \cap N(w)| \leqslant \mu/2] \leqslant \exp(-\mu/8) \leqslant n^{-c/16}$. By union bound, w.h.p., we have $|T_i \cap N(u) \cap N(w)| \geqslant \Delta/(4k)$ for all $i \in [k]$ and $u, w \in K$. ∎

As first application of Lemma 4.9, we introduce the Many-to-All Broadcast. It allows us to assume we have global communication within almost-clique if the number of messages to send is small enough.

Corollary 4.10 (Many-to-All Broadcast)**.** *Let K be an almost-clique with $O(\Delta/\log n)$ nodes with an $O(\log n)$-bit message to send to everyone in K. Suppose each node with a message broadcasts it, before each node in K broadcasts one messages it received, picked randomly. Then, w.h.p., all messages are received by every node in K.*

PROOF. Let T_i be the set of vertices which choose to repeat the i-th message. By Lemma 4.9, w.h.p., each vertex of K has at least one neighbor in every T_i, thus receives every message. ∎

4.5. Synchronized Color Trial

At its core, synchronized color trial (Algorithm 5) is about creating a random bijection between a set of colors and (most of) the uncolored nodes of an almost-clique. Our implementation uses the clique palette $L_\varphi(K) = [\Delta + 1] \setminus \varphi(K)$ as the set of colors and randomly permutes the nodes. The order of each node in the permutation tells it which color to take in the clique palette. This entails two difficulties. Firstly, to make use of its order in the sampled permutation, each node needs to know the matching color in the clique palette. We show that $O(1)$ rounds of BCONGEST suffice for all nodes to learn their clique palette. The second

issue is sampling the permutation, and entails a more involved process. For simplicity, we describe first a $O(\log\log n)$-round permutation sampling procedure, which suffices for Theorem 4.1. We then explain how to reduce it down to $O(1)$ rounds with a slightly more involved procedure.

4.5.1. Learning the clique palette. We learn the clique palette by dividing the color space into $O(\Delta/\log n)$ contiguous subpalettes. Given a 2-hop connecting set of nodes to handle each subpalette — with a trivial construction due to Lemma 4.9 — each node learns $L_\varphi(K)$ in $O(1)$ rounds. Recall that $\varphi(S)$ denotes the set of colors currently assigned to a set S of nodes.

ALGORITHM 10. Procedure LEARNPALETTE, in almost-clique K.
Parameters: Let $c = O(1)$ be a large enough constant, $k = \lfloor \Delta/(c\log n)\rfloor$.

(1) Each vertex in K samples $\boldsymbol{t}(v) \in [k]$ and let $T_1, \dots, T_k$ be the induced 2-hop connecting groups. Let $R_i := \{1 + \lfloor (i-1)\cdot(\Delta+1)/k\rfloor, \dots, \lfloor i\cdot(\Delta+1)/k\rfloor\}$, i.e., $R_1, \dots, R_k$ partition the color space $[\Delta+1]$.
(2) Each v encodes $R_{\boldsymbol{t}(v)} \cap \varphi(N(v)\cap K)$ into a $c\log n$-sized bit-map and broadcasts it.
(3) For each $i \in [k]$, each $v \in K$ combines the bit-maps received from its neighbors in T_i as

$$\bigcup_{u\in N(v)\cap T_i} (R_i \cap \varphi(N(u)\cap K))$$

and takes it for $R_i \cap \varphi(K)$.

Lemma 4.11. *Let K be an almost-clique.* LEARNPALETTE *has each $v \in K$ learn $L_\varphi(K)$ in $O(1)$ rounds of* BCONGEST *with high probability.*

PROOF. In $\Delta+1$-coloring, learning $L_\varphi(K)$ is equivalent to learning the *used* colors $\varphi(K)$. LEARNPALETTE requires $O(1)$ rounds of BCONGEST, as each node in K only sends one $c\log n$-bit message. Let us consider a color $\chi \in \varphi(K)$, a node $v \in K$, and argue that v learn χ. Let R_i be such that $\chi \in R_i$, and $u \in K$ a node with color χ. Since T_i 2-hop connects K

w.h.p., by Lemma 4.9, there exists a node in $T_i \cap N(u) \cap N(v)$. Such a node contains χ in the bitmap it computes in Step 2 of LearnPalette, and v receives this bitmap in Step 3. As this works for every $\chi \in \varphi(K)$ and $v \in K$, all $v \in K$ learn $\varphi(K)$. ∎

4.5.2. Sampling the permutation. At a high level, the $O(\log\log n)$ algorithm for permuting the nodes presented in this section has the nodes undergo two shuffling steps. Nodes first undergo a "rough shuffling", which puts them into buckets, roughly positioning them in the permutation. Each group then does a "fine shuffling" to give each node its exact position.

An important step in both our $O(\log\log n)$ and our $O(1)$ implementation is giving nodes $O(\log\log n)$-bit labels unique within their buckets. Using the smaller labels instead of the original node IDs allows each bucket to reduce bandwidth usage when they later need to describe a random permutation of its elements.

Algorithm 11. Procedure Relabel, in 2-hop connected set of nodes $T \subseteq V$, for subset $S \subseteq T$.

Parameters: Let $c = O(1)$ be a large enough constant, $x := \lceil c \log n / \log\log n \rceil$.

(1) Each $v \in S$ samples and broadcasts x labels in $[|S|^2 \log n]$, picked u.a.r. and independently.
(2) Each $v \in T$ broadcasts an x-sized bit-map indicating, for each $j \in [x]$, whether multiple nodes in $S \cap N(v)$ have the same j-th label.
(3) Compute by converge-cast in T the bitwise OR of the bitwise emitted in Step 2.
(4) If for a $j \in [x]$, all nodes in S have distinct j-th labels, vertices of S uses them as new labels. If there exist multiple such j, they choose the smallest one.

Lemma 4.12. *Suppose S has size* $\mathrm{poly}(\log n)$. Relabel *succeeds at relabeling S in $O(1)$* BCONGEST *rounds, with high probability.*

Proof. First, note that $O(1)$ BCONGEST rounds suffice to compute $|S|$ for Step 1, as T is 2-hop connected. Since $|S|^2 \log n \leqslant \mathrm{poly}(\log n)$, each label

sent by a node $v \in S$ during Step 1 is representable with $O(\log\log n)$ bits. Thus, $x \leqslant O(\log n/\log\log n)$ labels can be transmitted in $O(1)$ rounds.

As T 2-hop connects itself (a fortiori S), two nodes of S with a common j-th label are necessarily detected by a common neighbor during Step 2. Taking the bitwise OR of all x-sized bitmaps sent in this step, the nodes in T learn for which $j \in [x]$ there exists nodes of S that picked the same j-th label. And so, they also learn for which j all vertices in S sampled distinct labels. As such, they all agree on which j to use in Step 4 to relabel S.

We now analyze the probability that the relabeling succeeds, i.e., that a $j \in [x]$ as used in Step 4 exists. For each $j \in [x]$, each j-th sampled label in S has probability at most $1/(|S|\log n)$ of conflicting with one of the other $|S|-1$ many j-th labels. Hence, by union bound, the j-th labels have a collision with probability at most $1/(\log n)$. Having x independent instances implies success for at least one with probability at least $1-(\log n)^{-x} = 1-2^{-x\log\log n} = 1-2^{-c\log n} = 1-n^{-c}$, i.e., w.h.p. ∎

ALGORITHM 12. Procedure PERMUTE, in almost-clique K, on subset $S \subseteq K$ of the nodes.

Parameters: Let $c = O(1)$ be a large enough constant,

$$k := \lfloor \Delta/(c\log n) \rfloor, \quad \text{and} \quad x := \lceil c\log n/\log\log n \rceil \ .$$

(1) **Rough bucketing.** Each $v \in K$ independently picks a random $\boldsymbol{t}(v) \in [k]$ u.a.r.
For each $i \in [k]$, let $T_i := \{v \in K : \boldsymbol{t}(v) = i\}$ and $S_i := T_i \cap S$.

(2) **Counting buckets.** For each $i \in [k]$, the nodes in T_i compute and broadcast $|S_i|$.

(3) **Relabeling.** Within each T_i, $i \in [k]$, use RELABEL on S_i.

(4) **Permuting within buckets.** Within each T_i, the maximum ID node gathers the new labels of S_i, picks a random permutation ρ_i of S_i, and sends it to T_i, all along a BFS tree.

(5) **Output.** Each $v \in S_i$ takes $\pi(v) := \rho_i(v) + \sum_{j<i}|S_j|$ as its index in the output π.

Lemma 4.13. *With high probability,* PERMUTE *outputs a permutation of S in $O(\log\log n)$ rounds. For each permutation π of S, the probability of sampling π is bounded by $\frac{1}{(1-1/\mathrm{poly}(n))\cdot|S|!}$*

PROOF. By Lemma 4.9, the sets T_i computed in Step 1 2-hop connect K, w.h.p., and in particular have diameter 2. Assuming this holds, Step 2 only takes $O(1)$ rounds using a aggregation and dissemination on the depth-2 BFS tree within each T_i. This allows each $v \in S_i$ to compute $\sum_{j<i}|S_j|$ for the last step of the algorithm.

In addition, it also holds w.h.p. that each $S_i \subseteq T_i$ has size $O(\log n)$. Assuming this holds, running RELABEL in Step 3 only requires $O(1)$ rounds per Lemma 4.12, and it succeeds w.h.p. Finally, the process takes $O(\log\log n)$ rounds due to Step 4, during which a leader node within each T_i broadcasts $O(\log n)$ labels of $O(\log\log n)$ bits each.

We now argue the approximate uniformity of the sampling. Consider the random process in which each node in S picks a random ordered bucket independently and u.a.r, and then each bucket is permuted uniformly at random. Let μ be the distribution of the permutation generated by this process. Clearly, μ is the uniform distribution. PERMUTE is the same as this process, except it does not output anything if some high probability event $\mathcal{E}$ does not hold. More precisely, the high probability event $\mathcal{E}$ corresponds to all buckets being 2-connected, all buckets being of $O(\log n)$ size, and RELABEL succeeding. Let μ_1 be the distribution μ conditioned on $\mathcal{E}$ holding, and μ_2 be μ conditioned on $\mathcal{E}$ not holding. Distribution μ_1 is the output distribution of PERMUTE, and we have $\mu = (1-\mathbb{P}[\mathcal{E}])\mu_1 + \mathbb{P}[\mathcal{E}]\mu_2$. Thus, for each permutation π, we have $\mu_1(\pi) \leqslant \mu(\pi)/(1-\mathbb{P}[\mathcal{E}]) = 1/((1-1/\mathrm{poly}(n))|S|!)$. ∎

4.5.3. Reducing the complexity to a constant. Our $O(1)$ round implementation improves on the running time by splitting buckets from the first "rough shuffling" into sub-buckets, and arguing that most such buckets satisfy properties allowing them to use a leader to permute themselves as in Algorithm 12, while buckets that fail this second sub-bucketing are few enough that they can be efficiently permuted with the help of the whole almost-clique.

Lemma 4.14. *There is an algorithm that w.h.p. outputs a permutation of S in $O(1)$ rounds of* BCONGEST*. For each permutation π of S, the probability of sampling π is bounded by $\frac{1}{(1-1/\text{poly}(n))\cdot|S|!}$*

A key ingredient in the improved version of our algorithm is strengthening the properties satisfied by buckets. We call a *k-bucketing* of $S \subseteq K$ a function $t : S \to [k]$, defining sets $T_i := \{v \in K : t(v) = i\}$ for each $i \in [k]$. In our improved $O(1)$ round algorithm, we also perform a second k'-bucketing t' of each set T_i, defining sets $T_{i,i'} := \{v \in T_i : t'(v) = i'\}$ for each $(i, i') \in [k] \times [k']$. We will aim for the random subsets of almost-cliques formed to themselves have the properties almost-cliques. We seek to maintain the following properties:

Definition 4.15. For $\varepsilon, \varepsilon' \in (0, 1/2)$ and a set of nodes K,

- K is said to be *ε-almost-clique-like* (ε-AC-like) if $|N(v) \cap K| \geqslant (1-\varepsilon)|K|$ for all $v \in K$.
- For integers k and $i \in [k]$, a k-bucketing of K is said to *ε'-almost-clique-preserve* (ε'-AC-preserve) its i-th bucket T_i iff for all $i \in [k]$ and $v \in K$, we have
$$|N(v) \cap T_i| \in (1 \pm \varepsilon')|N(v) \cap K|/k \ .$$
The bucketing is said to be *ε'-almost-clique-preserving* (ε'-AC-preserving) if it ε'-AC-preserves its k buckets.

The following combinatorial lemma relates the two notions.

Lemma 4.16. *Let $\varepsilon, \varepsilon'$ be two positive constants such that $\varepsilon + \varepsilon' < 1/2$. Let K be ε-AC-like, and let T be an ε'-AC-preserved bucket of a k-bucketing of K. Then, T has size $|T| \in (1 \pm 2(\varepsilon + \varepsilon'))|K|/k$ and is $2(\varepsilon + \varepsilon')$-AC-like.*

Proof. For each $v \in K$, the bounds on $|K|$, $|N(v) \cap T|$, and $|N(v) \cap K|$ from K being ε-AC-like and T being ε'-AC-preserved yield:

$$|N(v) \cap T| \geqslant (1-\varepsilon')\frac{|N(v) \cap K|}{k} \geqslant (1-\varepsilon')(1-\varepsilon)\frac{|K|}{k} \tag{4.3}$$

and

$$|N(v) \cap T| \leqslant (1+\varepsilon')\frac{|N(v) \cap K|}{k} \leqslant (1+\varepsilon')\frac{|K|}{k} \ . \tag{4.4}$$

Counting edges between T and K two ways gives:

$$\sum_{v\in K}|N(v)\cap T| = \sum_{v\in T}|N(v)\cap K| \ .$$

In combination with $(1-\varepsilon)|T|\cdot|K| \leqslant \sum_{v\in T}|N(v)\cap K| \leqslant |T|\cdot|K|$ (from K being ε-AC-like), it gives us bounds on $|T|$:

$$\text{(from Eq (4.3))} \qquad |T| \geqslant (1-\varepsilon')(1-\varepsilon)\frac{|K|}{k} \geqslant (1-\varepsilon-\varepsilon')\frac{|K|}{k}$$

and

$$\text{(from Eq (4.3))} \qquad |T| \leqslant \frac{1+\varepsilon'}{1-\varepsilon}\cdot\frac{|K|}{k} \leqslant (1+2(\varepsilon+\varepsilon'))\frac{|K|}{k} \ .$$

(using that $1/(1-\varepsilon) \leqslant 1+2\varepsilon$ for $\varepsilon \leqslant 1/2$)

It follows that T is $2(\varepsilon+\varepsilon')$-AC-like because, for each $v \in K$,

$$\begin{aligned}
\text{(from Eq (4.3))} \qquad |N(v)\cap T| &\geqslant (1-\varepsilon')(1-\varepsilon)\frac{|K|}{k} \\
\text{(from Eq (4.4))} \qquad &\geqslant \frac{(1-\varepsilon')(1-\varepsilon)}{1+\varepsilon'}|T| \\
&\geqslant (1-2\varepsilon-2\varepsilon')|T| \ . \qquad \blacksquare
\end{aligned}$$

Note that if $\varepsilon+\varepsilon' < 1/4$ the sets T_i defined by an ε'-AC-preserving bucketing within an ε-AC-like set K 2-hop connect K. Following the same reasoning as for random groups (Lemma 4.9), a random k-bucketing in an almost-clique is ε'-AC-preserves its buckets w.e.h.p. in $\varepsilon'^2|K|/k$.

Lemma 4.17. *Let $\varepsilon, \varepsilon'$ be two positive constants such that $\varepsilon+\varepsilon' < 1/2$. Let K be ε-AC-like and k an integer. Consider a k-bucketing of K picked uniformly at random. For each $i \in [k]$, the probability that the i-th bucket fails to be ε'-AC-preserved is at most $2|K|\exp(-\varepsilon'^2|K|/(6k))$.*

In one of the last steps of our $O(1)$-round Permute algorithm, we perform a second bucketing within previously formed buckets and argue that only a few buckets from this second bucketing are not ε''-AC-preserved.

Algorithm 13. Procedure Permute, in almost-clique K, on subset $S \subseteq K$ of the nodes.

Parameters: Let $c = O(1)$ be a large enough constant,

$\varepsilon' := 1/24 - \varepsilon$, $\varepsilon'' := 1/12$, $k := \lfloor \Delta/(c \log n) \rfloor$, and $k' := \lceil c \log\log n \rceil$.

(1) **Rough bucketing.** Each $v \in K$ independently picks a random $\boldsymbol{t}(v) \in [k]$ u.a.r.
For each $i \in [k]$, let $T_i := \{v \in K : \boldsymbol{t}(v) = i\}$ and $S_i := T_i \cap S$.

(2) **Counting rough buckets.** For each $i \in [k]$, the nodes in T_i compute and broadcast $|T_i|$ and $|S_i|$.

(3) **Relabeling.** Within each T_i, $i \in [k]$, use RELABEL on S_i.

(4) Within each T_i, $i \in [k]$,
   (a) **Fine bucketing.** Each $v \in [T_i]$ picks a random bucket $\boldsymbol{t}'(v) \in [k']$.
   For each $(i, i') \in [k] \times [k']$, let $T_{i,i'} := \{v \in T_i : \boldsymbol{t}'(v) = i'\}$ and $S_{i,i'} := T_{i,i'} \cap S$.
   (b) **Counting fine buckets.** Compute and broadcast all $|T_{i,i'}|$ and $|S_{i,i'}|$ for $i' \in [k']$.
   (c) For each $i' \in [k']$, **if** $S_{i,i'}$ is ε''-AC-preserved in T_i,
   then: Permute within fine bucket. The maximum ID node of $T_{i,i'}$ aggregates the $O(\log\log n)$-bit labels of $S_{i,i'}$, picks u.a.r. a permutation $\rho_{i,i'}$ of $S_{i,i'}$, sends it to $T_{i,i'}$.
   else: each $v \in S_{i,i'}$ joins the set R, to be permuted in the next step.

(5) **Permuting leftover fine buckets.**
   (a) Each $v \in R$ picks a random $\boldsymbol{r}(v) \in [n^c]$ and broadcasts $(\mathsf{ID}(v), \boldsymbol{t}(v), \boldsymbol{t}'(v), \boldsymbol{r}(v))$.
   (b) Nodes in K use Many-to-All Broadcast to disseminate the tuples from R to all of K.
   (c) For each $(i, i') \in [k] \times [k']$ such that $S_{i,i'} \subseteq R$, nodes in $S_{i,i'}$ order themselves according to their $\boldsymbol{r}(v)$'s. Let $\rho_{i,i'}$ be the resulting permutation of $S_{i,i'}$.

(6) **Output.** $\forall i, i'$, $v \in S_{i,i'}$ takes index $\pi(v) := \rho_{i,i'}(v) + \sum_{j<i} |S_j| + \sum_{j'<i'} |S_{i,j'}|$ in output.

Proof of Lemma 4.14. First, the fact that our $O(1)$-round Permute procedure has an output distribution close to uniform follows from the same argument that showed this property for our $O(\log\log n)$-round Permute procedure.

(Steps 1 and 3.) By Lemma 4.17, Step 1 (rough bucketing) produces an ε'-AC-preserving bucketing with probability at least $2|K|\exp(-\varepsilon'^2|K|/(6k)) \leqslant n^{-\Omega(\varepsilon'^2 c)}$, i.e., w.h.p. We condition on this high-probability event. The rough bucketing being ε'-AC-preserving, by Lemma 4.16, each T_i is $2(\varepsilon+\varepsilon')$-AC-like, with $2(\varepsilon+\varepsilon') = (1/12)$. Each T_i thus has diameter 2 and can count it size and the size of its subset S_i in $O(1)$ rounds during Step 2. Since every node in K is adjacent to a node in T_i, all of K learns all $|S_i|$, $i \in [k]$.

(Step 4.) Relabeling works as in the previous $O(\log\log n)$-round algorithm. Consider now the second bucketing of Step 4a. As each $T_{i,i'}$ and $S_{i,i'}$ are of size at most $O(\log n)$, and there are $k' \in O(\log\log n)$ values to count in Step 4b, describing all those values only requires $O(\log^2\log n)$ bits. Counting all of them within T_i by aggregation along a BFS tree can be done in $O(1)$ rounds, and disseminating all values back to T_i is similarly fast. Note that vertices can detect which $T_{i,i'}$ is ε''-AC-preserved by verifying by counting $|N(v) \cap T_{i.i'}|$ and aggregating a the bitwise OR of a k'-bitmap indicating for which $i' \in [k']$ we detected a failure.

Within each T_i in which the second bucketing succeed, Step 4c finishes to permute its elements in $O(1)$ rounds, since the maximum ID node within each $T_{i,i'}$ only has to send $O(\log n/\log\log n)$ labels of size $O(\log\log n)$ in an 1/3-AC-like, and thus low diameter, set $T_{i,i'}$.

(Step 5.) We finish by arguing that permuting the elements in R within their $S_{i,i'}$ groups can be done in $O(1)$ rounds, w.h.p. By Corollary 4.10, if R contains at most $O(\Delta/\log n)$ nodes, Many-to-All-Broadcast succeeds in sharing all of R's tuples in $O(1)$ rounds, with high probability.

Let us now show that, w.h.p., R contains $O(\Delta/\log n)$ nodes. We henceforth fix the values $t(v) = \boldsymbol{t}(v)$ sampled in Step 1 so that the rough bucketing succeeding, i.e., every $|T_i| \in (1 \in 1/2)c\log n$. For each $i, i' \in [k] \times [k']$, let $\mathbf{X}_{i,i'}$ be the indicator random variable for the i'-th bucket in T_i not being ε'-AC-preserved. Let $\mathbf{Z} = \sum_{i=1}^{k}\sum_{i'=1}^{k'}\mathbf{X}_{i,i'}$ be the total number of buckets

which are not ε'-AC-preserved. Note that $\mathbf{Z}$ is a function of the independent variables $\boldsymbol{t}'(v)$ sampled during Step 4a.

From Lemma 4.17 (with $K = T_i$, $\varepsilon = \varepsilon'$ and $\varepsilon' = \varepsilon''$), we obtain $\mathbb{E}[\mathbf{X}_{i,i'}] \leqslant |T_i| e^{-\varepsilon''^2 |T_i|/(6k)}$ for every $i \in [k]$ and $i' \in [k']$. Since T_i has size $|T_i| \in (1 \pm 1/2) c \log n$, we have that,

$$\begin{aligned}\mathbb{E}[\mathbf{Z}] = \sum_{i=1}^{k} \sum_{i'=1}^{k'} \mathbb{E}[\mathbf{X}_{i,i'}] &\leqslant k \cdot k' \cdot 4(c \log n) \cdot e^{-\varepsilon''^2 c \log n/(12k')} \\ &= 4\Delta \cdot \log\log n \cdot e^{-\log n/(12^3 \log\log n)} \,.\end{aligned}$$

For n large enough, this yields

$$\mathbb{E}[\mathbf{Z}] \leqslant \Delta/(h^2 \cdot \log^4 n) \quad \text{where} \quad h := 2c \log n$$

Changing the value of each $\boldsymbol{t}'(v)$ affects at most two buckets. Therefore $\mathbf{Z}$ is 2-Lipschitz. Furthermore, f is h-certifiable, as it suffices to reveal $\boldsymbol{t}'(T_i)$ to show that one of its buckets is not ε'-AC-preserved. Applying Talagrand's inequality, we get that:

$$\mathbb{P}[\mathbf{Z} > 4c\Delta/\log^2 n] \leqslant 4 \cdot \exp\left(- \frac{\left(4c\Delta/\log^2 n - 60\sqrt{h\,\mathbb{E}[\mathbf{Z}]}\right)^2}{32h^2\,\mathbb{E}[\mathbf{Z}]} \right)$$

(using that $\sqrt{h\,\mathbb{E}[\mathbf{Z}]} \leqslant \sqrt{\Delta} \ll \Delta/\log^2 n$)

$$\leqslant 4 \cdot \exp\left(- \frac{c^2\Delta^2/\log^2 n}{32 \cdot \Delta/\log^4 n} \right)$$

(since $\Delta \geqslant \Omega(\log^3 n)$)

$$\leqslant 4 \cdot \exp(-\Delta/32) \ll 1/\mathrm{poly}(n) \,.$$

Therefore, with high probability, at most $O(\Delta/\log^2 n)$ buckets join R. Each has size $O(\log n)$, so R contains at most $O(\Delta/\log n)$ nodes with high probability. ∎

4.6. Colorful Matching

During the synchronized color trial (Algorithm 5) when coloring put-aside vertices (Section 4.7), we need publicly known lists of colors that approximate well dense vertices' palettes. We use *the clique palette* $L_\varphi(K) = [\Delta + 1] \setminus \varphi(K)$ as an approximation of $L_\varphi(v)$ for every $v \in K$. However, when K is larger than $\Delta + 1$, the clique palette may run out of colors before all vertices are colored. To circumvent the issue, we compute a *colorful*

matching. The idea was first used by [ACK19] to deal with that same issue. It was first used distributively in [FGHKN23; FGHKN24].

Definition 4.18. We say K contains a k-colorful matching in φ if $G[K]$ contains at least k repeated colors in φ.

The name comes from the fact that a k-colorful matching in K induces a k-sized anti-matching in $G[K]$ where the endpoints of each anti-edge are colored the same. As such, we often refer to k as the size of the colorful matching. Observe that pairs of non-adjacent vertices colored the same by slack generation contribute to the colorful matching. Nonetheless, we may bend the phrasing for convenience when we say that an algorithm computes or finds a colorful matching of a given size — we mean that it increased the size of the colorful matching by this much.

Basic accounting of used/unused colored in and out of almost-cliques yields Eq (4.5), which is the main guarantee we seek from the colorful matching.

Lemma 4.19 (Accounting Lemma I)**.** *Suppose K contains a k-colorful matching in φ. For every $v \in K$, we have*

$$|L_\varphi(v) \cap L_\varphi(K)| \geqslant \Delta - \deg(v) + |(N(v) \cup K) \setminus \operatorname{dom}\varphi| - a(v) + k \ . \tag{4.5}$$

In words, Eq (4.5) says that v has as many colors available as there are uncolored vertices in K, up to an error equal to its anti-degree, which is compensated by the size of the colorful matching.

Proof of Lemma 4.19. Since K contains at least k repeated colors, we know that

$$|K \cap \operatorname{dom}\varphi| = \sum_{\chi \in \varphi(K)} |\{v \in K : \varphi(v) = \chi\}| \geqslant |\varphi(K)| + k$$

and using the following two basic indentities

$$|\varphi(N(v) \cup K)| = |\varphi(E(v) \cup K)| \leqslant |\varphi(K)| + |\varphi(E(v))|$$
$$|K| = |K \cap \operatorname{dom}\varphi| + |K \setminus \operatorname{dom}\varphi|$$

we count colors available to v as

$$\begin{aligned} |L_\varphi(v) \cap L_\varphi(K)| &= \Delta + 1 - |K| + |K| - |\varphi(N(v) \cup K)| \\ &\geqslant \Delta + 1 - |K| + |K \setminus \operatorname{dom}\varphi| - |\varphi(E(v))| + k \end{aligned}$$

Using Eq (3.1) and rearranging terms, we obtain Eq (4.5) as

$$\begin{aligned}|L_\varphi(v) \cap L_\varphi(K)| &\geqslant \Delta - \deg(v) + |K \setminus \operatorname{dom}\varphi| + e(v) - |\varphi(E(v))| - a(v) + k \\ &\geqslant \Delta - \deg(v) + |K \setminus \operatorname{dom}\varphi| + |E(v) \setminus \operatorname{dom}\varphi| - a(v) + k \\ &\geqslant \Delta - \deg(v) + |(N(v) \cup K) \setminus \operatorname{dom}\varphi| - a(v) + k \ . \qquad \blacksquare\end{aligned}$$

Lemma 4.19 guarantees the clique palette is a good approximation of v's palette only when the colorful matching has size at least $a(v)$. Note that K can be a clique plus $O(1/\varepsilon)$ vertices of anti-degree $\varepsilon\Delta$, in which case we cannot ensure Eq (4.5) holds for every vertex in K. It nonetheless suffices to ensure that a constant fraction of the vertices verify Eq (4.5): those who do not verify it can be colored first (see Definition 4.25 for more details). The remainder of this section is dedicated to an algorithm for finding such a $\Omega(a(K)/\varepsilon)$-sized colorful matching.

Proposition 4.20. *Let $\lambda \leqslant O(1/\varepsilon)$ and $r \leqslant \Delta/2$ and assume $\Delta \gg \log n$. Suppose φ is the coloring from running* SlackGeneration *with $p \leqslant 1/8$. W.h.p. the coloring can be extended in $O(\lambda)$ rounds of* BCONGEST *such that every almost-clique with $a(K) \geqslant c \log n$ has a colorful matching of size at least $\lambda \cdot a(K)$. Moreover, it does not use colors $\{1, 2, \ldots, r\}$ and colors a vertex iff its color is repeated by exactly one other vertex in its almost-clique.*

4.6.1. Proof of Proposition 4.20. Let D be a set of colors, F a set of anti-edges and φ a partial coloring. Define

$$\operatorname{Avail}_{\varphi,D}(F) = \sum_{\{u,v\} \in F} |L_\varphi(u) \cap L_\varphi(v) \cap D| \tag{4.6}$$

This potential function measures how likely a random color try is to provide slack. It is easy to verify that when no vertex is colored, we have the potential is $\Omega(a(K)\Delta^2)$, since an almost-clique contains $a(K)|K|/2$ anti-edges and we use at least $\Delta/2$ colors. When using this algorithm after running slack generation, some anti-edges are lost. We shall assume for now that the potential is affected by a constant factor and we show in Lemma 4.23 that it is the case with high probability in n.

Algorithm 14 runs $O(1/\varepsilon)$ randomized color trials, retaining a color *only when it provides slack to its almost-clique* (Step 3). The proof of

Lemma 4.21 has two parts. The first one is akin to the analysis of SLACK-GENERATION and shows that we are likely to create $\gamma \cdot a(K)$ slack as long as $\text{Avail} \geqslant \Omega(a(K)\Delta^2)$. The second part shows that if Avail becomes small while running Algorithm 14, it must be because we found a large colorful matching. The later part is the one for which Step 3 in Algorithm 14 is crucial.

ALGORITHM 14. COLORFULMATCHING

Input: constants $\alpha \in (0,1)$, $0 < \lambda < \alpha/(36\varepsilon)$ and $r \leqslant \Delta/2$.
Output: a $\lambda \cdot a(K)$-colorful matching in each almost-clique such that $a(K) \geqslant \Theta(\log n)$.

Repeat $T = \Theta(\lambda/\alpha)$ times:

(1) every dense vertex samples $\boldsymbol{\chi}(v) \in (r, \Delta+1]$ uniformly at random.
(2) if $\boldsymbol{\chi}(v) \in L_\varphi(v) \setminus \boldsymbol{\chi}(N(v))$, set $\boldsymbol{\chi}(v) = \perp$.
(3) $v \in K$ retains its color in φ iff exactly one other vertex $u \in K$ has $\boldsymbol{\chi}(u) = \boldsymbol{\chi}(v)$.

Lemma 4.21. *Let $\alpha \in (0,1)$, $\lambda \in (0, \alpha/36\varepsilon)$, $r \leqslant \Delta/2$, φ be a partial coloring, K an ε-almost-clique and F the anti-edges in $G[K]$ with both endpoints uncolored. Suppose that $\text{Avail}_{\varphi,(r,\Delta+1]}(F) \geqslant \alpha \cdot a(K)\Delta^2$ and $a(K) \gg 1/\alpha$. After Algorithm 14, w.e.h.p. in $\alpha \cdot a(K)$, the size of the colorful matching in K has increased by at least $\lambda a(K)$.*

PROOF. For every $i \in \{0, 1, \dots, T\}$, call φ_i the coloring after the i-th iteration of Steps 1 to 3; by extension $L_i(v) = L_{\varphi_i}(v)$ and $\text{Avail}_i = \text{Avail}_{\varphi_i, L_{\varphi_i}(K)\setminus[r]}(F_i)$ where F_i is the set of anti-edges in $G[K]$ with both endpoints uncolored after the i-th iteration. Define $\mathbf{Z}_i$ as the number of colors retained *by exactly two vertices of K* during the i-th iteration of Step 3. For every $f = \{u, v\} \in F_i$, let $\mathbf{X}_{i,f}$ indicate if endpoints of f sampled the same color in the i-th iteration of Step 1 and no other vertex in $(K \cup N(u) \cup N(v)) \setminus \{u, v\}$ also sampled that color. Since every vertex independently samples a uniform color in a set of size $q = \Delta + 1 - r \geqslant \Delta/2$

and that u and v have at most $\varepsilon\Delta$ external neighbors each, we have that

$$\begin{aligned}\mathbb{E}[\mathbf{Z}_i] &\geqslant \sum_{\{u,v\}\in F_{i-1}} \mathbb{E}[\mathbf{X}_{i,f}] \\ &\geqslant \sum_{\{u,v\}\in F_{i-1}} \frac{|L_{i-1}(u)\cap L_{i-1}(v)\cap L_{i-1}(K)|}{(\Delta+1)^2}\left(1-\frac{1}{q}\right)^{|K|+2\varepsilon\Delta} \\ &\geqslant \gamma\cdot \mathrm{Avail}_{i-1}/\Delta^2 \quad \text{where} \quad \gamma = 2^{-8}\,.\end{aligned}$$

(using $1-x \geqslant 2^{-2x}$ when $0 \leqslant x \leqslant 1/2$)

To show concentration, we follow the same approach as in Section 3.3, by writing $\mathbf{Z}_i = \mathbf{Y}_i - \mathbf{B}_i$ where $\mathbf{Y}_i$ counts the number of colors sampled by both endpoints of at least one anti-edge in F_{i-1} and $\mathbf{B}_i$ counts the number of colors sampled by both endpoints of some anti-edge $\{u,v\} \in F_{i-1}$ *and* a vertex in $(K\cup N(u)\cup N(v))\setminus\{u,v\}$ also sampled that color. The variable $\mathbf{Y}_i$ is 1-Lipschitz and 2-certifiable (each color is certified by the random colors sampled by the endpoints of one anti-edge) and $\mathbf{B}_i$ is 1-Lipschitz and 3-certifiable (each color is certified by the random colors sampled by the endpoints of one anti-edge plus the random color of that other vertex). So Talagrand's inequality applies and we get that

$$\mathbb{P}\Big[|\mathbf{Y}_i - \mathbb{E}[\mathbf{Y}_i]| > \mathbb{E}[\mathbf{Z}_i]/4\Big] \leqslant 4\exp\left(-\frac{\Big(\mathbb{E}[\mathbf{Z}_i]/2 - 30\sqrt{\mathbb{E}[\mathbf{Y}_i]}\Big)^2}{16\,\mathbb{E}[\mathbf{Y}_i]}\right).$$

Using that $\mathbb{E}[\mathbf{Y}_i] = \mathrm{Avail}_{i-1}/q^2 = O(\mathbb{E}[\mathbf{Z}_i])$, for $\mathrm{Avail}_{i-1}/q^2 \gg 1$ this is

$$\leqslant \exp(-\Omega(\mathbb{E}[\mathbf{Z}_i])) = \exp(-\Omega(\mathrm{Avail}_i/\Delta^2))$$

Similarly, (with slightly different constants)

$$\mathbb{P}\Big[|\mathbf{B}_i - \mathbb{E}[\mathbf{B}_i]| > \mathbb{E}[\mathbf{Z}_i]/4\Big] \leqslant \exp\big(-\Omega(\mathrm{Avail}_{i-1}/\Delta^2)\big)\,.$$

And thus, w.e.h.p. in $\mathrm{Avail}_{i-1}/\Delta^2$, the variable $\mathbf{Z}_i$ is concentrated as $|\mathbf{Z}_i - \mathbb{E}[\mathbf{Z}_i]| \leqslant |\mathbf{Y}_i - \mathbb{E}[\mathbf{Y}_i]| + |\mathbf{B}_i - \mathbb{E}[\mathbf{B}_i]| \leqslant \mathbb{E}[\mathbf{Z}_i]/2$. So if $\mathrm{Avail}_i \geqslant \alpha\cdot a(K)\Delta^2/6$ for every $i\in[T]$, with probability $1 - T\exp(-\Omega(\alpha a(K)))$, the number of colors repeated in K by Algorithm 14 is

$$\sum_{i\in[T]} \mathbf{Z}_i \geqslant T\cdot\gamma/2\cdot\alpha\cdot a(K)/6 \geqslant \lambda\cdot a(K) \quad \text{by choosing} \quad T = \left\lceil\frac{12\lambda}{\alpha\gamma}\right\rceil.$$

If there exist $i \in [T]$ such that $\text{Avail}_i < \alpha \cdot a(K)\Delta^2/6$, we claim that a large colorful matching must exists. Indeed, Avail_i decreases only if an external neighbor gets colored or we retained a color in K during Step 3. Since vertices of K have at most $\varepsilon\Delta$ external neighbors, the total loss from colored external neighbors is upper bounded by $(1+\varepsilon)\varepsilon\Delta^2$. Each time we color a vertex in Step 3, Avail_i decreases by at most $4\varepsilon\Delta^2$: at most $2\varepsilon\Delta(\Delta+1)$ from removing in F_i all anti-edges incident on that vertex, and at most $a(K)|K|/2 \leqslant (1+\varepsilon)\varepsilon\Delta^2$ from removing one color in $L_i(K)$. Let us call k the number of vertices colored by the algorithm. We have that

$$\text{Avail}_0 - \text{Avail}_i \leqslant 2\varepsilon\Delta^2 + k \cdot 4\varepsilon\Delta^2$$

rearranging the terms and using that Avail_i is small along with the assumption that $\text{Avail}_0 \geqslant \alpha \cdot a(K)\Delta^2$, $a(K) \gg 1/\alpha$ and $\lambda < \alpha/(36\varepsilon)$, the algorithm colors

$$k \geqslant \frac{\text{Avail}_0 - \text{Avail}_i - 2\varepsilon\Delta^2}{4\varepsilon\Delta^2} \geqslant \frac{2\alpha/9 \cdot a(K)\Delta^2}{4\varepsilon\Delta^2} = \frac{\alpha}{18\varepsilon} \cdot a(K) \geqslant 2\lambda \cdot a(K)$$

many vertices. Since Step 3 colors vertices only if their color is repeated, the number of repeated colors at the end of the algorithm is at least $k/2$, which concludes the proof. ∎

It is not obvious that Step 3 in Algorithm 14 can be implemented in $O(1)$ rounds of BCONGEST as it involves communication between non-neighbors of K for each color. We show it can be done in $O(1)$ rounds of BCONGEST by using random groups and aggregation on trees.

Lemma 4.22. *Algorithm 14 runs in $O(\lambda/\alpha)$ rounds of* BCONGEST *with high probability.*

Proof. Steps 1 and 2 can easily be implemented in $O(1)$ rounds of BCONGEST as it only involves broadcasting colors. We focus on showing that Step 3 can be implemented in $O(1)$ rounds with high probability, which implies the lemma.

To spread the workload, split the colors of $(r, \Delta+1]$ in contiguous ranges $R_1, \dots, R_t$ of $k = \Theta(\log n)$ colors each. Make as many random groups $T_1, \dots, T_t$ in K. Note that $t \leqslant (\Delta+1)/k$, so by Lemma 4.9, w.h.p., every T_i 2-hop connects K. Now, vertices of T_i will compute redundant colors in R_i by aggregation on a tree $\mathscr{T}_i$. Importantly, $\mathscr{T}_i$ must span every

vertex which sampled a color in R_i. So, first compute in each $G[T_i]$ a depth-2 BFS tree rooted at an arbitrarily vertex. Every vertex with $\boldsymbol{\chi}(v) \in R_i$ at Step 3 broadcasts the identifier of one neighbor $p(v) \in N[v] \cap T_i$. We know this neighbor exist because T_i 2-hop connects K. If $v \in T_i$, then it chooses itself, i.e., $p(v) = v$. The tree $\mathscr{T}_i$ is the BFS tree on T_i extended with the edges $\{v, p(v)\}$ for every vertex with $\boldsymbol{\chi}(v) \in R_i$ at Step 3.

Vertices of $\mathscr{T}_i$ discriminate colors in R_i used at most once, at least twice and at least three times in K as follow. Each subtree of $\mathscr{T}_i$ aggregates three k-bit bitmaps: the first one represents which colors of R_i have been retained by vertices of the subtree after Step 2; the second one represents which color are repeated by vertices of the subtree; and the last one, which colors are used by at least three vertices. Consider a color $\chi \in R_i$ which gets repeated. The lowest common ancestor of u and v with $\boldsymbol{\chi}(u) = \boldsymbol{\chi}(v) = \chi$ in Step 3 sees from the first bitmap that χ is used by different vertices in its subtree (since they belong to different subtrees) and can thus update the second bitmap accordingly. Similar case analysis using the first and second bitmap allows to compute the third one. Propagating the second and third bitmap back down $\mathscr{T}_i$ informs every vertex in T_i of if it should retain its color or not. The convergecast can be implemented in $O(1)$ rounds of BCONGEST since it requires sending only one message to the parent and every vertex is involved in at most two such operation: one for the random color $\boldsymbol{\chi}(v)$ it samples and one for the random group it belongs to. ∎

To complete the proof of Proposition 4.20 using Lemma 4.21, we need to argue that $\mathrm{Avail}_{\varphi,(r,\Delta+1]}(F)$ is large when φ is the partial coloring *after slack generation*. Since SLACKGENERATION only colors activated vertices, it suffices to show that $\mathrm{Avail}_{\varphi,L_\varphi(K)\setminus[r]}(F')$ is large for F' the set of edges with both endpoints unactivated during SLACKGENERATION.

Lemma 4.23. *Suppose φ is the coloring obtained from* SLACKGENERATION *with $p \leqslant 1/8$. If F if the set of anti-edges in $G[K]$ with both endpoints uncolored, w.e.h.p. in $a(K)$ over the randomness of* SLACKGENERATION*, we have $Avail_{\varphi,(r,\Delta+1]}(F) \geqslant a(K)\Delta^2/32$.*

PROOF. We first bound the number of used colors in K by the number of vertices activated during SLACKGENERATION. Since $p \leqslant 1/8$, by

Chernoff (Eq (2.1)), w.e.h.p. in Δ, at most $2|K|/8 \leqslant 3\Delta/8$ vertices get activated. Furthermore, each $L_\varphi(u) \cap L_\varphi(v)$ looses at most $2\varepsilon\Delta$ colors from colored external neighbors. So each edge with both endpoints uncolored contributes at least $(\Delta+1-r)-3\Delta/8-2\varepsilon\Delta \geqslant \Delta/4$ to $\mathrm{Avail}_{\varphi, L_\varphi(K)\setminus[r]}(F)$.

To bound the number of anti-edges with both endpoints unactivated, we may use Lemma 3.17 with $S = K$, with the $a(K)|K|/2$ anti-edges of $G[K]$, but where vertices join the random set with probability $1-p \geqslant 7/8$ instead of p. This gives that w.p. $1-\exp(-(1-p)\frac{a(K)|K|/2}{5|K|}) = 1-\exp(-\Omega(a(K)))$ there are $(1-p)^2 a(K)|K|/4 \geqslant a(K)\Delta/8$ anti-edges with both endpoints unactivated. ∎

4.7. Reducing Put-Aside Sets

As explained in Section 4.1, coloring put-aside vertices is not immediate in BCONGEST. The procedure CompressTry described in Algorithm 15 runs non-adaptative random color trials to reduce the size of the put-aside set to something slightly sublogarithmic, so we can afford to disseminate messages (see Section 4.8, Step 4). This section applies only to the densests almost-cliques, called *full*, where $e(K) + a(K) < \ell = \Theta(\log^{1.1} n)$ (see Section 4.8, Definition 4.27).

Algorithm 15. Procedure CompressTry, in almost-clique $K \in \mathscr{K}_{full}$

Parameters: Let $c = O(1)$ be a large enough constant, $k := \lceil c \log n / \log^2 \log n \rceil$.
Input: On uncolored subset $S \subseteq K \setminus \operatorname{dom}\varphi$ of size $O(\Delta/\log n)$. Each node $v \in S$ has a publicly known list of colors $\mathrm{List}(v)$ of size $\mathrm{poly}(\log n)$ and an $O(\log\log n)$-bit identifier unique within S. For each $v \in S$, let $S_<(v) := \{u \in S : \mathsf{ID}(u) < \mathsf{ID}(v)\}$.

(1) Each $v \in S$ samples k colors $\boldsymbol{\chi}_1(v), \dots, \boldsymbol{\chi}_k(v)$ in $\mathrm{List}_\varphi(v)$, independently and u.a.r., and disseminates them to S by Many-to-All Broadcast (Corollary 4.10).
(2) For each $v \in S$, processed in increasing ID order:
If $\mathbf{X}_v := \{i \in [k] : \boldsymbol{\chi}_i(v) \in \mathrm{List}_\varphi(v) \setminus \varphi(S_v^-)\} \neq \emptyset$

then: v colors itself with $\boldsymbol{\chi}_i(v)$ in φ, where $i = \min \mathbf{X}_v$.
else: v stays uncolored.

Lemma 4.24. *Let $K \in \mathscr{K}_{full}$ and fix a set $S \subseteq K \setminus \operatorname{dom}\varphi$ of size $O(\log^{1.1} n)$ such that every vertex in S has a $O(\log\log n)$-bit identifier unique in S. Furthermore, suppose each $v \in S$ has a list $\operatorname{List}(v)$ of at most $c\log n^{1.1} n$ colors known to every $u \in S$, and such that $|\operatorname{List}_\varphi(v)| \geqslant |S| + z$ for a fixed $z \geqslant c\log n/\log\log n$. Then, w.p. $1 - e^{-z} - 1/\operatorname{poly}(n)$,* CompressTry *colors all but z nodes in S. Furthermore,* CompressTry *uses $O(\log n/\log\log n)$ bandwidth.*

Proof. Let us first argue about the bandwidth of CompressTry (Algorithm 15). The procedure has each node in S send $k = \left\lceil \frac{c\log n}{\log^2\log n} \right\rceil$ colors from publicly known lists of $\operatorname{poly}(\log n)$ colors together its $O(\log\log n)$ ID. Many-to-All broadcast (Corollary 4.10) disseminates these messages to all of S w.h.p. in only $O(1)$ rounds, given that $|S| \leqslant O(\Delta/\log n)$ for $\Delta \geqslant \Omega(\log^3 n)$. Each disseminated message is of size $O(k \cdot \log\log n + \log\log n) = O(\log n/\log\log n)$, giving the claimed bandwidth.

We now argue the success probability of the procedure. Algorithm 15 essentially simulates the following sequential algorithm: nodes of S, in the order of their IDs, each perform k TryColor, coloring themselves with the first successful one. They act as if they were connected, never adopting a color already taken by a node of smaller ID. Colors tried by a node v are sampled independently in a set which does not depend on any other colors tried, so can all be sampled in advance.

This is easily simulated in a distributed setting by CompressTry. Once each node $v \in S$ knows the colors and IDs of all other nodes in S, it can compute the behavior of all the nodes of smaller ID S_v^- as they each pick the first of their tried colors not taken by an earlier node, if it exists. Once v has computed the colors adopted by nodes in S_v^-, it knows whether it can adopt any of its own colors and potentially color itself.

We now bound the probability that more than z nodes in S fail to get colored. Call $\mathbf{F}_{v,i}$ the random vector

$$\mathbf{F}_{v,i} = (\boldsymbol{\chi}_1(S_<(v)), \ldots, \boldsymbol{\chi}_k(S_<(v)), \boldsymbol{\chi}_1(v), \ldots, \boldsymbol{\chi}_{i-1}(v)) \ ,$$

containing all the randomness of the nodes with ID smaller than v and of the first $i-1$ random color trials of v. For each v, regardless of the colors adopted and tried by the nodes of smaller ID — i.e., any fixed value of $\mathbf{F}_{v,i}$ — , we have $|\mathrm{List}_\varphi(v) \setminus \varphi(S_<(v))| \geqslant z$. This means that as an uncolored node v tries its i-th color in the sequential process, regardless of previous TryColor attempts by v or nodes of smaller ID, it always succeeds with probability at least

(4.7)
$$\mathbb{P}[\boldsymbol{\chi}_i(v) \in \mathrm{List}_\varphi(v) \setminus \varphi(S_<(v)) \mid \mathbf{F}_{v,i}] \geqslant \frac{z}{|\mathrm{List}(v)|} \geqslant \frac{1}{\log\log n \cdot \log^{0.1} n} \ .$$

(using $z \geqslant c \log n / \log\log n$ and $|\mathrm{List}(v)| \leqslant c \log^{1.1} n$)

Let $\mathbf{X}_v$ be the random variable indicating if v failed to adopt any color by the end of the process. By the chain rule, Eq (4.7) implies

$$\begin{aligned}
&\mathbb{P}[\mathbf{X}_v = 1 \mid \boldsymbol{\chi}_1(S_<(v)), \ldots, \boldsymbol{\chi}_k(S_<(v))] \\
&\qquad = \prod_{i=1}^{k} \mathbb{P}[\boldsymbol{\chi}_i(v) \notin \mathrm{List}_\varphi(v) \setminus \varphi(S_<(v)) \mid \mathbf{F}_{v,i}] \\
&\qquad \leqslant \left(1 - \frac{1}{\log\log n \cdot \log^{0.1} n}\right)^k \\
\text{(by definition of } k\text{)} \qquad &\qquad \leqslant \exp\left(-\frac{c \log^{0.9} n}{\log^3 \log n}\right) \\
\text{(for } n > 2\text{)} \qquad &\qquad \leqslant \exp\left(-\frac{c \log^{0.1} n}{10}\right) := p \ .
\end{aligned}$$

The expected number of uncolored nodes is $\mathbb{E}[\sum_v \mathbf{X}_v] \leqslant p|S| \leqslant z/4$ for large enough constant c. We get concentration from Chernoff with domination: the probability that more than z nodes fail to adopt a color is at most $\exp(-z)$ (by Eq (2.5)). ∎

4.8. The Full Algorithm

In this section, we describe our algorithm and complete the proof of Theorem 4.1. Algorithm 16 gives a high-level description of our algorithm.

ALGORITHM 16. High-Level Description of Our Algorithm.

Parameters: Let $c = O(1)$ be a large enough constant,

$$\ell = c\log^{1.1} n\ , \quad \varepsilon = 10^{-5} \quad \text{and} \quad \lambda = 401\ . \tag{4.8}$$

(1) **Setting up.** Compute an ε-almost-clique decomposition $V_{sparse}, K_1, \dots, K_t$. Compute outliers O_K and inliers $I_K = K \setminus O_K$ in each almost-clique K (see Definition 4.25), as well as put-aside sets P_K (see Proposition 3.24). We define a value $x(K) = \Theta(a(K) + e(K) + \ell)$ for each almost-clique (see Eq (4.10)). By extension, let $x(v) = x(K)$ for each $v \in K$.

Almost-cliques are categorized as full, open, or closed (Definition 4.27). The following three steps aim at generating slack for each type:

(a) SLACKGENERATION with $r = 400\varepsilon\Delta$ and $p = 1/200$.
(b) COLORFULMATCHING with $r = 400\varepsilon\Delta$ and λ as in Eq (4.8).
(c) COMPUTEPUTASIDESETS of size 201ℓ in every $K \in \mathscr{K}_{full}$.

Each sparse node has $\Omega(\Delta)$ permanent slack from the slack generation step; hence, we color them with colors $[\Delta + 1]$ in $O(\log^* n)$ rounds with MULTICOLORTRIAL. We color outliers O_K with colors from $[\Delta + 1] \setminus [x(K)]$ with MULTICOLORTRIAL using the $\Omega(\Delta)$ temporary slack provided by inactive inliers.

(2) **Synchronized Color Trial.** In each almost-clique, we compute the clique palette $L_\varphi(K)$ and sample a random permutation π of $K \setminus (\operatorname{dom}\varphi \cup P_K)$ using PERMUTE from Section 4.5. Each $v \in K \setminus (\operatorname{dom}\varphi \cup P_K)$ with $\pi(v) \leqslant |L_\varphi(K)| - x(K)$ tries the $\pi(v)$-th color of $L_\varphi(K) \setminus [x(K)]$. If $\pi(v) > |L_\varphi(K)| - x(K)$, then v remains uncolored.

(3) **Completing the Coloring.** Uncolored nodes of $H = G\left[\bigcup_{K \in \mathcal{K}_{full} \cup \mathcal{K}_{closed}} K \setminus P_K\right]$ now have ample available reserved colors, i.e., they satisfy

$$|[x(v)] \cap L_\varphi(v)| \geqslant \deg_\varphi(v; H) + x(v)/10 \ .$$

Hence, inliers of full and closed almost-cliques (apart from put-aside nodes) are colored in $O(\log^* n)$ rounds by MultiColorTrial with colors $[x(v)]$.

In open almost-cliques, the coloring process has four steps. Throughout B is the set of uncolored vertices $v \in K \in \mathcal{K}_{open}$ with many available non-reserved colors, namely such that $|L_\varphi(v) \setminus [x(K)]| > \gamma_{4.32} \cdot e(K)$, where $\gamma_{4.32}$ is a universal constant described in Lemma 4.32.

- (*i*) Repeat $O(1)$ times: vertices of B call TryColor using colors $L_\varphi(v) \setminus [x(v)]$.
- (*ii*) Run MultiColorTrial in $G[B]$ in $O(\log^* n)$ rounds using colors $[x(v)]$.
- (*iii*) Uncolored nodes run $O(1)$ rounds of TryColor from the palettes $L_\varphi(v)$.
- (*iv*) Uncolored nodes in open almost-cliques run MultiColorTrial using colors $[x(v)]$.

(4) **Coloring Put-Aside Sets.** We color put-aside sets in two steps: first, we reduce their size to $O(\log n / \log\log n)$ by running *non-adaptive* randomized color trial. Then, each node broadcasts $|P_K|$ colors from a $\text{poly}(\log n)$-sized set of colors to all other uncolored nodes. This uses $O(\log n / \log\log n) \times O(\log\log n) = O(\log n)$ bits and can be completed in $O(1)$.

The key technical idea is to *reserve* colors $\{1, 2, \ldots, x(K)\}$ in each clique, where $x(K)$ is an integer that depends on the density of K (see Eq (4.10)). It is straightforward to see that reserved colors $[x(K)]$ are not used during Steps 1 and 2. After Step 2, in full and closed almost-cliques, the value of $x(K)$ is chosen to be greater than nodes' uncolored degrees. This allows using lists $\text{List}(v) := [x(v)]$ for the MultiColorTrial in

Step 3. In open almost-cliques, additional work is needed to reduce uncolored degrees without using too many reserved colors.

4.8.1. Step 1: Setting up. Assume we have an ε-almost-clique decomposition V_{sparse}, K_1, ..., K_t (see Proposition 4.2). Sparse nodes can be colored in $O(\log^* n)$ rounds, so we focus our attention on almost-cliques. We call outliers the (possibly empty) set of nodes in each almost-clique whose external degree or anti-degree derives more than a constant factor from the average.

Definition 4.25 (Inliers/Outliers)**.** For each K, we define its set of *outliers* as

$$O_K = \{v \in K : e(v) \geqslant 30e(K) \text{ or } a(v) \geqslant 30a(K)\} \ . \tag{4.9}$$

We call the remaining *uncolored* nodes $I_K = K \setminus (\operatorname{dom}\varphi \cup O_K)$ *inliers*.

In each almost-clique, outliers represent only a small fraction of the vertices; hence, they can be colored beforehand with the temporary slack provided by their $\Omega(\Delta)$ uncolored neighbors in I_K. Importantly, SlackGeneration and ColorfulMatching color only a small constant fraction of the vertices.

Claim 4.26. *After Step 1, w.h.p., each K has $|I_K| \geqslant 0.9\Delta$.*

Proof. By Markov inequality, outliers represent at most a $1/15$ fraction of K. Furthermore, nodes get colored during slack generation w.p. at most $p = 1/200$ (see Algorithm 2). By Chernoff, w.h.p., at most a $1/100$ fraction of K gets colored. The colorful matching comprises[2] $2\lambda a(K) \leqslant 10^3\varepsilon\Delta \leqslant \Delta/100$ nodes by our choice of ε (Eq (4.8)). Therefore, $|I_K| \geqslant (1 - 1/15 - 1/100 - 1/100)|K| \geqslant 0.9\Delta$. ∎

We classify almost-cliques in three categories, depending on the degree that nodes have after Step 2. Each type of almost-clique receives slack

[2] Algorithm 14 colors vertices only if they provide slack, i.e., if their color is added to the colorful matching, and only two vertices per such color. So if the matching is larger than intended, it suffices to learn which colors were given by Algorithm 14 (e.g., using Lemma 4.11) and uncolor enough vertices

from different sources: full almost-cliques from put-aside sets, open almost-cliques from the slack generation step, and closed almost-cliques from the colorful matching.

Definition 4.27 (Full/Open/Closed Cliques)**.** For each $i \in [t]$, we say that $K = K_i$ is:

- *full* if $a(K) + e(K) < \ell$, where ℓ is defined in Eq (4.8),
- *open* if K is not full and $2a(K) < e(K)$, and
- *closed* if K is neither full nor open.

We denote by $\mathscr{K}_{full}$ (respectively $\mathscr{K}_{open}$ and $\mathscr{K}_{closed}$) the set of full almost-cliques (respectively open and closed almost-cliques).

In each clique, we reserve $x(K)$ colors depending on the clique's density. We will ensure that $[x(K)] \subseteq L_\varphi(K)$ until we color inliers with MultiColorTrial (Step 3). For an almost-clique K, define

$$x(K) = \begin{cases} 200\ell & \text{if } K \in \mathscr{K}_{full} \\ 400a(K) & \text{if } K \in \mathscr{K}_{closed} \\ 400e(K) & \text{if } K \in \mathscr{K}_{open} \end{cases}. \tag{4.10}$$

By extension, we write $x(v) = x(K)$ for each $v \in K$.

Put-Aside Sets. Recall that to color in $O(\log^* n)$ rounds with MultiColorTrial, nodes need slack at least $\ell = \Theta(\log^{1.1} n)$ (Lemma 4.8 where $\kappa = 1/10$). Nodes from very dense almost-cliques do not receive enough permanent slack from the slack generation phase. Following Chapter 3, we overcome this issue by putting aside a set $P_K \subseteq K$ of $\Theta(\ell)$ nodes in each full almost-clique to provide temporary slack. By using repeated applications of LowDegreeSample, as described in [HKNT22, Appendix C], we obtain the same guarantees as in Proposition 3.24 but only need that $\Delta \gg \ell^2 = O(\log^{2.2} n)$. These sets remain uncolored until the very end of the algorithm. These are necessary only in full almost-cliques, whose nodes have $O(\ell)$ external neighbors. By Proposition 3.24, we can find sets $P_K \subseteq K$ of 201ℓ vertices for each $K \in \mathscr{K}_{full}$ in $O(1)$ rounds such that no edge connects put-aside sets from different almost-cliques.

4.8.2. Step 2: Synchronized Color Trial. We assume that vertices run Algorithm 5 as described in Chapter 3. By Lemma 4.14, PERMUTE samples in $O(1)$ rounds a permutations for which Lemma 3.25 fails with probability inflated by a factor a most two.

Lemma 4.28 shows that full and closed almost-cliques have enough colors, even when we reserve $x(K)$ colors. It follows from a careful adjustment of constants in the size of put-aside sets and colorful matching.

Lemma 4.28. *For all $K \in \mathscr{K}_{full} \cup \mathscr{K}_{closed}$, we have that*

$$|L_\varphi(K)| - x(K) \geqslant |K \setminus (\operatorname{dom}\varphi \cup P_K)| \ .$$

PROOF. We consider each type of almost-clique separately. In a full almost-clique K, recall that we computed a set P_K of size $201\ell = \Theta(\log^{1.1} n)$ vertices that remain uncolored. The set S of nodes that participate in the synchronized color trial is $|S| = |K \setminus (\operatorname{dom}\varphi \cup P_K)| \geqslant 0.75\Delta$ (by of Claim 4.26 and $\Delta \gg \ell$). The number of colors used in K is bounded by the number of colored nodes; hence, $|L_\varphi(K)| \geqslant \Delta - (|K| - |K \setminus \operatorname{dom}\varphi|)$. Since each full almost-clique has size at most $\Delta + \ell$, we infer $|L_\varphi(K)| \geqslant |K \setminus \operatorname{dom}\varphi| - \ell$. Put-aside sets have size $|P_K| = 201\ell$ and are uncolored, so

$$\begin{aligned} |K \setminus (\operatorname{dom}\varphi \cup P_K)| &= |K \setminus \operatorname{dom}\varphi| - 201\ell \\ &\leqslant |L_\varphi(K)| - 200\ell = |L_\varphi(K)| - x(v) \ . \end{aligned}$$

Suppose now that K is closed. Denote by t the number of nodes colored during the slack generation step or as outliers. In closed almost-clique, we compute a colorful matching of size $M \geqslant \lambda a(K)$. Hence $|L_\varphi(K)| \geqslant \Delta - t - M$. On the other hand, each color used by the colorful matching is repeated, thus used by at least two nodes. Therefore, the number of uncolored nodes is

$$\begin{aligned} |K \setminus \operatorname{dom}\varphi| &\leqslant |K| - t - 2M \\ \text{(because } |K| \leqslant \Delta + a(K)\text{)} \qquad &\leqslant (\Delta - t - M) - M + a(K) \\ \text{(because } M \geqslant \lambda a(K) = x(K) + a(K)\text{, Eq (4.8))} \qquad & \\ &\leqslant |L_\varphi(K)| - x(K) \ . \end{aligned}$$

∎

Corollary 4.29. *After Step 2, w.h.p., every $K \in \mathscr{K}_{full} \cup \mathscr{K}_{closed}$ contains at most $48\max\{\ell, e(K)\}$ uncolored non-put-aside vertices.*

PROOF. In Step 2, we run Algorithm 5 with $S = K \setminus P_K$ and $\mathscr{C} = L_\varphi(K) \setminus [x(K)]$ in every K. By Lemma 4.28, we have that $|S| = |K \setminus (\operatorname{dom}\varphi \cup P_K)| \geqslant 0.9\Delta - |P_K| \geqslant 0.95\Delta \geqslant 0.8|K|$ colors. By Lemma 3.25 with $\alpha = 0.8$, the number of uncolored vertices in $K \setminus P_K$ for each $K \in \mathscr{K}_{full} \cup \mathscr{K}_{closed}$ is at most $8\alpha^{-1}e(K) + O(\log n) \leqslant 48 \max\{\ell, e(K)\}$ since $\ell \gg \log n$. ∎

In open almost-cliques, Lemma 4.28 does not hold.

Lemma 4.30. *After Step 2, w.h.p., each $K \in \mathscr{K}_{open}$ has at most $450e(K)$ uncolored vertices.*

PROOF. In open almost-cliques K, there are no put-aside vertices and Algorithm 5 in K uses colors $\mathscr{C} = L_\varphi(K) \setminus [x(K)]$. Because the colorful matching is larger than any uncolored vertices' anti-degree, Lemma 4.19 implies that $|L_\varphi(K)| \geqslant |K \setminus \operatorname{dom}\varphi|$, so that $|\mathscr{C}| = |L_\varphi(K)| - x(K) \geqslant |K \setminus \operatorname{dom}\varphi| - x(K)$. Since $x(K) \leqslant 400\varepsilon|K|$, we also have that $|\mathscr{C}| \geqslant (0.9 - 400\varepsilon)|K| \geqslant 0.75\Delta \geqslant |K|/2$. By Lemma 3.25 with $\alpha = 1/2$ and $S \subseteq K \setminus \operatorname{dom}\varphi$ any set of $|L_\varphi(K)| - x(K)$ uncolored vertices, w.h.p., at most $x(K) + 8\alpha^{-1}e(K) + O(\log n) \leqslant 450e(K)$ vertices remain uncolored in K after the synchronized color trial. ∎

4.8.3. Step 3: Completing the Coloring. Let H be the graph induced by inliers from full and closed almost-cliques that are not put-aside nodes. The following lemma implies that after SYNCHRONIZEDCOLORTRIAL, the nodes of H satisfy the assumptions of Lemma 4.8 and can therefore be colored by SLACKCOLOR in $O(\log^* n)$ rounds with lists $\text{List}(v) = [x(v)]$.

Lemma 4.31. *After Step 2, w.h.p., each $v \in H$ satisfies*

$$|[x(v)] \cap L_\varphi(v)| \geqslant \deg_\varphi(v, H) + x(v)/10 \ .$$

PROOF. First, we bound the uncolored degrees in H after Step 2 as $\deg_\varphi(v) \leqslant 0.9x(v) - e(v)$. As we shall explain next, this implies Part ii) of Lemma 4.8.

Let $v \in K$ and assume first $K \in \mathscr{K}_{full}$. Since only inliers remain to be colored, $e(v) \leqslant 30e(K) \leqslant 30\ell$ (by Eq (4.9)) and after the synchronized color trial, w.h.p., at most 48ℓ nodes in $K \setminus P_K$, i.e., in $K \cap H$, remain

uncolored (by Corollary 4.29). Overall, $\deg_\varphi(v,H) \leqslant 80\ell$ and $\deg_\varphi(v,H) + e(v) \leqslant 110\ell \leqslant 0.9x(v)$ (by Eq (4.10)). If $K \in \mathscr{K}_{closed}$, then $\deg_\varphi(v,H) \leqslant 80e(K) \leqslant 160a(K)$ because $e(K) \leqslant 2a(K)$. Hence, $\deg_\varphi(v,H) + e(v) \leqslant (160+60)a(K) \leqslant 0.9x(v)$.

Observe that, since $x(v)$ has the same value for each $v \in K$, and colors from $[x(K)]$ are not used to color nodes of K in Steps 1 and 2, the only reason a color $\chi \in [x(v)]$ might not belong to $L_\varphi(v)$ is if it is used by an external neighbor of v. Thus, for all $v \in H$,

$$|[x(v)] \cap L_\varphi(v)| \geqslant x(v) - e(v) \geqslant \deg_\varphi(v,H) + x(v)/10 \ . \qquad \blacksquare$$

We henceforth assume nodes of H are colored. The remainder of this section focuses on *open almost-cliques*, where nodes receive too little slack to be colored immediately after SynchronizedColorTrial. Lemma 4.32 analyses Steps (i) and (ii) of Step 3 in Algorithm 16. In these steps, we color nodes with too little slack in $[x(v)]$. We guarantee that the remaining uncolored nodes can be colored later through Eq (4.11).

Lemma 4.32. *There exists a universal constant $\gamma_{4.32} = \Theta(\varepsilon)$ for which the following holds. At every step after Step 3-(ii), w.h.p., uncolored nodes in* open almost-cliques *verify*

$$|[x(v)] \cap L_\varphi(v)| > \deg_\varphi(v) + \gamma_{4.32} \cdot e(K) \ . \tag{4.11}$$

Before proving Lemma 4.32, recall that trying random colors decreases uncolored degrees by a constant factor w.h.p. as long as uncolored degrees are larger than $\Theta(\log n)$ and available colors represent a constant fraction of uncolored degrees. Formally:

Lemma 4.33. *Let H' be a vertex-induced subgraph of H and $\mathrm{List}(v) \subseteq [\Delta+1]$ a list of colors for each v. Suppose there exists a globally known constant $\alpha \in (0,1]$ such that every uncolored v satisfies $|\mathrm{List}_\varphi(v)| = |\mathrm{List}(v) \setminus \varphi(N(v))| \geqslant \max\{\alpha \cdot \deg_\varphi(v), c\log n\}$. If nodes of H independently call* TryColor *w.p. $p_t = \alpha/3$ — i.e., try a uniform color $\boldsymbol{\chi}(v) \in \mathrm{List}_\varphi(v)$ and retain it if $\boldsymbol{\chi}(v) \notin \boldsymbol{\chi}(N_<(v))$ —, then, w.p. $1 - n^{-\Theta(c)}$, the uncolored degree of every node in H' has decreased by a factor $(1-\alpha/36)$ or is smaller than $\alpha^{-1} \cdot c\log n$.*

PROOF. Say a vertex is active if it calls TRYCOLOR. The expected number of activated neighbors of v is $p_t \deg_\varphi(v, H') \leqslant |\operatorname{List}_\varphi(v)|/3$. Since each vertex becomes active independently, the classic Chernoff Bound (Eq (2.1) with $\delta = 1$ and $\mu_{\mathsf{H}} = |\operatorname{List}_\varphi(v)|/3$) implies that, w.p. $1 - n^{-\Theta(c)}$, each vertex v has at most $2|\operatorname{List}_\varphi(v)|/3$ active neighbors. Similarly, vertices with $\deg_\varphi(v, H') \geqslant \alpha^{-1} \cdot c \log n$ have at least $(\alpha/6) \deg_\varphi(v, H')$ active neighbors w.p. $1 - n^{-\Theta(c)}$. Condition on both bounds henceforth.

Let v be a vertex with $\deg_\varphi(v, H') \geqslant \alpha^{-1} \cdot c \log n$. Let u be an active neighbors of v. Observe that, given our conditioning on activations, u gets colored with probability at least $1/3$ because its active neighbors block at most $2/3$ of its available colors in $\operatorname{List}(u)$. Thus, the expected number of neighbors of v to get colored is at least $(\alpha/18) \deg_\varphi(v, H')$. By giving priority to nodes of smaller identifier — i.e., a vertex retains its color if it was not sampled by nodes of smaller identifiers — we can apply the Chernoff Bound with stochastic domination (Lemma 2.3) and obtain that w.h.p. at least $(\alpha/36) \deg_\varphi(v, H')$ neighbors of v get colored. As such, its degree has decreased at least by a $(1 - \alpha/36)$ factor. ∎

We can now prove that after Step 3-(*ii*) the uncolored nodes in open almost-cliques verify Eq (4.11).

PROOF OF LEMMA 4.32. After SYNCHRONIZEDCOLORTRIAL in an open almost-clique K, at most $450e(K)$ nodes remain uncolored in K (Lemma 4.30). Adding their at most $30e(K)$ external neighbors, the uncolored degree of each node is at most $500e(K)$. Let $\gamma_{sg} = \Theta(\varepsilon)$ be the universal constant such that every dense vertex with $e(v) \geqslant \gamma_{sg}^{-1} \cdot c \log n$ received $\gamma_{sg} \cdot e(v)$ slack after SLACKGENERATION (see Lemma 3.18 and Part 3 in Proposition 3.6). The universal constant in Eq (4.11) is defined as $\gamma_{4.32} = \gamma_{sg}/8$. Recall that we defined B as the set of uncolored vertices $v \in K \in \mathscr{K}_{open}$ such that $|L_\varphi(v) \setminus [x(v)]| \geqslant \gamma_{4.32} \cdot e(K)$. Importantly, once a vertex exited B, it has $|L_\varphi(v) \setminus [x(v)]| < \gamma_{4.32} \cdot e(K)$ and this remains true as we extend the coloring. In other words, B can only loose vertices.

Let $\operatorname{List}(v) = [\Delta + 1] \setminus [x(v)]$ and $\alpha = \gamma_{4.32}/500$, such that $|\operatorname{List}_\varphi(v)| = |L_\varphi(v) \setminus [x(v)]| \geqslant \alpha \deg_\varphi(v)$ for each $v \in B$. In Step 3-(*i*), vertices of B repeat the following for $T = \frac{36}{\alpha} \ln(500) = O(1)$ iterations: with probability

$p_t = \alpha/3$, call TryColor using lists $\text{List}_\varphi(v)$, i.e., *without using reserved colors*. By Lemma 4.33, w.h.p., each call reduces the uncolored degree in $G[B]$ by a $(1-\alpha/36)$ factor. Hence, by repeating this process T times, w.h.p., the maximum uncolored degree of vertices in $B \cap K$ is at most $(1-\alpha/36)^T \cdot 500e(K) \leqslant e(K)$ (note that in open almost-cliques, we have $e(K) \geqslant \alpha^{-1} \cdot c \log n$). Since none of the reserved colors was used in K up to this point, reserved colors can only be blocked by external neighbors. Thus, nodes of B have

$$|[x(v)] \cap L_\varphi(v)| \geqslant x(v) - e(v) \geqslant 300e(K_v) \geqslant \deg_\varphi(v, G[B]) + x(v)/10 \ .$$

MultiColorTrial in Step 3-(ii) with $\text{List}(v) = [x(v)]$ therefore colors all nodes of B in $O(\log^* n)$ rounds with high probability (Lemma 4.8).

We now prove that the remaining uncolored vertices verify Eq (4.11). Recall that in open cliques, vertices receive slack from slack generation. To deal with vertices of small external degree, we use the following technical claim:

Claim 4.34. *For every $v \in K \in \mathscr{K}_{open}$, we have*

$$|L_\varphi(v)| \geqslant \deg_\varphi(v) + (\gamma_{sg}/4) \cdot e(K) \ .$$

Proof. There are two cases depending on if $e(v)$ is large compared to $e(K)$ or not.

When $e(v) < e(K)/4$. For each $v \in K \in \mathscr{K}_{open}$, we have $\Delta - \deg(v) + e(v) \geqslant e(K)/2$. That is because for all $v \in K$, we have $\Delta+1 \geqslant \deg(v)+1 = |K \cap N(v)| + e(v) + 1 = |K| + e(v) - a(v)$. Thus, on average $\Delta + 1 \geqslant |K| + e(K) - a(K) \geqslant |K| + e(K)/2$. We deduce $\Delta - \deg(v) \geqslant (|K| + e(K)/2) - (|K| + e(v)) \geqslant e(K)/2 - e(v) \geqslant e(K)/4$, where the last inequality uses the assumption on $e(v)$. This conclude this case since v always has $\deg_\varphi(v) + (\Delta - \deg(v)) \geqslant \deg_\varphi(v) + e(K)/4$ colors available and $\gamma_{sg} < 1$.

When $e(v) > e(K)/4$. By Part 3 in Proposition 3.6, the vertex v is $\Omega(\varepsilon e(v))$-sparse. Since $e(v) \geqslant e(K)/4 \geqslant \ell/8$ from K being open, Proposition 3.15 implies that, w.h.p., for all such v we have $|L_\varphi(v)| \geqslant \deg_\varphi(v) + \gamma_{sg} \cdot e(v)$ after SlackGeneration. This remains true as we extend φ and thus, we always have that $|L_\varphi(v)| \geqslant \deg_\varphi(v) + \gamma_{sg} \cdot e(v) \geqslant \deg_\varphi(v) + \gamma_{sg}/4 \cdot e(K)$.

∎

Eq (4.11) follows because every uncolored vertex has at least $\gamma_{sg}/4 \cdot e(K)$ colors available while vertices not in B can only have few in $L_\varphi(v) \setminus [x(v)]$. Formally, Eq (4.11) is derived as

$$\begin{aligned}
|[x(v)] \cap L_\varphi(v)| &= |L_\varphi(v)| - |L_\varphi(v) \setminus [x(v)]| \\
(v \notin B) \qquad &\geqslant |L_\varphi(v)| - \gamma_{4.32} \cdot e(K) \\
(\text{Claim 4.34}) \qquad &\geqslant \deg_\varphi(v) + \gamma_{sg}/4 \cdot e(K) - \gamma_{4.32} \cdot e(K) \\
(\text{setting } \gamma_{4.32} = \gamma_{sg}/8) \qquad &\geqslant \deg_\varphi(v) + \gamma_{4.32} \cdot e(K) \ . \qquad \blacksquare
\end{aligned}$$

Step 3-$(iii), (iv)$ completes the coloring of open almost-cliques. We conclude the analysis of Step 3 by showing that only put-aside sets are left to color.

Lemma 4.35. *After Step 3, w.h.p., the only uncolored nodes are in put-aside sets.*

Proof. By Lemma 4.31, nodes from full and closed almost-cliques that are not in put-aside sets are colored at the beginning of Step 3 using MultiColorTrial with lists $\text{List}(v) = [x(v)]$. In open almost-cliques, by Lemma 4.32, all uncolored nodes verify Eq (4.11) after Step 3-(ii).

Eq (4.11) implies Part ii) in Proposition 4.5 with $\text{List}(v) = [x(v)]$ and $s(v) = \gamma_{4.32} \cdot e(K)$. It thereby allows to run SlackColor (recall that $x(v) = \Theta(e(K))$ and $e(K) \geqslant \ell/4$) and color remaining vertices in $O(\log^* n)$ rounds with high probability. Hence, at the end of Step 3, the only uncolored nodes are put-aside sets in full almost-cliques. $\blacksquare$

4.8.4. Step 4: Coloring Put-Aside Sets. Our goal, in this section, is to reduce the size of put-aside sets to $O(\log n / \log\log n)$. Once this is achieved, coloring their remaining nodes only takes $O(1)$ rounds. In almost-cliques where $a(K) \geqslant c \log n$, we computed a colorful matching w.h.p. and can use the clique palette as proxy for the vertices' palettes. For the remaining almost-cliques, we can use Many-to-All broadcast to learn colors of anti-neighbors.

Claim 4.36. *Let $K \in \mathscr{K}_{full}$ such that $a(K) \leqslant c \log n$ and $S \subseteq P_K$ a set of $O(\log n)$ put-aside vertices. Every vertex in K can learn the set $\{A(u), \varphi(A(u)) : u \in S\}$ in $O(1)$ rounds of* BCONGEST.

PROOF. Since $a(K) < c \log n$, each node has at most $30c \log n$ anti-neighbors in the almost-clique. If we relabel nodes of S using identifiers in $[|S|]$ (e.g., by broadcasting their IDs with Corollary 4.10), every $u \in K$ can describe the set $S \setminus N(u)$ with a bit-map in one $O(\log n)$-bit message. Note that only $O(\log^2 n)$ nodes will need to send a bit-map, i.e. at most $O(\log n)$ anti-neighbors per node in S. By Corollary 4.10, all messages can be disseminated in $O(1)$ rounds to all nodes in K. Every vertex thereby learns $\{A(u) : u \in S\}$. The colors $\{\varphi(A(u)) : u \in S\}$ can be learned by sending $\varphi(u)$ along with the bitmap describing $S \setminus N(u)$. ∎

Let us first explain how we color put-aside vertices with Many-to-All broadcast once their uncolored degrees are $O(\log n / \log \log n)$.

Lemma 4.37. *Suppose all $S_K \subseteq P_K$ is a set of $O(\log n)$ uncolored nodes in each $K \in \mathscr{K}_{full}$ such that the maximum uncolored degree in S_K is $O(\log n / \log \log n)$. Then, w.h.p. we can extend the coloring to every vertex in S_K in $O(1)$ rounds of* BCONGEST.

PROOF. Recall that no edge exists between put-aside sets. Hence, we color each S_K independently.

If $a(K) \geqslant c \log n$, the almost-clique contains a colorful matching of size at least $a(v)$ for every $v \in P_K$, thus, by Lemma 4.19, the clique palette has enough colors for every node, i.e., $|L_\varphi(K) \cap L_\varphi(v)| \geqslant |P_K|$. By Lemma 4.11, vertices learn $L_\varphi(K)$ in $O(1)$ rounds and find a coloring by running the greedy coloring algorithm assuming every vertex of S_K is adjacent (every color is used at most once).

If $a(K) < c \log n$, we can assume without loss of generality that $|L_\varphi(K)| = O(\log^3 n)$. Indeed, if $L_\varphi(K)$ is larger, the vertices learn in $O(1)$ rounds the set $D \subseteq L_\varphi(K)$ of the $O(\log^3 n)$ smallest colors in the clique palette. They each have $|S_K|$ colors available in D because they have at most $O(\log^{1.1} n) \ll |D|$ external neighbors. And so they can be colored as when $a(K) \geqslant c \log n$ with D in place of $L_\varphi(K)$.

We therefore assume $a(K) \leqslant c \log n$ and $|L_\varphi(K)| \leqslant O(\log^3 n)$. By Claim 4.36, every vertex learns $A(v)$ and $\varphi(A(v))$ for every $v \in P_K$ in $O(1)$ rounds. In particular, every vertex of K knows the neighbors of v in P_K (since it knows $A(v)$) and every $v \in P_K$ can describe a list of $|N(v) \cap S_K|$

colors from its palette in a single $O(\log n)$-bit message: by listing colors of $L_\varphi(K) \cup \varphi(A(v))$ which are available. Since we assumed $|N(v) \cap S_K| = O(\log n/\log\log n)$ and each color is described in $O(\log\log n)$ bits (the clique palette and $\varphi(A(v))$ are known to all in K), this fits in a single $O(\log n)$-bit message. All nodes can now compute locally the same coloring of S_K without additional communication, e.g, by simulating a greedy sequential algorithm with the lists. ∎

The key difficulty in coloring put-aside sets lies in reducing their sizes to $O(\log n/\log\log n)$. We use the procedure CompressTry introduced in Section 4.7, which simulates a sequential algorithm where nodes of the put-aside set, in the order of their IDs, each perform $O(\log n/\log\log n)$ times a *non-adaptive* TryColor with slack z.

Lemma 4.38 shows how we use CompressTry to reduce the size of the put-aside sets. In almost-cliques with colorful matching, nodes have $a(K) \geqslant \Omega(\log n)$ slack; CompressTry directly reduces P_K to $O(\log n/\log\log n)$ nodes by using the clique palette. In almost-cliques where $a(K) < c\log n$, we first put-aside $O(\log n)$ nodes to reduce P_K to $O(\log n)$ using the clique palette. Then, nodes add colors used by their anti-neighbors to their list, and CompressTry finishes to reduce P_K to $O(\log n/\log\log n)$.

Lemma 4.38. *There is a $O(1)$-round* BCONGEST *algorithm reducing the number of uncolored nodes in P_K to $O(\log n/\log\log n)$ with high probability.*

Proof. For almost-cliques such that $a(K) \geqslant c\log n$, Lemma 4.24 allows us to directly reduce P_K to a set of size $z := c\log n/\log\log n$. This is because, in such almost-cliques, we compute a colorful matching of size $\lambda a(K) \geqslant a(K) + a(v)$, for each $v \in P_K$ (which are inliers). Therefore, when using the clique palette as list $\text{List}(v) = L_\varphi(K)$ for every vertex, by Lemma 4.19, ensures that $|\text{List}_\varphi(v)| = |L_\varphi(K) \cap L_\varphi(v)| \geqslant |P_K| + a(K) \geqslant |P_K| + z$. Note that the clique palette can be publicly learned in $O(1)$ rounds by Lemma 4.11 and if $L_\varphi(K)$ is larger than $c\log^{1.1} n > e(v) + |P_K| + z = \Theta(\log^{1.1} n)$, we can simply choose the $c\log^{1.1} n$ smallest colors of $L_\varphi(K)$. CompressTry succeeds only w.p. $1 - e^{-z}$, but by repeating independently $\log\log n$ times, the probability that at least one instance succeeds is $1 - e^{-z\log\log n} + 1/\text{poly}(n) = 1 - n^{-c}$. Overall, we

need $\log\log n \times O(\log n/\log\log n) = O(\log n)$ bandwidth. To detect the successful instance, we aggregate on a depth-2 BFS tree the number of uncolored vertices left in every instance. Since each instance contain at most $O(\log^{1.1} n)$ vertices, it takes $O(\log\log n)$ bandwidth per instance, thus $O(\log n)$ bandwidth overall.

Henceforth, we assume that $a(K) < c\log n$. The main difference is that we do not have a colorful matching, so the clique palette does not approximate $L_\varphi(v)$ well. We settle this in two steps.

From $O(\log^{1.1} n)$ to $O(\log n)$. Let $S \subseteq P_K$ be an arbitrary subset of P_K of $31c\log n$ nodes. By Lemma 4.19, $|L_\varphi(K) \cap L_\varphi(v)| \geqslant |P_K| - a(v) \geqslant |P_K \setminus S| + c\log n$. Therefore, CompressTry with the clique palette $L_\varphi(K)$ as list for every vertex and $z = c\log n$ reduces P_K w.h.p. to size $32c\log n$ (the $c\log n$ nodes left uncolored in $P_K \setminus S$ by CompressTry and the $31c\log n$ uncolored nodes of S).

From $O(\log n)$ to $O(\log n/\log\log n)$. We may assume vertices have at least $2z$ uncolored neighbors in P_K, where $z = c\log n/\log\log n$, since we can first color all vertices with at most $2z = O(\log n/\log\log n)$ neighbors in $O(1)$ rounds using Lemma 4.37. Instead of using only the clique palette, we augment lists with colors of anti-neighbors. Let $\text{List}(v) = L_\varphi(K) \cup \varphi(A(v))$ be the list for vertex v. Since every external neighbor of $v \in P_K$ is colored, the colors available for v are all included in its list: $L_\varphi(v) \subseteq \text{List}(v)$. In particular, it contains at least $2z$ colors — since that it how many uncolored neighbors v has. If we now put-aside a set $S \subseteq P_K$ of z nodes, the lists contain $|P_K \setminus S| + z$ colors available and, by[3] Lemma 4.24, the set $P_K \setminus S$ contains $O(\log n/\log\log n)$ uncolored vertices with probability at least $1 - e^{-z}$. Running this $O(\log\log n)$ times in parallel achieves a $1-1/\text{poly}(n)$ probability of success while still using $O(\log n)$-bit bandwidth. To conclude, observe that vertices learn the colors of $\varphi(A(v))$ by Claim 4.36 in $O(1)$ rounds and detect the successful instance by simple convergecast on a depth-2 tree. ∎

4.8.5. Proof of Theorem 4.1. By Proposition 4.2, we can compute the almost-clique decomposition in $O(1)$ rounds. By aggregation on a

[3] if the set is larger than $c\log^{1.1} n \geqslant e(v) + |P_K| + z$, we can simply pick any subset of this size

depth-2 BFS tree, nodes in each almost-clique can count $a(K)$ and $e(K)$, thus know to which category their almost-clique belongs to, as well a their value of $x(K)$. Then, with w.p. p_s every node decides independently to try a color in $[\Delta+1] \setminus [x(v)]$ (for consistency, let $x(v) = 0$ for all $v \in V_{sparse}$). Finally, in each almost-clique with $a(K) \geqslant c \log n$, we compute a colorful matching of size at least $\lambda a(K)$ (by Proposition 4.20). By Proposition 3.24, we compute put-aside sets P_K in $O(1)$ rounds.

Sparse Nodes & Outliers. Each $v \in V_{sparse}$ has permanent slack $\Omega(\Delta)$ (by Lemma 3.18 and because they are $\Omega(\Delta)$-sparse). Hence, we color V_{sparse} in $O(\log^* n)$ rounds of MultiColorTrial (by Lemma 4.8 with $\text{List}(v) = [\Delta+1]$). Since nodes know $a(K)$ and $e(K)$, they can tell if they are outliers. Outliers have $(0.9-\varepsilon)\Delta \geqslant \Delta/2$ available colors from their inactive inliers neighbors (Claim 4.26). Contrary to sparse nodes, we must avoid coloring outliers of K with colors from $[x(K)]$. By definition $x(K) = 10^3\varepsilon\Delta$ (Eq (4.10)); by our choice of ε, outliers have slack $(1/2 - 10^3\varepsilon)\Delta \geqslant \Delta/3$ even when trying colors from $\text{List}(v) = [\Delta+1] \setminus [x(v)]$. By Lemma 4.8, outliers are colored in $O(\log^* n)$ rounds with high probability.

Inliers. Henceforth, we condition on the success of Steps 1 and 2 in every clique. By Lemma 4.35, after Step 3, w.h.p., the only put-aside sets remain to color. By Lemmas 4.37 and 4.38, we can color put-aside sets in $O(1)$ rounds. ∎

CHAPTER 5

Sub-Logarithmic Coloring in Streaming-Congest

To illustrate the robustness of our BCONGEST algorithm, we implement it in a model where vertices have $\text{poly}(\log n)$ memory. Since vertices broadcast one $O(\log n)$-bit message each round, what might exceed our memory requirement is processing the messages received from neighbors when $\Delta \gg \text{poly}(\log n)$. Inspired by the semi-streaming model, we propose the following definition.

Definition 5.1. We define BC-Stream to be the BCONGEST model in which, per round, each node receives the messages from its neighbors in a streaming fashion, using $O(\log^c n)$ memory for some fixed $c > 0$.

Note that results in BC-Stream constrain the size of the messages more than equivalent results in CONGEST or BCONGEST. In the latter models, the size of the messages can be freely changed between $c \log n$ and $c' \log n$ for two positive constants c and c' without changing asymptotic complexities by more than a multiplicative constant factor. This is because, without a memory constraint, for $c > c' > 0$, nodes can simulate an algorithm using $c \log n$-bit messages by buffering the $c' \log n$-bit messages received from each neighbor over $\lceil c/c' \rceil$ rounds. Such buffering uses $\Theta(\Delta \log n)$ memory and is impossible in BC-Stream when $\Delta \gg \text{poly}(\log n)$. In BC-Stream, having a T-round algorithm for a given problem means that there exist constants $c > 0$ such that given that nodes can send messages of size $c \log n$, they can solve the problem in T rounds.

In this chapter, we explain how to adapt the algorithm of Chapter 4 to the BC-Stream model, and thus prove the following theorem.

THEOREM 5.2. *There exists a* BC-Stream *randomized algorithm that* $(\Delta + 1)$*-colors* n*-vertices graph with maximum degree* Δ *in* $\text{poly}(\log \log n)$ *rounds with high probability.*

Running a randomized color trial remains feasible under BC-Stream constraints. Vertices can also count their degree (and external degree, uncolored degree, etc) exactly. As such, when $\Delta \leqslant \text{poly}(\log n)$, the algorithm of [BEPS16] with that of [GK21] still applies. Hence, this chapter assumes $\Delta \geqslant \text{poly}(\log n)$. The technical difficulties in implementing Algorithm 16 to overcome are:

(1) (high-degree) nodes cannot store all colors used in their neighborhood, in order to know their palette;
(2) dense nodes cannot learn the full clique palette nor the full permutation π during the synchronized color trial (Step 2); and
(3) coloring the put-aside sets the the very end (Step 4).

Dealing with the first issue is fairly straightforward since in order to overcome the broadcast constraint, nodes sample colors in publicly known sets of colors (e.g., $[\Delta+1]$ or $[x(K)]$). When we do so, a constant fraction of the color sparse consist of available colors. After sampling colors in such a set, a node can learn which sampled colors belong to its palette in one communication round (where each colored node broadcasts its color). So calls to TryColor can be implemented in with one extra round of BC-Stream. Each step of our MultiColorTrial for BCONGEST (Algorithm 9) requires $O(\log n)$ memory, provided that the family of representative sets can be computed with small memory. As explained in [HN23, Section 7], representative sets can be built explicitly using *strong averaging sampler*. Although the authors of [HN23] do not mention memory explicitly, since the sampler of [Hea08] is implementable with a $\text{poly}(\log n)$-sized circuit, they provide an explicit construction and a procedure to compute the j-th representative set of the family using $\text{poly}(\log n)$ memory (see [HN23, Algorithm 6]).

Since Theorem 5.2 only claims a $\text{poly}(\log \log n)$ runtime, we do not need to resort the the CompressTry technique to color put-aside sets. Observe that the many-to-all broadcast (Corollary 4.10) works in BC-Stream if the total number of messages is $\text{poly}(\log n)$ or each vertex knows in advance which messages it needs to store. In particular, they can learn the clique palette — or a set of, say, $\Theta(\log^3 n)$ colors from the clique palette when it is too large to store entirely. In almost-cliques where there is no colorful

matching, we can implement Claim 4.36 to learn anti-neighbors and their colors. By trying random colors from the clique palette (or the clique palette augmented with colors from anti-neighbors), vertices get colored with constant probability (see Lemma 11.5), thus we can reduce the size of all put-aside sets to $O(\log n)$ in $O(\log\log n)$ rounds. Since all relevant colors are known an there are $\text{poly}(\log n)$ of them, they can each be represented in $O(\log\log n)$ bits. Broadcasting all available takes $O(\log\log n)$ rounds with many-to-all broadcast, which concludes the coloring of put-aside sets.

The synchronized color trial (Step 2 of Algorithm 16) requires more care. Note that a node v merely needs to know its index in the permutation $\pi(v)$ and the $\pi(v)$-th color in the clique-palette. Lemmas 4.11 and 4.13 are both based on the idea of "random bucketing". Let us focus on the permutation and consider Algorithm 12. As each bucket contains $O(\log n)$ nodes, RELABEL requires only $\text{poly}\log n$ memory (Algorithm 11). What remains, then, is to compute the prefix sum $\sum_{j<i}|S_j|$ counting the number of elements in buckets of lower indices (Step 5 of Algorithm 12). Compared to BCONGEST, the challenge is to avoid double counting. Indeed, in Step 2 of PERMUTE, nodes receive $\Theta(\log n)$ times each term $|S_j|$ of the sum.

Computing prefix sums $\sum_{j<i}|S_j|$ can be done in $O(\log\log n)$ rounds of BC-Stream. To achieve this, we progressively merge together the S_i's into larger groups, keeping track of the groups' sizes as they merge. When groups have size z, we merge $z^{1/2}$ of them together. Computing the size of the result of this merge involves summing $z^{1/2}$ group sizes. In each group, nodes choose a term to learn in the sum at random (among the $z^{1/2}$ terms). In expectation, $z^{1/2}$ nodes are assigned to each term. Because of the highly connected structure of almost-cliques, we can elect a *unique* node for each term, allowing us to aggregate all values without double counting. Since the sizes of the groups grow polynomially, after $O(\log\log n)$ rounds, all sums have been computed.

Lemma 5.3. *Let T_i be a family of sets such as described in Lemma 4.9. Suppose nodes of each group T_i knows some value $y_i \leqslant \text{poly}(n)$. There is a $O(\log\log n)$-round* BC-Stream *algorithm such that w.h.p. all nodes in T_i learn $\sum_{j<i} y_j$.*

The remainder of this Chapter is dedicated to proving Lemma 5.3. We focus our attention on a clique K. We call a *spanning group* a subset $T \subseteq K$ of size $O(\log n)$ and such that for any pair of vertices $u, w \in K$ we have $|T \cap N(u) \cap N(w)| \geqslant C \log n$. Note that the sets T_i produced by each v sampling a random index $i \in [\Delta/(4C \log n)]$ are a family of disjoint spanning groups, w.h.p. (see Lemma 4.9).

We begin by dividing the $\{T_i\}_{i \in [k]}$ in ranges of $z_0 = C \log n$ groups. Groups in the same range *merge*: they learn their prefix sum *inside the range* as well as the sum of all y_j's in the group. At this point of the algorithm, there are no issues of double counting as each node only learns $O(\log n)$ values determined in advance by its spanning group (Lemma 5.4).

We then run $O(\log \log n)$ iterations that recursively merge groups. At iteration i, we merge ranges of $z_i^{1/2}$ groups, where z_i is a lower bound on the size of each group. The size of newly formed groups is at least $z_{i+1} = z_i^{3/2}$. To compute the prefix sums in Lemma 5.3, nodes learn $\sum_j y_j$ over groups of smaller index within their range, as well as the sum over all values for its range. Since each range merges $z_i^{1/2} \ll z_i$ groups, we can randomly assign each term of $\sum_j y_j$ to a unique node in each group. Since groups are union of spanning groups T_i, they are well connected and allow for simple aggregation. This process is formalized in Lemma 5.5.

Lemma 5.4. *Let $T_1, \ldots, T_k$ be a family of spanning groups and let $z_0 = C \log n$. Furthermore, fix some y_i for each $i \in [k]$ and suppose each $v \in T_i$ knows y_i. There is a $O(1)$-round* BC-Stream *algorithm such that nodes of T_i learn all y_j for $1 + \left\lfloor \frac{i-1}{z_0} \right\rfloor z_0 \leqslant j \leqslant \left\lfloor \frac{i-1}{z_0} + 1 \right\rfloor z_0$.*

Proof. Each node must learn $z_0 < |T_i|$ values. If each node in T_i broadcasts the message (i, y_i), then a node v can receive and store its $z_0 = O(\log n)$ values because it has neighbors in each T_i. ∎

Lemma 5.5. *Let K be an almost-clique and $S_1, \ldots, S_m$ be m disjoint subsets of K that are union of disjoint spanning groups, and each of size at least $z \geqslant C^2 \log^2 n$. Furthermore, fix some y_i for each $i \in [m]$ and suppose each $v \in S_i$ know y_i. Then, in $O(1)$ rounds of* BC-Stream*, w.h.p. nodes of S_i can learn the sums*

- *$\sum_j y_j$ where $1 + \left\lfloor \frac{(i-1)}{z^{1/2}} \right\rfloor z^{1/2} \leqslant j < i$, and*

- $\sum_j y_j$ *where* $1 + \left\lfloor \frac{(i-1)}{z^{1/2}} \right\rfloor z^{1/2} \leqslant j \leqslant \left\lfloor \frac{i-1}{z^{1/2}} + 1 \right\rfloor z^{1/2}$.

PROOF. To avoid cumbersome notations, we focus on groups $S_1, \ldots, S_{z^{1/2}}$. To prove the lemma, it suffices to repeat the same process in parallel for each contiguous sub-range of $z^{1/2}$ indices in $[m]$. In each set S_i, nodes sample a random value $r(v) \in [z^{1/2}]$ and form random subsets $R_{i,j} = \{v \in S_i : r(v) = j\}$, one for each term in the sum. By linearity of the expectation, we have that $\mathbb{E}[|R_{i,j}|] = |S_i|/z^{1/2} \geqslant z^{1/2}$. By Chernoff Bound and union bound over j, w.e.h.p. in $z^{1/2}$, hence w.h.p. in n, all $|R_{i,j}|$ have size at least $z^{1/2}/2$. Furthermore, $R_{i,j}$ has strong diameter 2. Indeed, all $u, w \in R_{i,j}$ have $C \log n$ neighbors in common in each spanning group $T \subset S_i$. Since a spanning group $T \subseteq S_i$ has size $O(\log n)$, there must be at least $z/O(\log n)$ such groups. Counting $C \log n$ shared neighbors in $N(u) \cap N(w)$ for each of the $z/O(\log n)$ spanning group contained in S_i, we get that u and w have $\Omega(z)$ common neighbors. Therefore, by Chernoff, w.h.p., u and w have at least $\Omega(z^{1/2})$ common neighbor in $R_{i,j}$.

Nodes $v \in S_i$ broadcast the message (i, y_i) for each $i \in [m]$. Since $|S_i| \geqslant z$ for each $i \in [m]$, we must have $m \leqslant |K|/z \leqslant \Delta/(C \log n)$ different messages. Therefore, they are disseminated in $O(1)$ rounds (by Corollary 4.10). This only uses $O(\log n)$ memory per vertex since each $v \in R_{i,j}$ for $i, j \leqslant z^{1/2}$ stores only the value of y_j.

We now explain how to aggregate these values to compute the two sums in each S_i. Elect an arbitrary *leader* in S_i and arbitrary *chiefs* $R_{i,j}$ for each $j \leqslant z^{1/2}$. Each chief broadcast the ID of one neighbor in S_i shared with the leader. This yields a depth-2 tree induced in S_i, with the leader as root and chiefs as leaves. We aggregate the two desired sums on the tree. Note that the chief in group $R_{i,j}$ is the only node in S_i to broadcast y_j. This avoids double counting. Once the leader has computed the sums, two rounds of BFS diffuse their values to all nodes in S_i. ∎

PROOF OF LEMMA 5.3. We repeatedly aggregate values of larger and larger groups of nodes. We define the following sequence:

$$z_0 = C \log n \ , \quad z_1 = z_0^2 \quad \text{and} \quad z_{i+1} = z_i^{3/2} \ .$$

Our algorithm starts with spanning groups $S_{0,j} := T_j$ and merges $T_{1+(j-1)z_0}$, ..., $T_{j\cdot z_0}$ together in $S_{1,j}$. For the i-th step, it merges $z_i^{1/2}$ groups

$$S_{i,1+(j-1)z_i^{1/2}}, \ldots, S_{i,j\cdot z_i^{1/2}}$$

into $S_{i+1,j}$. It maintains the invariant $|S_{i,j}| \geqslant z_i$ for all iterations i and sets j. By Lemmas 5.4 and 5.5, each iterations takes $O(1)$ rounds. After $O(\log\log\Delta)$ iterations, we merged all groups. Since after each merger, nodes of T_i know the sum $\sum_j y_j$ over all $j < i$ that it was merged with, once all groups are merged together, all groups know their prefix sum. ∎

Part II

Distributed Palette Sparsification

CHAPTER 6

The Distributed Palette Sparsification Theorem

The *Palette Sparsification Theorem* of Assadi, Chen, and Khanna (ACK, henceforth) [ACK19] is a beautiful and powerful sparsification result for the $(\Delta+1)$-coloring problem. ACK show that we can $(\Delta+1)$-color any graph G, by list-coloring a *sparse* sub-graph $\widetilde{G}$, which has only $\tilde{O}(n)$ edges. Their theorem led to several breakthroughs for sublinear algorithms, including graph streaming algorithms, sublinear query algorithms, and massively parallel computation algorithms.

More precisely, the theorem states that for any graph G, if we independently sample random a list $\mathbf{L}(v)$ of $O(\log n)$ colors for each vertex $v \in V$, with high probability, the graph G is $\mathbf{L}$-list-colorable. That is, there exists a coloring of G where each v is assigned a color $\varphi(v) \in \mathbf{L}(v)$. To compute a $(\Delta+1)$-coloring of G, one then computes an $\mathbf{L}$-list-coloring of the sub-graph $\widetilde{G}$ retaining only edges $uv \in E$ where $\mathbf{L}(u) \cap \mathbf{L}(v) \neq \emptyset$. A simple argument shows that $\widetilde{G}$ is sparse and has maximum degree $O(\log^2 n)$, thereby giving the aforementioned sub-linear algorithms.

The ACK result gives rise to the hope that there might be an ultimately scalable (distributed) solution for the $(\Delta+1)$-coloring problem, where each graph node needs to interact and coordinate with only $\text{poly}(\log n)$ of its neighbors. However, all known applications of the palette sparsification theorem require gathering the sparsified subgraph $\widetilde{G}$ in one location, and solving the resulting list-coloring problem in a centralized fashion. This is prohibitively expensive in distributed models with restrictive communication, e.g., if each node can send/receive only $\text{poly}\log n$ bits per round.

In this part of the thesis, we remedy this problem by giving a nearly-optimal *distributed* version of the palette sparsification theorem. Informally, we show that there is a fast distributed algorithm for coloring the sparsified subgraph, and using communications only on the sparsified graph (modulo

a small relaxation in the graph's degree, compared to ACK). This leads to the first poly-logarithmic randomized algorithms in constrained settings studied in the distributed literature [RGHZL22; GKKLP18; GH16; GK13; GZ22; AGGHSKL19].

In constrained distributed models such as *cluster graphs* and *the node capacitated clique*, where nodes can effectively send/receive only $\operatorname{poly}\log n$ bits per rounds (or more generally, $\operatorname{poly}\log n$ bit aggregate summaries of the messages), no $\operatorname{poly}\log n$ algorithm is known. A major impediment is this: all known algorithms work by computing the coloring gradually, and in the intermediate steps, nodes need to learn which colors are already used by their neighbors. This forces communications that need $\Omega(\Delta)$ bits.

The palette sparsification theorem of [ACK19] reduces the problem of $(\Delta+1)$-coloring G to a list-coloring problem on a graph $\widetilde{G}$ with $O(\log^2 n)$ maximum degree. Hence, it seemingly opens the road for ultimately scalable distributed algorithms, where each node sends/receives only $\operatorname{poly}(\log n)$ bits. However, that hinges on whether $\widetilde{G}$ can be colored fast *distributively*. Unfortunately, the proof of [ACK19] is intrinsically centralized (for reasons explained in Section 6.2.1). All applications of [ACK19] use centralization to compute the $(\Delta+1)$-coloring. The research question at the core of this chapter is to investigate the discrepancy between the locality of the $(\Delta+1)$-coloring problem and the locality of the induced list-coloring problem on the sparsified graph.

Can the sparsified graph be colored ***locally****?*

6.1. Our Results

Our answer is two-fold. We design an algorithm for $(\Delta+1)$-coloring such that the color of a vertex v depends only on its $O(\log^2 \Delta)$-hop neighborhood in $\widetilde{G}$ (when $\Delta \geqslant \Omega(\log^4 n)$), and we concretely give efficient distributed algorithms with small messages to compute such a coloring. Conversely, we show that no algorithm can achieve a locality smaller than $\widetilde{\Omega}(\log \Delta)$. We next state the results in a more formal manner.

We present a CONGEST algorithm to list-color the sparsified graph in $O(\log^2 \Delta)$ rounds when $\Delta \geqslant \Omega(\log^4 n)$. When $\Delta \leqslant O(\log^4 n)$, the input

graph G is sparse already and can be colored by the $O(\log^3 \log n)$-round state-of-the-art CONGEST algorithm [HKNT22; HNT22; GK21].

THEOREM 6.1 (Distributed Palette Sparsification Theorem). *Suppose that each node in an n-vertex graph G with maximum degree $\Delta \gg \log^4 n$ samples $\Theta(\log^2 n)$ colors u.a.r. from $[\Delta+1]$. There is a distributed message-passing algorithm operating on the sparsified graph, that computes a valid list-coloring in $O(\log^2 \Delta)$ rounds, using $O(\log n)$-bit messages. In particular, each node needs to communicate with only $O(\log^4 n)$ different neighbors.*

Assumption on Δ. In Theorem 6.1, we assume that $\Delta \geqslant c \log^4 n$ for some large constant $c > 0$. We remark that this assumption can be removed by sampling lists of $\widetilde{O}(\log^4 n)$ colors instead of $O(\log^2 n)$ because it would cause vertices to sample all colors with high probability. When $\Delta \leqslant c \log^4 n$, the graph is sparse to begin with and the algorithm of Chapter 4 colors the graph in $O(\log^3 \log n)$ rounds with every vertex communicating with at most $\Delta \leqslant c \log^4 n$ different neighbors, as in Theorem 6.1.

Our Techniques in a Nutshell. We shall give an overview of our algorithm in Section 6.2. For now, we merely mention three aspects in which our algorithm differs significantly from both streaming and distributed algorithms.

(1) Contrary to [ACK19], we cannot afford to color dense clusters sequentially. In particular, when we color a cluster, we cannot assume that colors on the outside are adversarial. We give an algorithm that functions as long as conflicting colors with the outside are only a small fraction of the color space. To ensure that this property holds, we *precondition* clusters. That is, we reduce the number of connections between clusters beforehand, so that, later in the algorithm, random decisions on the outside only harm a small enough fraction of nodes on the inside. This preconditioning might be useful in other applications of palette sparsification.

(2) We introduce a new technique of *augmenting trees* to distributively color dense clusters of the graph. It consists of $O(\log \Delta)$ steps for growing trees rooted at uncolored nodes such that if a leaf can

recolor itself, we can color the root. We show that a constant fraction of the uncolored nodes are colored by this process, resulting in the $O(\log^2 \Delta)$ runtime.

(3) To reach the nearly-optimal $O(\log^2 \Delta)$ runtime, we need this process to succeed with high probability *even when* $o(\log n)$ *nodes remain uncolored*. We overcome this issue by locally amplifying probabilities and using that only few nodes remain to resolve contentious efficiently.

Lower Bound. We give evidence that this round complexity is in the right ballpark, by showing that $\Omega(\log \Delta / \log \log n)$ rounds are needed to compute a valid coloring after sparsification of neighborhoods by uniformly random palette sparsification.

THEOREM 6.2. *Any* LOCAL *algorithm that operates on the sparsified graph and computes a* $(\Delta+1)$*-coloring with at least a constant probability of success needs* $\Omega\left(\frac{\log \Delta}{\log \log n}\right)$ *rounds. This holds even if the original graph is a* $(\Delta+1)$*-clique, even if the distributed algorithm running on the sparsified graph uses unbounded messages, and even if each node samples a large* $\operatorname{poly} \log n$ *number of colors in the sparsification.*

Let us give a brief overview of the implications of Theorem 6.1 to distributed models beyond CONGEST.

Corollary 1: Distributed Streaming. The semi-streaming model — where we skim through the (very large) set of edges of the graph and store only $\operatorname{poly} \log n$ bit of memory per node before solving the problem — has been studied extensively [AMS96; BJKST02; CCF02; CM04; AGM12; ACK19]. A frequent technique in this setting is *(distributed) sketching* [AGM12; KLMMS14; GMT15; ACK19]: nodes locally compress their neighborhoods to $\operatorname{poly} \log n$-bit sketches before combining all of them centrally. We find it helpful to think of our algorithm in the following similar setting: first, nodes look at their edges in one streaming pass, using only $\operatorname{poly} \log n$ memory; nodes then communicate in a distributed fashion using only edges they stored locally. We emphasize, however, that it is crucial for our applications that *nodes communicate only with* polylog$n$ *nodes per*

round. Contrary to the semi-streaming model, it does not suffice to reduce the problem to $\widetilde{O}(n)$ edges: each neighborhood must contain at most $\operatorname{poly}\log n$ edges. See Section 6.3.1 for a more precise definition.

Corollary 2: Coloring Cluster Graphs. A natural situation that arises frequently in distributed graph algorithms is that of *cluster graphs*, as we explain next (this appears under various names, see e.g., [RGHZL22; GKKLP18; GH16; GK13; GZ22]). Suppose that in the course of some algorithm, the nodes have been partitioned into vertex-disjoint (low-diameter) clusters. The corresponding *cluster graph* is an abstract graph with one node for each cluster, where two clusters are adjacent if they contain neighboring nodes. Note that this corresponds to (graph-theoretically) *contracting* each cluster, an operation that is easy centrally but has no meaningful distributed counterpart. In distributed settings with such cluster graphs, we often need to solve certain graph problems on this cluster graph to facilitate other computations. Distributed computation on the cluster graph assumes we have a low-depth cluster tree that spans each cluster and can be used for broadcast and convergecast in the cluster. One round of communication on the cluster graph involves: (1) broadcasting a $\operatorname{poly}(\log n)$-bit message from the cluster center to all its nodes; (2) passing information on the edges between neighboring clusters; (3) convergecasting any $\operatorname{poly}(\log n)$-bit aggregate function from the cluster nodes to the center.

Prior to this work, it remained open whether one can compute a $(\Delta+1)$-coloring in $\operatorname{poly}\log n$ rounds of communication on the cluster graph. Here, Δ denotes the maximum number of clusters that are adjacent to a cluster. The more traditional approaches to $(\Delta+1)$-coloring (e.g., [Lin92; Joh99; BEPS16]) fall short of this $\operatorname{poly}\log n$ round complexity goal as they usually need to learn the colors remaining available to one cluster, after some partial coloring of other clusters, and that may require gathering Δ bits at the cluster center. Our distributed palette sparsification theorem resolves this and gives the first efficient distributed $(\Delta+1)$-coloring on cluster graphs, as we state informally next. See Section 6.3.2 for definitions and the actual result.

Corollary 3: Coloring in the Node Capacitated Clique. Another immediate consequence of our work is the first poly $\log n$-round Node Capacitated Clique algorithm for $(\Delta+1)$-coloring. This model was introduced by [AGGHSKL19] to capture peer-to-peer systems in which nodes have access to global communication in the network while being restrained to $O(\log n)$-bit messages to $O(\log n)$ nodes within one communication round. To identify the edges to use in the sparsified graph, we assume the nodes have access to shared randomness; alternatively, as we show in Section 6.3.3, this can be replaced by an existential construction.

Related Work and Problems. The groundbreaking palette sparsification theorem of Assadi, Chen, and Khanna [ACK19] showed that $(\Delta+1)$-coloring was possible in the semi-streaming model, even in dynamic streams. The sparsification property was fundamental, as it allowed also for optimal algorithms in two, seemingly unrelated models: a sublinear-time algorithm in the query model, and a two-round algorithm in the Massively Parallel Computation model with $\tilde{O}(n)$ memory per machine. The theorem was extended to several more constrained coloring problems in [AA20], such as $O(\Delta/\log\Delta)$-coloring triangle-free graphs, and to deg $+1$-list coloring in [HKNT22]. It was also a crucial ingredient in the recent semi-streaming algorithm for Δ-coloring [AKM23].

Palette sparsification is a form of a sampling technique that holds in the restrictive *distributed sketching model*. The latter corresponds to multi-party communication with shared blackboard model and vertex partitioned inputs, as well as to the *broadcast congested clique* (though the congested clique term usually refers to the case that the message size is $O(\log n)$). Starting with the seminal work of [AGM12], many graph problems have been solved with distributed sketching. Though, notably, the problems of maximal independent set and maximal matching—which are closely related to $(\Delta+1)$-coloring—have been shown to require much larger space [AKZ22], even if allowed multiple rounds of writing to the shared blackboard.

Organization of Part II. In the remainder of this chapter, we give a high-level overview of the main ideas for Theorems 6.1 and 6.2. We conclude the chapter with formal statements for each aforementioned corollary. In

Chapter 7 we give the complete proof for Theorem 6.1 and in Chapter 8 the proof for Chapter 8.

6.2. Technical Introduction

In this section, we outline the techniques we use to prove Theorems 6.1 and 6.2. Our algorithm builds on existing literature, both streaming [ACK19; AA20; AKM23] and distributed [SW10; EPS15]. It differs nonetheless from both in key aspects. On the one hand, streaming algorithms [ACK19; AA20] require a global view of the sparsified graph — which we avoid by computing the coloring in a distributed fashion. On the other hand, existing distributed algorithms [BEPS16; HSS18; CLP20; HKNT22] require that nodes communicate with all of their neighbors — which we avoid since nodes communicate only on the sparsified graph.

In Section 6.2.1, we review the palette sparsification theorem of [ACK19] and explain why it does not extend to our setting. Then, in Section 6.2.2, we outline the technical novelties in our algorithm. Finally, in Section 6.2.3, we describe an overview of our lower bound result.

6.2.1. Comparison with Palette Sparsification.

The proof of Assadi, Chen and Khanna relies on the almost-clique decomposition (Section 3.2). As explained in Section 3.3, a random color trial provides sparse vertices with $\Omega(\Delta)$ slack with high probability. This allows [ACK19] to color sparse vertices greedily: when a sparse vertex is reached, no matter what colors were given to neighbor, it has $\Omega(\Delta)$ colors available, so sampling $O(\log n)$ colors in $[\Delta+1]$ suffices to ensure that at least one is available with high probability. So the part of the algorithm of [ACK19] for coloring sparse vertices is straightforward to implement in $O(\log n)$ rounds distributively. Theorem 6.1 improves the round complexity to $O(\log \Delta)$ by using SlackColor (Section 3.4) to color sparse nodes, but we do not claim any significant technical novelty in that part.

In [ACK19], most of the effort (and novelty) goes into the handling of almost-cliques. They iterate over almost-cliques *sequentially* and color each one assuming the coloring on the outside is *adversarial*. Clearly, in

our setting, we cannot afford to process almost-cliques one by one. Furthermore, to achieve the $O(\log^2 \Delta)$ runtime claimed by Theorem 6.1, for reasons expounded in Section 6.2.2.3, we cannot assume the outside colors to be adversarial. When we color each almost-clique, we must carefully resolve contentions with the outside, including other almost-cliques that are getting colored in parallel.

To color a fixed almost-clique K, [ACK19] looks for a perfect matching in the bipartite graph with vertices of K on the one side, colors in $[\Delta + 1]$ on the other and an edge between $v \in K$ and $\chi \in [\Delta + 1]$ if χ is in $\mathbf{L}(v)$ and not used by an outside neighbor. If $|K| \leqslant \Delta + 1$, classic results from random graph theory (e.g., [Bol11, Section VII.3]) show that, with high probability, a perfect matching exists, and therefore a list-coloring is possible. While the mere existence of the matching is enough for [ACK19], Theorem 6.1 provides a *distributed algorithm* to compute it. Note that learning the full topology of K in $O(\log \Delta)$ rounds of LOCAL to then decide on a matching does not work because it does not account for conflicts with concurrent almost-cliques. Furthermore, our algorithm uses $O(\log n)$-bit messages, which prohibit centralization approaches. Our main technical contribution is the design of an $O(\log^2 \Delta)$-round algorithm using $O(\log n)$-bit messages to compute this matching in almost-cliques in parallel (see Section 7.5), while also managing outside conflicts.

In large almost-cliques, i.e., such that $|K| > \Delta + 1$, such a perfect matching cannot exist. Like in Chapter 4, we compute a colorful matching (Section 4.6) to deal with those. In almost-cliques where $a(K) \gg \log n$, we use the same algorithm as in BCONGEST but, contrary to BCONGEST, we also compute a colorful matching in almost-cliques where $a(K) \leqslant O(\log n)$.

Another noteworthy challenge for us is regarding *probability amplification*. Contrary to the centralized setting, we cannot afford algorithms with a constant probability of success. Indeed, amplifying success probability with $O(\log n)$ independent repetitions would exceed our $O(\log^2 \Delta)$ runtime. We deal with this issue by increasing the number of colors sampled, compared to [ACK19], such that we always have concentration on large enough quantities, i.e. at least $\Omega(\log n)$. Naturally, trying more colors creates more conflicts, which must be resolved.

6.2.2. Our Approach and New Ideas. The coloring is computed in three main steps. We give a brief overview of the main novelty for each.

6.2.2.1. *Step 1: Preconditioning of Almost-Cliques.* The preconditioning step strengthens the properties of our almost-cliques. More precisely, it computes a partial coloring such that all uncolored nodes are clustered in almost-cliques and have $O(\Delta/\log n)$ connections to uncolored nodes in other almost-cliques, compared to the usual $\varepsilon\Delta$ (see Theorem 7.11 for a formal definition). This property is key to ensure, later in our algorithm, that random decisions outside of a cluster cannot seriously impede its progress on the inside (see Lemma 7.28). We now give further details on that aspect.

The key property of cliques that our algorithm uses is that when k nodes are uncolored, there are k colors that are used by no one in the clique. The colorful matching (recall Lemma 4.19) allows us to extend this to almost-cliques. However, we still need to ensure that if a node (re)colors itself with one of these k available colors, it will not create a conflict with an external neighbor. For the sake of concreteness, assume only one node is left to color in almost-clique K, i.e., $k = 1$. In our algorithm, a constant fraction of K samples $c \log n$ colors for some large enough $c > 0$. This way, the probability that at least one node in this almost-clique finds that one available color is at least $1 - (1 - 1/\Delta)^{\Delta \cdot c \log n} \geqslant 1 - 1/\text{poly}(n)$. Having nodes sample more colors has a drawback: it increases the competition for colors. If a node $v \in K$ has $\varepsilon\Delta$ external neighbors in active almost-cliques, the probability that at least one of them blocks the one color that v is looking for is $1 - (1 - 1/\Delta)^{\varepsilon\Delta \cdot c \log n} \geqslant 1 - 1/\text{poly}(n)$ (for $c \gg 1/\varepsilon$). During preconditioning, we ensure that nodes in active almost-cliques have at most $\Delta/(c \log n)$ external neighbors in other active almost-cliques. The probability that an external neighbor blocks the one color that v is looking for becomes $1 - (1 - 1/\Delta)^{\frac{\Delta c \log n}{c \log n}} \approx 1 - 1/e$. Therefore, only a small fraction of K is affected by the randomness outside of K. This argument is made formal in Lemma 7.28.

To precondition almost-cliques, we use that nodes that have $\Omega(\Delta/\log n)$ connections to nodes in other almost-cliques are $\Omega(\Delta/\log n)$ sparse. By coloring nodes in a carefully chosen order, we get Theorem 7.11. While the

preconditioning algorithm in itself is not a major contribution of our work, we believe it could find further use in the (distributed) coloring literature.

6.2.2.2. *Step 2: Distributed Colorful Matching.* In almost-cliques where $a(K) \gg \log n$, the BCONGEST algorithm (Algorithm 14 in Section 4.6) computes a colorful matching of size $\Omega(a(K)/\varepsilon)$ with high probability. Since it only have nodes try uniform colors in $[\Delta + 1]$, it works here as well, i.e., all colors can be sampled in advance.

The main difference with the BCONGEST algorithm, is that we need a colorful matching also in almost-cliques where $a(K) \leqslant O(\log n)$. Note that when $a(K)$ is smaller than some constant, nodes have hardly any anti-edges. Hence we can also assume $a(K) \gg 1$; meaning the previous algorithm succeeds with constant probability. Instead of trying a single color, nodes try $\Theta(\log n)$ colors at the same time. Clearly, a large enough colorful matching exists: using the sampled colors, the previous process can be implemented in $O(\log n)$ rounds. To find that matching efficiently (in $O(\log \Delta)$ rounds), we capitalize on the fact that there are few non-edges in the almost-clique (Lemma 7.23). Since the probability of a non-edge having both endpoints sample a common color is $\Theta(\log^2 n/\Delta)$, only $O(a(K) \log^2 n) = O(\log^3 n)$ potential monochromatic non-edges are sampled. By taking advantage of the high expansion property of the sparsified almost-clique, we can disseminate the list of $O(\log^3 n)$ sampled monochromatic edges in $O(\log \Delta)$ rounds to all nodes in the almost-clique, which can then compute the colorful matching locally. Because $\Omega(\Delta)$ colors are available in the almost-clique, the concurrent coloring of external neighbors can block at most a small fraction of the colors (Lemma 7.20).

6.2.2.3. *Step 3: Augmenting Trees.* To color almost-cliques, we take advantage of the fast expansion of the sparsified almost-clique to find many *augmenting paths*. Our definition of augmenting path corresponds precisely to the one for computing maximal matching in the random bipartite graphs induced by the random lists of colors [HK73; Mot94]. We emphasize, however, that general-purpose algorithms for maximal matching do not directly apply in our setting because of conflicts between concurrent almost-cliques. Furthermore, computing an *exact* maximum matching is a global problem,

and in fact requires at least $\Omega(\sqrt{n})$ rounds of CONGEST in general, even in low-diameter graphs [AKO18].

We now explain how we color all almost-cliques in $O(\log^2 \Delta)$ rounds. The algorithm runs $O(\log \Delta)$ iterations of the following process. Suppose k is number of uncolored nodes in K at the current iteration. We say that color χ is *available* to a node v for this iteration if χ is not used in K, it is not used by colored external neighbors of v nor sampled by active external neighbors during this iteration. In each iteration, we grow a forest of *augmenting paths* (Definition 7.25). An augmenting path is a path $u_0, u_1, \dots u_i$ in the *sparsified* almost-clique such that 1) u_0 is the only uncolored node, 2) each u_{j-1} for $j \in [i]$ can (re)color itself with the color of u_j and 3) the last node u_i of the path knows an available color χ. Provided with such a path, we can recolor u_i with χ and each u_{j-1} with the color of u_j, thereby coloring the uncolored endpoint u_0. Our algorithm builds on the following idea: if we have a path $u_0, \dots, u_i$ verifying 1) and 2) but not 3), then u_i samples a uniform color $\chi \in [\Delta + 1]$ and finds an available one with probability $\Omega(k/\Delta)$.

The two prior steps of our algorithm are key to ensure u_i has probability $\Omega(k/\Delta)$ to find an available color *that it can adopt*. The colorful matching ensures (almost) all nodes of K will have k colors available (Lemma 4.19). On the other hand, the argument sketched in Section 6.2.2.1 shows that because of the preconditioning step, with high probability over the randomness outside of K, at least $\Omega(\Delta)$ nodes in K can adopt $k/2$ of the colors available to them. We say of nodes that cannot adopt $k/2$ available colors that they are spoiled (Definition 7.27). Since they represent a small fraction of K and the path explores the almost-clique randomly, we are unlikely to fail due to spoiled nodes (Lemma 7.31).

This simple algorithm colors u_0 with probability $\Omega(k/\Delta)$. To color each uncolored node with constant probability, even when $k \ll \Delta$, we grow $\Delta/(\alpha k)$ paths verifying 1) and 2) from each uncolored nodes for some large enough constant $\alpha > 1$. The expected number of paths to find an available color is $\Delta/(\alpha k) \cdot \Omega(k/\Delta) = \Omega(1/\alpha)$. Therefore, we color $\Omega(k/\alpha)$ nodes in expectation. Since we must avoid collisions between paths from different

uncolored nodes, we find it helpful to further restrict paths to grow trees. See Figure 1 for a high-level description of one iteration.

AUGMENTING PATH ALGORITHM

Let k be the number of uncolored nodes.

Growing the forest. The uncolored nodes are the roots of the forest. Repeat $O(\log \frac{\Delta}{\alpha k})$ times:

(1) Each leaf samples a set S_u of $O(\log n)$ colors.
(2) Remove from S_u the colors used by external neighbors, nodes in the forest, or sampled by other leaves.
(3) For each $\chi \in S_u$ find $v_{c,u} \in K$ colored χ and connect them to v in the forest.

Harvesting the trees.

(1) If $k \geqslant \Omega(\log n)$, each leaf tries one random color in $[\Delta + 1]$. If a leaf can retain its color, we recolor the path connecting it to the root, root included.
(2) If $k \leqslant O(\log n)$, each leaf tries $O(\log n)$ random colors in $[\Delta + 1]$. The roots learn $\Theta(k)$ colors with which leaves in their trees can recolor themselves. They disseminate this list in $O(\log \Delta)$ rounds to all nodes of K. Nodes then compute a color-leaf matching and recolor the corresponding paths.

Figure 1. High level description of one iteration.

An iteration has two phases: the *growing phase* (Algorithm 23), where we grow the trees, and the *harvesting phase* (Algorithms 24 and 25), where we try to recolor augmenting paths. The growing phase needs $O(\log \frac{\Delta}{\alpha k})$ rounds because each time we increase the number of paths by a constant factor. The harvesting phase differs depending on k. That is because when $k \leqslant O(\log n)$, we cannot show progress *with high probability* with a simple concentration on k. See Section 7.5.1 for a more detailed description.

6.2.3. Lower Bound. We complement our upper bound with a lower bound on the distributed complexity of coloring a graph after random

palette sparsification. The lower bound applies even if the graph prior to the palette sparsification was simply a complete graph. Concretely, the lower bound states the following. Assume that we have a complete graph K_n on n nodes $V = \{1, \dots, n\}$ that we want to color with $n = \Delta + 1$ colors. Every node $v \in V$ samples a random subset S_v of colors $C = \{1, \dots, n\}$ as follows. For each node $v \in V$ and each color $x \in K$, x is included in S_v independently with probability $p = f(n)/n$, where $f(n) \geqslant c \ln n$ for a sufficiently large constant c and $f(n) \leqslant \text{polylog} n$. Recall that the sparsified graph is the graph induced by all edges of K_n between nodes $u, v \in V$ with $S_u \cap S_v \neq \emptyset$. We prove that any distributed message passing algorithm on the sparsified graph requires $\Omega(\log n / \log \log n)$ rounds to properly color the K_n in such a way that each node v is colored with a color from its sample S_v. This holds even if the message sizes are not restricted.

Relation to perfect matching on random bipartite graphs. Note that this is equivalent to the following bipartite matching problem. Define a bipartite graph $B = (V \cup C, E_B)$, where $C = \{1, \dots, n\}$ represents the set of colors. There is an edge between nodes $v \in V$ and $x \in K$ whenever $x \in S_v$. A *valid coloring* of the nodes in V then corresponds to a *perfect matching* in the bipartite graph B. Note that if $p \geqslant c \ln n/n$ for a sufficiently large constant $c > 0$, then the bipartite graph B has a perfect matching with high probability. This (in even sharper versions) is well known in the random graph literature (e.g., [Bol02, Section VII.3]) and can be proven by checking Hall's condition for any non-empty subset of V. Our lower bound essentially shows that distributedly computing a perfect matching in the random graph B requires $\Omega(\log n / \log \log n)$ rounds with at least constant probability, even in the LOCAL model (i.e., even if the nodes in B can exchange arbitrarily large messages). Note that the sparsified subgraph of K_n and the bipartite graph B can simulate each other with only constant overhead in the distributed setting. Any T-round algorithm on the sparsified graph can be run in $O(T)$ rounds on B and any T-round algorithm on B can be run in $O(T)$ rounds in the sparsified subgraph of K_n (in the second case, each color node $x \in K$ can be simulated by one of the nodes $v \in V$ for which $x \in S_v$).

Lower bound on computing a perfect matching in random bipartite graphs. In general, it is not too surprising that computing a perfect matching of a graph is a global problem where nodes at different ends of the graph need to coordinate. Consider for example the problem of computing a perfect matching of a $2n$-node cycle. There are exactly two such perfect matchings and deciding which of the two matchings to choose cannot be decided without global coordination within the cycle. However, the case of a random bipartite graph needs much more care.

Our lower bound is based on the following observation regarding perfect matchings in bipartite graphs. Let v_0 be some node of a bipartite graph H and for each $d \geqslant 0$, let V_d be the set of nodes of H that are at hop distance exactly d from v_0. Since H is bipartite, a node in a set V_d can only be connected to nodes in sets V_{d-1} and V_{d+1}. Clearly, $|V_0| = 1$ and, because v_0 must be matched, there must be exactly one matching edge between nodes in V_0 and nodes in V_1. Further, since every other node of V_1 must be matched to nodes in V_2, there must be exactly $|V_1|-1 = |V_1|-|V_0|$ matching edges between nodes in V_1 and nodes in V_2. With a similar argument, the number of matching edges between nodes in V_2 and nodes in V_3 is exactly $|V_2|-|V_1|+|V_0|$. By extending this argument, one can see that for every d, the number of matching edges between V_d and V_{d+1} depends on the sizes of all the sets $V_0, \ldots, V_d$. Changing the size of a single one of those sets also changes the number of matching edges between V_d and V_{d+1}.

For the lower bound proof, we now proceed as follows. Assume that there is a T-round distributed algorithm that computes a perfect matching of the random bipartite graph B. We consider some node v_0 in the random bipartite graph B and two integers ℓ and h such that $\ell > 0$ and $h - \ell > T$. We consider the decisions of the assumed distributed perfect matching algorithm for nodes in V_h. Note that in T rounds, nodes in V_h do not see nodes at distance more than T, and in particular, they do not see nodes in V_ℓ. However, by the above observation, the number of matching edges between nodes in V_h and nodes in V_{h+1} depends on the knowledge of $|V_\ell|$. If T is sufficiently small and a large fraction of the graph is outside the T-hop neighborhoods of nodes in V_h, then even collectively, the nodes in V_h have significant uncertainty about the value of $|V_\ell|$. Therefore, they

cannot determine the number of matching edges between V_h and V_{h+1} (and thus their matching edges) with reasonable probability. The actual proof that formalizes this intuition is somewhat tedious. The details appear in Chapter 8.

6.3. Corollaries for Other Models

Let us conclude this chapter with a formal discussion of Theorem 6.1 in distributed models where vertices have $\text{poly}(\log n)$ memory, are low-diameter trees in the communication network or can communicate with up to $\text{poly}(\log n)$ neighbors per round.

6.3.1. Coloring in Distributed Streaming. We introduce the Dist-Stream model to capture memory limitations in the CONGEST model. It can be seen as a distributed variant of the semi-streaming model for graph problems.

Definition 6.3 (**Local Streaming Model**)**.** In the Dist-Stream model, there are n nodes with unique $O(\log n)$-bit identifiers and $p(n) = \text{poly}(\log n)$ bits of local space. The nodes have no initial information but have a limited source of randomness. There are two phases: a streaming phase and a communication phase.

- **(Streaming Phase)** Nodes receive their incident edges in the graph G as a stream. Attached to each edge are (some of the) random variables of the incident vertices. I.e., each node v receives a sequence $(v, u_i, s_i)_i$, where s_i^j is the random bits of neighbor u_i in iteration j [1].
- **(Communication Phase)** The nodes communicate in synchronous rounds with their neighbors with $O(\log n)$ bit messages (as in the CONGEST model). They can only send a message to a neighbor whose ID they have stored, and *we additionally limit them to send/receive* $\text{poly}\log n$ *messages per rounds.*

[1] An alternative would be to supply the nodes with shared randomness. Then the ID of the other node would suffice to learn its random bits.

At the end of the computation, each node outputs its color, which together should form a valid $(\Delta+1)$-coloring. The objective is to minimize the total number of communication rounds.

Corollary 6.4. *There exists a* Dist-Stream *algorithm using* $O(\log^4 n)$ *memory per node and* $O(\log^2 \Delta)$ *rounds of communication.*

6.3.2. Coloring in the Cluster Graph Model. We first define the cluster graph model (a variant appears in [GKKLP18] and similar concepts appear in other places in the literature, see e.g., [RGHZL22; GH16; GK13; GZ22]). Then, we state our result.

Definition 6.5 (**Cluster graph model**)**.** Consider a cluster graph defined as follows: Given a graph $G = (V, E)$, suppose that the nodes have been partitioned into vertex-disjoint clusters. Define the cluster graph as an abstract graph with one node for each cluster, where two clusters are adjacent if they include two nodes that are neighboring each other in G. Furthermore, for each cluster, we are given a cluster center and cluster tree that spans from the cluster center to all nodes of the cluster. One round of communication on the cluster graph involves the following three operations:

- (**Intra-cluster broadcast**) Each cluster center starts with a $\text{poly}(\log n)$-bit message and this message is delivered to the nodes in its cluster.
- (**Inter-cluster communication**) For each edge $e = \{v, u\}$ for which v and u are in two different clusters, node v can send a $\text{poly}(\log n)$-bit message and this message is delivered to u, simultaneously for all such inter-cluster edges.
- (**Intra-cluster convergecast**) Each node can start with a $\text{poly}(\log n)$-bit message and, in each cluster, we deliver a $\text{poly}(\log n)$-bit aggregate of the messages of the cluster's nodes to the cluster center. The aggregate function can be computing the minimum, maximum, summation, or even gathering all messages if there are at most $\text{poly}(\log n)$ many. These suffice for our application. More generally, this intra-cluster convergecast operation can be any problem that can be computed in $O(h)$ rounds of the CONGEST model communication on a given tree of depth h and using $\text{poly}(\log n)$-bit messages.

Corollary 6.6. *There is a distributed randomized algorithm that computes a* $(\Delta+1)$*-coloring in* $\text{poly}(\log n)$ *rounds of the cluster graphs model.*

Proof Sketch. The proof follows essentially directly from our distributed palette sparsification theorem, stated in Theorem 6.1. We just need to discuss how the cluster graph computes and simulates the corresponding sparsified graph.

Each cluster center samples the $\text{poly}(\log n)$ colors of its node in the palette sparsification theorem. Then, via intra-cluster broadcast, the cluster center delivers these colors to all nodes of its cluster. Afterward, via inter-cluster communication, each node sends the colors of its cluster to all neighboring nodes in other clusters. Each node v in a cluster $\mathcal{C}$ that notices a neighboring cluster $\mathcal{C}'$ that sampled a common color remembers the cluster identifier of $\mathcal{C}'$, as a neighboring cluster in the sparsified variant of the cluster graph. We then perform one intra-cluster convergecast, where each node starts with the neighboring clusters that it remembered as neighboring clusters in the sparsified graph, and we gather all of these neighboring cluster identifiers to the cluster center. Since each cluster has $\text{poly}(\log n)$ neighboring clusters after the sparsificaiton, this can be done as a $\text{poly}(\log n)$-bit aggregation.

In the course of this process, we could also elect for each pair of neighboring clusters $\mathcal{C}$ and $\mathcal{C}'$ in this sparsified graph one physical edge from node $v \in \mathcal{C}$ to a node $u \in \mathcal{C}'$. For instance, that can be the edge (v, u) with the highest ID tuple. Again, this fits easily as a $\text{poly}(\log n)$-bit aggregation.

At this point, each cluster center knows all its $\text{poly}(\log n)$ neighboring clusters and has identified a physical edge connected to each neighboring cluster. Hence, the cluster graph model can simulate one round of the CONGEST model communication on the sparsified graph. Therefore, to compute a $\Delta+1$ coloring of the cluster graph, it suffices to invoke Theorem 6.1. ∎

6.3.3. Coloring in the Node Capacitated Clique. We show, in fact, that any 1-pass Dist-Stream algorithm with $\text{poly}\log n$ memory and bandwidth can be turned into a $\text{poly}\log n$ rounds NCC algorithm.

Lemma 6.7. *Let* $\mathscr{A}$ *be a randomized* Dist-Stream *algorithm using* one streaming pass *and* T*-communication rounds. If it has bandwidth* B *and*

communication with at most D different neighbors within a round, then there is an algorithm emulating $\mathscr{A}$ with high probability in the NCC *model in $O(\log n + \frac{T \cdot BD}{\log n})$ communication rounds.*

Consider an arbitrary communication round of Dist-Stream. In the worst case, a node must send B bits to D nodes. Since in the NCC model, a node can only communicate $O(\log n)$ bits to $O(\log n)$ nodes in G within a round, it can emulate one communication round of Dist-Stream in $O(BD/\log n)$ rounds. Note that this upper bound can be improved in some specific cases, e.g., if the algorithm only broadcast messages, but we ignore such optimizations here. This gives the following claim:

Claim 6.8. *If nodes know the* $\operatorname{poly}\log n$ *random bits of their neighbors, then emulating $\mathscr{A}$ requires $O\left(\frac{T \cdot (BD)}{\log n}\right)$.*

The only information missing to nodes in order to run the Dist-Stream algorithm is the initial state of their neighbors. As nodes have no memory restriction in NCC, we are free to use pseudo-random initial states. For a function S mapping nodes to $\operatorname{poly}\log n$ bits binary strings, we write $\mathscr{A}[S]$ for the algorithm $\mathscr{A}$ where each node u has the string $S(u)$ as random bits.

Lemma 6.9. *For any fixed n, there is a family $\mathcal{S}$ of* $\operatorname{poly}(n)$ *functions mapping nodes to initial states such that for any n-node graph input, if we run $\mathscr{A}$ on a random function in $\mathcal{S}$, the algorithm succeeds with high probability.*

Proof of Lemma 6.9. Fix a n-node graph G. Sample t functions $S_1, \ldots, S_t$ assigning $\operatorname{poly}\log n$-bits binary string to nodes.

Since $\mathscr{A}$ is correct with high probability, it means that on a random S_i, algorithm $\mathscr{A}[S_i]$ fails with probability at most $1/n$. For a fixed G, call $\mathbf{X}_i$ the random variable equal to one iff $\mathscr{A}[S_i]$ fails on G. In expectation, the number of bad assignments is $\mathbb{E}\left[\sum_{i\in[t]} \mathbf{X}_i\right] \leqslant t/n$. Samples are independent; hence, by Chernoff, we get

$$\mathbb{P}\left[\sum_{i\in[t]} \mathbf{X}_i > \frac{2t}{n}\right] \leqslant \exp\left(-\frac{t}{3n}\right). \tag{6.1}$$

We conclude the proof by using the union bound on all n nodes graphs. There are at most 2^{n^2} input graphs G on n nodes. Therefore, for some large enough $t = \Omega(n^3)$, the bound in Eq (6.1) is strictly less than 1; hence, there is a family $\mathcal{S} = \{S_1, \ldots, S_t\}$ such that, for all n-nodes graphs, the probability that $\mathscr{A}[S_i]$ fails for a random $i \in [t]$ is at most $2/n$. ∎

Proof of Lemma 6.7. For a fixed n-sized network. Nodes can locally compute the family $\mathcal{S}$ described in Lemma 6.9 (recall there is not memory or local time constraints on nodes in the NCC model). The node of minimum ID then samples a random index $i \in [|\mathcal{S}|]$ and broadcast it. Since $|\mathcal{S}| \leqslant \text{poly}(n)$, index i can be described in $O(\log n)$ bits. Broadcasting a message to every one takes $O(\log n)$ rounds.

Nodes then know the randomness of every node in G as well as their adjacency list. They can therefore run the streaming phase without any communication. Once this is done, they can emulate $\mathscr{A}$ in $O(TBD/\log n)$ rounds. By Lemma 6.9, it fails with probability at most $1/\text{poly}(n)$. ∎

Corollary 6.10. *There is a distributed randomized algorithm that computes a* $(\Delta + 1)$*-coloring in* $\text{poly}(\log n)$ *rounds of* NCC.

CHAPTER 7

Proof of the DPS Theorem

In this section, we give the CONGEST algorithm for Theorem 6.1.

THEOREM 6.1 (Distributed Palette Sparsification Theorem). *Suppose that each node in an n-vertex graph G with maximum degree $\Delta \gg \log^4 n$ samples $\Theta(\log^2 n)$ colors u.a.r. from $[\Delta+1]$. There is a distributed message-passing algorithm operating on the sparsified graph, that computes a valid list-coloring in $O(\log^2 \Delta)$ rounds, using $O(\log n)$-bit messages. In particular, each node needs to communicate with only $O(\log^4 n)$ different neighbors.*

We will call $\mathbf{L}(v)$ the random list of color sampled by v. We begin by describing how those lists are sampled (Algorithm 17) and prove useful properties of the sparsified graph. Then, in Section 7.2, we give the formal statements for the main three coloring steps of our algorithm and combine them to prove Theorem 6.1. Details for each step are given in the subsequent sections.

Organization of the Chapter. We begin by describing how random list of colors are sampled and the properties it induces for almost-cliques in the sparsified graph. Based of this, we formally describe in Section 7.2 the three main coloring steps of our algorithm. Section 7.3 analyzes the preconditioning step. Section 7.4 describes the adaptations needed to compute a colorful matching while communicating only on the sparsified graph. Section 7.5 contains the most novel part of our algorithm: the augmenting path algorithm.

7.1. Palette Sampling and The Sparsified Graph

Parameters. Assume that $\Delta \geqslant c \log^4 n$, for some sufficiently large universal constant $c > 0$. We define the following parameters[1] for our algorithm:

(7.1)

$$\begin{aligned} \alpha &\stackrel{\text{def}}{=} 500 : && \text{quantify the number of leaves} \\ & && \text{that an augmenting tree must have,} \\ \beta &\stackrel{\text{def}}{=} \lfloor \alpha \cdot c \log n \rfloor : && \text{used to bound the number of sampled colors,} \\ \varepsilon &\stackrel{\text{def}}{=} 10^{-8} : && \text{a small enough constant.} \end{aligned}$$

The constant c is sufficiently large for events w.e.h.p. in β/α to hold, even when we union bound over polynomially many events. It is independent of the constants α and ε. We use the following relation between our parameters:

$$\varepsilon \leqslant 1/\alpha^2 \qquad \text{and} \qquad 2\alpha < 1/(108\varepsilon)\ . \tag{7.2}$$

Palette Sampling. Similar to [ACK19], we see the lists of random colors $\mathbf{L}(v)$ in our algorithm as a source of *fresh random colors*. Whenever a node samples a color in $[\Delta+1]$, it reveals a new color from its list. To simplify the analysis, we partition $\mathbf{L}(v)$ into sub-lists, each used for a different purpose in the algorithm. The main difference with [ACK19] is that lists are larger: $O(\log^2 n)$ colors instead of $O(\log n)$.

ALGORITHM 17. Palette Sampling for a node v.

- $\mathbf{L}_1(v)$: sample $O(\log^2 n)$ colors independently, with repetition, and uniformly at random in $[\Delta + 1]$.
- $\mathbf{L}_2(v) = \mathbf{L}_{2,14}(v) \cup \mathbf{L}_2^*(v)$ where
  - $\mathbf{L}_{2,14}(v)$ is a sequence of t colors sampled independently in $[\Delta + 1]$ with repetition, where $t = O(1/\varepsilon)$ is the number of iterations needed in Algorithm 14 to compute a colorful matching of size $2\alpha \cdot a(K)$.

[1] which we have not attempted to optimize.

  - Sample each color independently in $\mathbf{L}_2^*(v)$ with probability $\gamma_{7.22}\beta/\Delta$ for some constant $\gamma_{7.22}$ defined in Lemma 7.22.
- $\mathbf{L}_3(v) = \{\mathbf{L}_{3,i}(v) = \mathbf{L}_{3,i}^g(v) \cup \mathbf{L}_{3,i}^h(v),\ \text{for } i \in [\beta]\}$ where we sample each color $\chi \in [\Delta+1]$ in $\mathbf{L}_{3,i}^g$ independently with probability $\frac{20\beta}{\Delta}$ and $\mathbf{L}_{3,i}^h$ is a list of β colors sampled in $[\Delta+1]$ with repetition.

By union bound, a fixed color $\chi \in [\Delta+1]$ is included in $\mathbf{L}_2(v)$ with probability $O(\frac{\beta}{\Delta})$ and in $\mathbf{L}_3(v)$ with probability $O(\frac{\beta^2}{\Delta})$. Simple computations shows the following claim.

Claim 7.1. *For every $\chi \in [\Delta+1]$ and $v \in V$, we have that*

$$\frac{\beta^2}{\Delta} \leqslant \mathbb{P}[\chi \in \mathbf{L}(v)] \leqslant O\left(\frac{\beta^2}{\Delta}\right).$$

Since each color is included in $\mathbf{L}_2^*$ and $\mathbf{L}_{3,\ell}$ independently, a simple Chernoff bound proves the following claim:

Claim 7.2. *There exists some number $L_{\max} = O_\varepsilon(\log n)$ such that, w.h.p., we have $|\mathbf{L}_2(v)| \leqslant L_{\max}$ and $|\mathbf{L}_{3,\ell}(v)| \leqslant O(\log^2 n)$ for all $v \in V$ and $\ell \in [\beta]$. Furthermore, each $\mathbf{L}_{3,\ell}^g(v)$ and $\mathbf{L}_{3,\ell}^h(v)$ contains at least 6β different colors.*

Our algorithm will color each $v \in V$ with a color from its list $\mathbf{L}(v)$. Following the observation of [ACK19], edges between nodes with non-intersecting lists can be dropped. A standard argument shows the induced subgraph is sparse with high probability.

Definition 7.3 (The Sparsified Graph)**.** For a graph $G = (V, E)$ and lists $\mathbf{L}(v)$ such as described in Algorithm 17, let $\widetilde{G} = (V, \widetilde{E})$ be the subgraph of G with edges $uv \in E$ such that $\mathbf{L}(u) \cap \mathbf{L}(v) \neq \emptyset$. We call $\widetilde{G}$ the *sparsified graph*. For any set $S \subseteq V$, we denote by $\widetilde{S}$ the induced subgraph $\widetilde{G}[S]$.

Claim 7.4 ([ACK19, Lemma 4.1])**.** *For any graph G, w.h.p., the sparsified graph $\widetilde{G}$ has maximum degree $O(\log^4 n)$.*

Expansion of the Sparsified Almost-Clique. Similarly to algorithms for the LOCAL and CONGEST models, our algorithm uses the almost-clique

decomposition. Before explaining how we even compute it, let us establish some properties of sparsified almost-cliques.

Let K be an ε-almost-clique. The sparsified clique $\widetilde{K}$ is a random graph on (almost) Δ nodes where edges are sampled with probability $O(\log^4 n/\Delta)$. As such, the graph $\widetilde{K}$ has typical properties of random graphs. We denote by $E_G(A, B)$ the set of edges in E_G with one endpoint in A and the other endpoint in B. Contrary to the typical random graphs where edges are sampled independently with some fixed probability, here edges of $\widetilde{K}$ are induced by the sampling of random lists.

Lemma 7.5 (Expansion)**.** *Let K be an almost-clique, and assume $\Delta \geqslant \beta^4$. With high probability, for all subsets $S \subseteq K$ of size at most $|S| \leqslant 3\Delta/4$,*

$$|E_{\widetilde{K}}(S, K \setminus S)| \geqslant |S|\beta^4/3200 \qquad \text{and} \qquad |N_{\widetilde{K}}(S) \cap (K \setminus S)| \geqslant \Omega(|S|).$$

This result is well known when edges are sampled *independently* with probability β^4/Δ. In fact, for our applications in Dist-Stream, NCC or cluster graphs, vertices can simply do that. However, since Theorem 6.1 assumes that communication occurs only on the sparsified graph, we show that it also holds as a result of the sampling of colors rather than edges. Observe that, the edges are induced by the sampled colors, edges incident to the same vertex are not sampled in the sparsified graph independently anymore. This causes technical complications toward proving the expansion properties claimed in Lemma 7.5. To preserve the flow of the chapter, we defer this proof to Section A.2.

Lemma 7.5 implies the following result as any two nodes can reach more than half of the clique in $O(\log \Delta)$ hops.

Corollary 7.6. *The sparsified almost-clique $\widetilde{K}$ has diameter $O(\log \Delta)$.*

Also, observe that two nodes from the same almost-clique that sampled the same color must be within distance 2 in the sparsified graph.

Claim 7.7. *For an almost-clique K, let u and v be two nodes of K and $\chi \in [\Delta + 1]$ be an arbitrary color. Then, with high probability, there exist at least $2\beta^2/5$ nodes $w \in N_G(u) \cap N_G(v) \cap K$ that sample $\chi \in \mathbf{L}(w)$. In particular, if $\chi \in \mathbf{L}(u) \cap \mathbf{L}(v)$ then u and v are at two hops from each other in the sparsified graph $\widetilde{G}$.*

PROOF. Let $q \stackrel{\text{def}}{=} \frac{\beta^2}{\Delta}$. Nodes u and v share at least $(1-2\varepsilon)\Delta$ neighbors in K and each $w \in N_G(u) \cap N_G(v) \cap K$ sampled the color c with probability (at least) q. In expectation, at least $(1-2\varepsilon)\Delta \cdot q \geqslant (4/5)\beta^2$ such w sampled c. Since each w samples its color independently, the classic Chernoff bound applies. At least $(2/5)\beta^2$ shared neighbors sampled c with probability $1 - \exp(-\Omega(\beta^2)) \gg 1 - 1/\text{poly}(n)$. ∎

Analysis of RandomPush. An interesting consequence of expansion in the sparsified almost-cliques is that it allows us to route up to $O(\log^3 n)$ random messages in $O(\log \Delta)$ rounds through random dissemination.

ALGORITHM 18. RANDOMPUSH.
Input: An almost-clique K with x messages.
For each node v that knows at least one message and each incident edge, v picks a random message that it knows and sends it along the edge.

Lemma 7.8. *Let $x \leqslant \beta^3$ messages of $O(\log n)$ bits each known by exactly one node in the sparsified almost-clique $\widetilde{K}$. After $O(\log \Delta)$ iterations of* RANDOMPUSH*, each node in the sparsified almost-clique learns all x messages, with high probability.*

PROOF. Consider a particular message, and for each $i \in [O(\log \Delta)]$, let S_i be the set of nodes in the almost-clique that know this message before iteration i. Let $\overline{S_i} = K \setminus S_i$.

Initially, $|S_1| \geqslant 1$. Each node has degree $\widetilde{\Delta} = \Theta(\beta^4)$, w.h.p., by Claim 7.4. Thus, nodes forward the message to $\Omega(\beta)$ of its neighbors, so $|S_2| \geqslant \beta$, w.h.p. We now show that S_i grows geometrically while $|S_i| \leqslant 3\Delta/4$, then afterwards $\overline{S_i}$ decreases geometrically.

By Lemma 7.5, while $|S_i| \leqslant 3\Delta/4$, there are at least $|S_i|\beta^4/3200$ edges between S_i and $\overline{S_i}$. For an uninformed node v in $\overline{S_i}$, let $d_v^{S_i} = |N_{\widetilde{K}}(v) \cap S_i|$ be its number of informed neighbors. Letting $\mathbf{X}_v$ be the random variable

indicating that v learns the message in this iteration, we have that

$$\mathbb{P}[\mathbf{X}_v = 1] = 1 - \left(1 - \frac{1}{x}\right)^{d_v^{S_i}} \geqslant 1 - \frac{1}{1 + d_v^{S_i}/x} = \frac{d_v^S}{x + d_v^{S_i}}$$

(using $(1-x)^y \leqslant \frac{1}{1+xy}$ for all $x \in [0,1]$ and $y > 0$)

Hence, the expected number of nodes that learn the message is at least

$$\sum_{v \in \overline{S_i}} \frac{d_v^{S_i}}{x + d_v^{S_i}} \geqslant \frac{\widetilde{\Delta}}{x + \widetilde{\Delta}} \cdot \frac{|S_i|\beta^4}{3200\widetilde{\Delta}} \geqslant \Omega(|S_i|) \ ,$$

since by concavity of the function $f(y) = y/(x+y)$, this sum is minimized when the degrees d_v^S are as unevenly distributed as possible, with $\frac{|S_i|\beta^4/3200}{\widetilde{\Delta}}$ nodes satisfying $d_v^S = \widetilde{\Delta}$ and the rest satisfying $d_v^S = 0$.

Since the $\mathbf{X}_v$ are independent, by Lemma 2.2, it holds w.h.p. that $|S_{i+1}| \geqslant (1 + \Omega(1))|S_i|$ while $|S_i| \in [\beta, 3\Delta/4]$. Therefore, after $i \in \Theta(\log \Delta)$ iterations, $|S_i| \leqslant 3\Delta/4$.

The argument after $|S_i| \geqslant 3\Delta/4$ is similar. The nodes in $\overline{S_i}$ have at least $|\overline{S_i}|\beta^4/3200$ edges with S_i (Lemma 7.5 with $\overline{S_i}$). In expectation, $\Theta(|\overline{S_i}|)$ of them get learn the message in each iteration in expectation, and this holds w.h.p. while $|\overline{S_i}| \geqslant \beta$. When $|\overline{S_i}|$ drops below $O(\beta^3)$, since each node in $\overline{S_i}$ is adjacent to $\Omega(\beta^4)$ nodes in S_i, it receives the message $\Omega(\beta)$ times in expectation, and thus receives it with high probability. ∎

Computing the Almost-Clique Decomposition. To compute the almost-clique decomposition, we use the same algorithm as in BCONGEST (Section 4.2). The main difference is that vertices cannot compute the number of friendly edges they are incident to. As [ACK19] showed, it suffices to determine this approximately, which is easily done through sampling.

We describe the algorithm computing the decomposition, Algorithm 19, as a Dist-Stream algorithm for simplicity. It uses a sparse subgraph of G that is independent of the sparsified subgraph induced by the random lists (Definition 7.3). Observe, however, that Algorithm 19 could be implemented by sampling colors and using edges of the conflict graph. We briefly explain why. Two nodes u and v adjacent in the sparsified graph share at least one color χ. To know if they share a large fraction of their neighborhood — i.e., if they are friends (Definition 3.10) — notice that the number of nodes

in $N(u) \cap N(v)$ that sample χ is concentrated, and therefore provides an unbiased estimator for the size of this set. Using a bandwidth compression technique introduced for BCONGEST in Section 4.2, u and v can compare their neighborhoods in $O(1)$ rounds using $O(\log n)$ bandwidth. To know if v has many friends — i.e., if it is popular (Definition 3.11) — notice that its neighboring edges are sampled independently in the sparsified graph. Therefore, a node will detect a lot of friendly edges in the sparsified graph if and only if it is sufficiently popular. We prefer the following less technical and more general algorithm that does not depend on the sparsified graph and could be of independent interest. For completeness, technical details for computing the almost-clique decomposition on the sparsified graph can be found in Section A.3.

ALGORITHM 19. Detecting friendly edges and popular vertices in DistStream.

Input: a graph G with n vertices and maximum degree Δ
Output: a set $E_\delta \subseteq E_G$

When sparsifiying the input graph. Each vertex samples $\mathbf{X}(v) \in [\alpha\Delta]$ uniformly at random where $\alpha = \lceil 16/\delta \rceil$. Each node v then computes

- the set $F(v) = \mathbf{X}(N(v)) \cap [b]$ where $b = \Theta(\delta^{-4} \log n)$, and
- a set $E_s(v)$ of $O(\log n/\delta^2)$ random edges using reservoir sampling.

Communication Phase. Each v performs the following algorithm:

(1) Send $F(v)$ to each neighbor in $E_s(v)$.
(2) Add in E_δ every edge $\{u, v\}$ such that $|F(u) \cap F(v)| \geqslant (1 - 0.75\delta)b/\alpha$.
(3) Let V_δ be the v with at least $(1 - 1.5\delta)\Delta$ edges in $E_\delta \cap E_s(v)$.
(4) Run Algorithm 1

Lemma 7.9. *Algorithm 19 is a $O(\log \Delta)$-round algorithm computing an ε-almost-clique decomposition. It only broadcasts $O(\varepsilon^{-4} \log n)$-bit messages and samples $O(\varepsilon^{-2} \log n)$ edges per node.*

The following lemma states that by sampling edges with probability $\Theta(\log n/\delta^2\Delta)$ edges in its neighborhood, a node can distinguish between it being δ-popular and it not being 2δ-popular. It follows directly from the

Chernoff bound (Lemma 2.3) as the number of sampled edges allows us to estimate w.h.p. the number of friendly edges up to $(\delta/2)\Delta$ by sampling.

Lemma 7.10 (Detecting Popular Nodes)**.** *Let $\delta \in (0, 1/10)$. If edges are sampled in E_s with probability $p = \Theta(\log n/(\delta^2\Delta))$, then, with high probability, for every node u, we have that*

- *if u is δ-popular, it samples at least $(1 - 1.5\delta)\Delta p$ δ-friendly edges in $E_s(u)$;*
- *if u is not 2δ-popular, it samples fewer than $(1 - 1.5\delta)\Delta p$ 2δ-friendly edges in $E_s(u)$.*

Lemma 7.9 follows from Lemmas 4.3 and 7.10 through an argument almost identical to Section 4.2 except for the fact that popular vertices are only detected approximately and that not all edges are known to vertices. The former difference changes only constant factors in the proof of Proposition 4.2. For the latter, as already observed in [ACK19, Lemma 4.10], even if we use only the friendly edges of the sparsified graph, the almost-clique decomposition remains connected. As explained in Lemma 7.5, w.h.p., the sparsified almost-clique has a constant rate vertex expansion, hence diameter $O(\log \Delta)$. Computing the D_i such as in Algorithm 1 thus takes $O(\log \Delta)$ rounds. Note that our construction is slightly different from that of [ACK19] because in Step (3) of Algorithm 1, we add to each cluster all sparse vertices connected to them through at least $(1 - \Theta(\delta))\Delta$ edges. Similarly to popular vertices, it suffices to find such vertices approximately through edge sampling.

7.2. The Distributed Palette Sparsification Theorem

We henceforth assume that lists were sampled as in Algorithm 17 and that vertices computed the almost-clique decomposition. We now go over the three main coloring steps of the algorithm.

Step 1: Preconditioning Almost-Cliques. When we compute the colorful matching or build augmenting trees, nodes might sample $\Theta(\log n)$ random colors within a round. If a node has $\Omega(\Delta)$ neighbors, this might result in all colors being blocked by its external neighbors. To circumvent this

issue, we use standard techniques from distributed coloring to strengthen guarantees given by the almost-clique decomposition (Proposition 3.6).

THEOREM 7.11. *Let $\varepsilon \in (0, 1/3)$ be a constant independent of n and Δ, and η be any number (possibly depending on n and Δ) such that $\Delta/\eta \geqslant c \log n$ for a large enough constant $c > 0$. There exists an algorithm computing a partial coloring of G where all uncolored nodes are partitioned in ε-almost-cliques $K_1, \ldots, K_t$ for some t such that for every $v \in K_i$ for all $i \in [t]$ has*

$$|N(v) \cap \bigcup_{j \neq i} K_j| \leqslant e_{max} = \Delta/\eta \tag{7.3}$$

Furthermore, the algorithm runs in $O(\log \Delta + \log \eta)$ rounds, uses $O(\eta \log n)$ colors from lists $\mathbf{L}_1$, and samples $O(\eta \log n)$ edges per node.

Note that the bound on the external degree given by Eq (7.3) is much stronger than the one from the classical almost-clique decomposition. Henceforth, we assume we are given the coloring and decomposition of Theorem 7.11 with maximum external degree

$$e_{max} \stackrel{\text{def}}{=} \Delta/\eta \quad \text{where} \quad \eta \stackrel{\text{def}}{=} \max(160\alpha\beta, L_{\max}/\varepsilon) \ , \tag{7.4}$$

where $L_{\max} = O(\varepsilon^{-2} \log n)$ is the upper bound on the size of lists $\mathbf{L}_2$ of Claim 7.2. We prove Theorem 7.11 in Section 7.3.

Step 2: Colorful Matching. Recall that an almost-clique K is said to have a k-colorful matching if there is a set M of k repeated colors in K. The clique palette $L_\varphi(K) = [\Delta + 1] \setminus \varphi(K)$, with respect to φ, is the set of colors not used in K. Let us recall the accounting lemma from Section 4.6:

Lemma 4.19 (Accounting Lemma I)**.** *Suppose K contains a k-colorful matching in φ. For every $v \in K$, we have*

$$|L_\varphi(v) \cap L_\varphi(K)| \geqslant \Delta - \deg(v) + |(N(v) \cup K) \setminus \operatorname{dom} \varphi| - a(v) + k \ . \tag{4.5}$$

By Lemma 4.19, a promising vertex is one that has at least as many colors available in the clique palette as there are uncolored vertices in its almost-clique.

Definition 7.12. Let φ be a (partial) coloring, K be an almost-clique with a colorful matching of size M. We say v is *promising* if it satisfies $a(v) \leqslant M$. Otherwise it is *unpromising*.

We explain in Section 7.4 how we compute a matching large enough to ensure that a very large fraction of the vertices are promising. When $a(K) \gg \log n$, the algorithm is an adaptation of the BCONGEST algorithm. In the case where $a(K) \leqslant O(\log n)$, we use an existential argument from [ACK19] along with careful routing.

THEOREM 7.13. *Let $c > 0$ be a constant such that $c < 1/(108\varepsilon)$. There is a $O(\log \Delta)$-round algorithm computing a colorful matching of size at least $c \cdot a(K)$ with high probability in all cliques K of average anti-degree $a(K) \geqslant 1/(2\alpha)$.*

Corollary 7.14. *After Step 2, every almost-clique contains at most Δ/α* unpromising *nodes.*

PROOF. In a almost-clique K with $a(K) \leqslant 1/(2\alpha)$, at most $|K|/(2\alpha)$ nodes have anti-degree at least 1, by Markov inequality. In a clique K with $a(K) \geqslant 1/(2\alpha)$, we compute a colorful matching M of size $2\alpha \cdot a(K)$. By Markov inequality, at most $|K|/(2\alpha)$ nodes have anti-degree more than $|M|$. In both cases, at most $|K|/(2\alpha) \leqslant 2\Delta/(2\alpha) = \Delta/\alpha$ nodes are unpromising. ∎

Step 3a: Reducing the number of uncolored nodes. Before we apply the augmenting path algorithm, we reduce the number of uncolored in vertices each almost-clique to $\Delta/(\alpha\beta)$. As explained in Lemma 4.33, when nodes try colors from their palettes, they get colored with constant probability. In our setting, nodes cannot directly sample colors from their palette as they must use colors from the lists they sampled in Algorithm 17. If they have large enough palette though, (uninformed) sampling $O(\log n)$ colors in $[\Delta + 1]$ is enough to find one in their palette with constant probability.

Lemma 7.15. *There exists a $O(\log \log n)$-round algorithm such that, with high probability, the number of uncolored nodes in each almost-clique is afterwards at most $\Delta/(\alpha\beta)$. Furthermore, nodes only use $O(\log n \cdot \log \log n)$ fresh random colors from $\mathbf{L}_1$.*

PROOF. Sample vertices in a set U independently w.p. $1/(4\alpha\beta)$. By Chernoff, w.h.p., every vertex in K has at least $\Delta/(8\alpha\beta)$ uncolored neighbors in U and U has size at most $\Delta/(2\alpha\beta)$. Vertices of U remain inactive at this step of the algorithm. So every uncolored vertex outside of U has at least $\Delta/(8\alpha\beta)$ available colors throughout. As such, after sampling $16\alpha\beta$ colors in $[\Delta+1]$ with repetition, the probability that v does not find an available color is at most $\left(1-\frac{\Delta/8\alpha\beta}{\Delta+1}\right)^{16\alpha\beta} < 1/2$. If a vertex finds an available color, with probability $1/3$, it tries the first one that it sampled, i.e., it broadcasts it and retains it if no neighbor with smaller identifier also tries that color. Observe that the first available color sampled by v is uniformly distributed in $L_\varphi(v)$. As such, in this algorithm, vertices try a uniform color from their palette $L_\varphi(v) = [\Delta+1] \setminus \varphi(N(v))$ with probability at most $1/3$ and at least $1/6$. By the same analysis as in Lemma 4.33 (with $\text{List}(v) = [\Delta+1]$ and $\alpha = 1$), we get that the number of uncolored vertices in $K \setminus U$ decreases by a constant factor. So after $O(\log(\alpha\beta)) = O(\log\log n)$ iterations, w.h.p., $K \setminus U$ contains at most $\Delta/(2\alpha\beta)$ uncolored vertices. Along with the vertices of U, at most $\Delta/(\alpha\beta)$ vertices remain uncolored after this step. ∎

Step 3b: Finishing the coloring with augmenting paths. Now that the number of uncolored nodes is small, we resort to the new technique of coloring with *augmenting paths* outlined in Section 6.2.2.3.

THEOREM 7.16. *Assume all almost-cliques have at most $\Delta/(\alpha\beta)$ uncolored nodes and at most Δ/α unpromising nodes. There is a $O(\log\Delta)$-round algorithm* AUGMENTINGPATH *that colors a constant fraction of the nodes in each almost-clique with high probability.*

We give a detailed description of the AUGMENTINGPATH algorithm and the proof of Theorem 7.16 in Section 7.5. We conclude this section with the proof of our main theorem.

Proof of Theorem 6.1. We precondition almost-cliques (using Theorem 7.11) with $\eta = O(\log n)$ (Eq (7.4)) in $O(\log\Delta)$ rounds and using $O(\log^2 n)$ random colors. The colorful matching requires $O(\log\Delta)$ rounds (Theorem 7.13) and almost-cliques have at most Δ/α unpromising nodes

(Corollary 7.14). For $O(\log\log n) \leqslant O(\log\Delta)$ rounds, nodes try random colors from their palettes. All almost-cliques are left with $\Delta/(\alpha\beta)$ uncolored nodes (by Lemma 7.15). We run AUGMENTINGPATH for $O(\log\Delta)$ times. Each time, the number of uncolored nodes decreases by a constant factor with high probability (Theorem 7.16). Overall, we use $O(\log^2\Delta)$ rounds to complete the coloring.

7.3. Preconditioning the Almost-Clique Decomposition

Assume we computed an ε'-almost-clique decomposition using Algorithm 19 for $\varepsilon' = \varepsilon/4$. In this section, we use this decomposition to compute the partial coloring described in Theorem 7.11.

THEOREM 7.11. *Let $\varepsilon \in (0, 1/3)$ be a constant independent of n and Δ, and η be any number (possibly depending on n and Δ) such that $\Delta/\eta \geqslant c\log n$ for a large enough constant $c > 0$. There exists an algorithm computing a partial coloring of G where all uncolored nodes are partitioned in ε-almost-cliques $K_1, \ldots, K_t$ for some t such that for every $v \in K_i$ for all $i \in [t]$ has*

$$|N(v) \cap \bigcup_{j \neq i} K_j| \leqslant e_{max} = \Delta/\eta \tag{7.3}$$

Furthermore, the algorithm runs in $O(\log\Delta + \log\eta)$ rounds, uses $O(\eta\log n)$ colors from lists $\mathbf{L}_1$, and samples $O(\eta\log n)$ edges per node.

Let us first recall two key results from Chapter 3: if every vertex gets activated and tries a uniform color in $[\Delta+1]$, then ζ-sparse vertices where $\zeta \gg \log n$ get $\Omega(\zeta)$-slack (Proposition 3.15); dense vertices have sparsity proportional to their external degree (Part 3). As in Chapter 3, when vertices have slack linear in their uncolored degree, they can be colored fast. Implementing MULTICOLORTRIAL on the sparsified graph requires some minor modifications as vertices cannot directly sampled colors in their palette. Since we apply SLACKCOLOR only to $\widetilde{\Omega}(\Delta)$-sparse vertices, it suffices to sample $\text{poly}(\log n)$ times more colors than in Algorithm 3. We obtain the following lemma and defer the proof to Section A.4 to preserve the flow of the chapter.

Lemma 7.17. *Let $\delta > 0$ be a real number, G be an n-vertex graph with maximum degree $\Delta \gg \delta^{-1} \log^{1.1} n$ and φ a partial coloring of G such that every vertex has $|L_\varphi(v)| \geqslant \deg_\varphi(v) + \delta\Delta$ and $\deg_\varphi(v) \leqslant (\delta/2)\Delta$. If every vertex independently samples a list $\mathbf{L}_1(v)$ of $O(\delta^{-1} \log n)$ colors from $[\Delta+1]$ with replacement, there exists a $O(\log^* n)$-round randomized algorithm that $\mathbf{L}_1$-list-colors G with high probability by communicating only the sparsified graph.*

In Lemma 7.17, we assume that $\deg_\varphi(v) \leqslant (\delta/2)\Delta$ so that we do not need to run Phase (1) of Algorithm 4. In Phase (1), vertices try uniform colors from their palette to reduce their uncolored degree to at most $(\delta/2)\Delta$. Since we will apply Lemma 7.17 with $\delta = O(1/\log n)$, we do not rely on the argument of Lemma 3.21 for Phase (1). We still try one uniform color from the vertices' palette, but do so slightly more carefully — through an argument akin to that of Lemmas 4.33 and 7.15. In Algorithm 20, Phase (1) takes $O(1)$ rounds in Step 3 (and $\delta = \Theta(1)$) but $O(\log \log n)$ rounds for extroverted nodes from introverted almost-cliques at Step 4 (where $\delta = \Theta(1/\beta)$).

In Theorem 7.11, we get rid of high external degree nodes using such nodes have a lot of slack. Algorithm 20 carefully generates slack to color all nodes of high external degree.

First, we claim that high-external-degree nodes can be easily detected by randomly sampling edges.

Claim 7.18. *There is an algorithm partitioning the dense nodes into two classes:* extroverted *nodes of external degree at least Δ/η, and* introverted *nodes of external degree at most $\Delta/(2\eta)$. The algorithm samples $O(\eta \log n)$ edges per node.*

Proof. Let $e_{max} = \Delta/\eta$. During the streaming phase[2], a node samples edges with probability $p \stackrel{\text{def}}{=} \frac{\eta\beta}{\Delta}$. Once the nodes have computed the almost-clique decomposition, they know which edges connect them to external neighbor. If a node sampled fewer than 0.75β edges to external neighbors, it classify itself as introvert; otherwise, as extrovert.

[2]Again, we present this as a Dist-Stream algorithm for the sake of simplicity but it could be emulated through sampling of colors. See Lemma 7.10.

- Consider a node with external degree at most $e_{max}/2$. In expectation, it samples at most $pe_{max}/2 \leqslant \beta/2$ edges to external neighbors. By Chernoff, it samples fewer than 0.75β edges with high probability. Nodes with external degree less than $e_{max}/2$ are classified as introverts.
- Consider an extroverted node, i.e., with external degree at least e_{max}. In expectation, it samples at least $pe_{max} \geqslant \beta$ edges to external neighbors. By Chernoff, it samples at least 0.75β edges with high probability. All nodes with external degree more than e_{max} are classified as extrovert, w.h.p.

Nodes with external degree between $e_{max}/2$ and e_{max} can be arbitrarily classified as introvert or extrovert. ∎

Based on the classification of vertices obtained from Claim 7.18, we classify almost-cliques depending on how many vertices of each types they have.

Definition 7.19 (Extrovert/Introvert)**.** An almost-clique is *extrovert* if it has more than $3\varepsilon'\Delta$ extroverted nodes, and *introvert* otherwise.

ALGORITHM 20. The algorithm preconditioning almost-cliques.

Input: an ε'-almost-clique decomposition $V_{sparse}, K_1, \ldots, K_t$ for some t.

Output: a decomposition such as in Theorem 7.11.

(1) In each almost-clique K_i, let $W_i \subseteq K_i$ be its set of extroverted nodes. Each almost-clique learns if it is introvert or extrovert in $O(\log \Delta)$ rounds by aggregating the size of W_i on a BFS tree. Denote by J the set of indices $i \in [t]$ such that K_i is extrovert.
(2) SLACKGENERATION with $p = 1/20$ in $G[V_{sparse} \cup \bigcup_{i \in J} K_i]$
(3) Let $V' = V_{sparse} \cup \bigcup_{i \notin J} W_i \cup \bigcup_{i \in J} (K_i \setminus W_i)$ be the set containing sparse nodes, extroverted nodes from introverted almost-cliques and introverted nodes from extroverted almost-cliques. All nodes in V' have slack $\Omega(\varepsilon'^2\Delta)$ and can be colored in $O(\log \Delta)$ rounds by SLACKCOLOR.

(4) Run randomized color trial for $O(\log \eta)$ rounds in extroverted almost-cliques. The number of uncolored nodes left in each W_i for $i \in J$ is at most $O(\Delta/\eta)$. Complete the coloring of extroverted almost-cliques using SlackColor.

Proof of Theorem 7.11. After Step 2, nodes in V_{sparse} have $\Omega(\varepsilon'^2\Delta)$ permanent slack and extroverted nodes have $\Omega(e_{max}) = \Omega(\Delta/\eta)$ permanent slack (by Proposition 3.15 and Part 3). Let $J \subseteq [t]$ the set of extroverted almost-cliques. In Step 3, we color nodes of V' where

$$V' = V_{sparse} \cup \bigcup_{i \notin J} W_i \cup \bigcup_{i \in J} (K_i \setminus W_i) \ .$$

Dense nodes of V' receive slack from their inactive neighbors in $V \setminus V'$.

- An extroverted nodes $v \in W_i$ in some introverted almost-clique K_i with $i \notin J$ has $|N(v) \cap (K \setminus W_i)| \geqslant (1-3\varepsilon')\Delta$ introverted neighbors in K_i. Note that none of them was colored in Step 2.
- A introverted node $v \in K$ in an extroverted almost-clique K_i with $i \in J$ has at least $|N(v) \cap W_i| \geqslant (3\varepsilon' - 2\varepsilon')\Delta = \varepsilon'\Delta$ extroverted neighbors in K_i. Each such neighbor gets colored in Step 2 with probability at most $1/20$; hence, w.h.p. at least $0.9\varepsilon'\Delta$ are uncolored.

Adding sparse nodes, all nodes in V' have slack $\gamma\Delta$, where γ is a small enough universal constant. Every vertex in V' gets activated with probability $1/3$. If a vertex is activated, it samples $O(1/\gamma)$ colors for a large enough hidden constant to find one available with probability at least $1/2$. If it found an available color, it tries to adopt it. As such, the same argument as in Lemmas 4.33 and 7.15 applies and, w.h.p., the uncolored degrees in V' decrease by a constant factor. After repeating this $O(\log 1/\gamma) = O(1)$ times, the uncolored degrees in V' are small enough to apply Lemma 7.17 with $\delta = \gamma$. It colors all the remaining nodes in V' in $O(\log \Delta)$ rounds and $O(\log n)$ fresh colors with high probability.

Let us now extend the coloring to all extroverted almost-cliques. Since they have $\gamma \cdot \Delta/\eta$ slack, for some small constant γ depending only on ε, by an argument similar to Lemmas 4.33 and 7.15, w.h.p., we reduce the uncolored

degree of each node to $(\gamma/2) \cdot \Delta/\eta$ by having vertices, for $O(\log \eta)$ rounds, get activated w.p. $1/3$ and try one available color. It suffices to samples $O(\eta)$ colors per random color trials to ensure that a vertex finds an available color with at least $1/2$. So each vertex uses $O(\eta \cdot \log\log n)$ random colors from $\mathbf{L}_1$ in the reduction step. Nodes can now be colored by SLACKCOLOR in $O(\log \Delta)$ rounds and using $O(\eta \log n)$ colors (Lemma 7.17 with $\delta = \gamma/\eta$).

We now prove that our coloring verifies the properties of Theorem 7.11. The crux is that the only uncolored nodes remaining are introverted nodes in introverted almost-cliques. For each introverted almost-clique K in the ε'-almost-clique decomposition, we get an ε-almost-clique K' with the claimed properties by simply removing colored nodes. This is because K' is an ε'-almost-cliques from which we removed at most $3\varepsilon'\Delta$ extroverted nodes. Hence, the upper bound $|K'| \leqslant (1+\varepsilon')\Delta \leqslant (1+\varepsilon)\Delta$ trivially holds (recall $\varepsilon' = \varepsilon/4$) and for all $v \in K'$, we have $|N(v) \cap K'| \geqslant (1-4\varepsilon')\Delta = (1-\varepsilon)\Delta$. Furthermore, all nodes of K' are introverted, therefore they are connected to at most Δ/η nodes in other almost-cliques. Note however that they can be connected to $\varepsilon\Delta$ colored nodes (as they include sparse nodes and extroverted nodes from K). ∎

7.4. Colorful Matching

In this section, we show the following theorem:

THEOREM 7.13. *Let $c > 0$ be a constant such that $c < 1/(108\varepsilon)$. There is a $O(\log \Delta)$-round algorithm computing a colorful matching of size at least $c \cdot a(K)$ with high probability in all cliques K of average anti-degree $a(K) \geqslant 1/(2\alpha)$.*

Throughout this section, we fix a almost-clique K and fix the colors *used and sampled* outside of K adversarially. At the beginning of this step, nodes of K are uncolored.

Recall the BCONGEST algorithm from Section 4.6, in which vertices repeatidly try colors in $[\Delta+1]$ and retain it only if it is repeated in the almost-clique. We apply the same technique but modify slightly the definition of $\text{Avail}_{\varphi,D}$ to also exclude colors *sampled* by dense external neighbors. More precisely, for a set D of colors and anti-edges F in an almost-clique

K, in this section, we define

(7.5)

$$\text{Avail}_{\varphi,D}(F) = \sum_{\{u,v\}\in F} |(L_\varphi(u)\cap L_\varphi(v)\cap D)\setminus \mathbf{L}_2\left(\bigcup_{K'\neq K}(N(u)\cup N(v))\cap K'\right)| \,.$$

to be the number of colors that anti-edges can adopt in D without conflicting with external neighbors, including all possible colors in $\mathbf{L}_2$ that external neighbors might use to color themselves. We can afford to lose so many colors because of preconditioning and (in contrast to Section 7.5), at this stage $\Omega(\Delta)$ colors are still available in the almost-clique. The following lemma states that, at the beginning of this step, many edges have many available colors regardless of conditioning of random variables outside of K.

Lemma 7.20. *Let $D = [\Delta+1]$ and F be the set of all anti-edges in K. For any (possibly adversarial) conditioning outside of K, we have $\text{Avail}_D(F) \geqslant a(K)\Delta^2/3$.*

Proof. For a fixed edge $e \in F$, we bound from below its number of available colors. Each *colored* neighbor blocks at most one color. Observe that to compute a colorful matching, uncolored nodes use only colors sampled in $\mathbf{L}_{2,14}$ and $\mathbf{L}_2^*$. By Claim 7.2, an *uncolored* neighbor in another almost-clique blocks at most $L_{\max} = O(\log n)$ colors. Since a node in K has at most $\varepsilon\Delta$ colored neighbors (necessarily outside K) and at most $e_{max} = \Delta/\eta$ neighbors in other almost-cliques (by Eq (7.3) of Theorem 7.11 and Eq (7.4)), we have

$$\text{Avail}_D(e) \geqslant \Delta + 1 - 2\varepsilon\Delta - 2L_{\max}\cdot\Delta/\eta \geqslant (1-4\varepsilon)\Delta \,.$$

Summing over all edges, we get

$$\text{Avail}_D(F) = \sum_{e\in F}\text{Avail}_D(e) \geqslant \frac{a(K)|K|}{2}(1-4\varepsilon)\Delta \geqslant a(K)\Delta^2/3 \,. \qquad \blacksquare$$

This means that even with our more restrictive condition on colors, the BCONGEST algorithm from Section 4.6 would find a $\Omega(a(K)/\varepsilon)$-sized colorful matching. To implement the algorithm, vertices begin by broadcasting in $O(\log\Delta) = O(\log\Delta)$ rounds all colors from $\mathbf{L}_2$. Then, in Algorithm 14, the only step that requires more than trying a uniform color in $[\Delta+1]$ is to verify if that color is sampled exactly two times in K.

Lemma 7.21. *There is a randomized $O(1)$-round algorithm that, w.h.p., let every vertex in K learn if its color is used only once, exactly twice or at least three times in K. Furthermore, in $O(\log \Delta)$ rounds, all vertices of K can learn the total number of repeated colors in K.*

PROOF. Every vertex samples a uniform color $\boldsymbol{\chi}(v) \in [\Delta + 1]$ (e.g., by taking the first color in $\mathbf{L}_{2,14}$). For each $i \in [(\Delta + 1)/\beta]$, call X_i the set of vertices with $\boldsymbol{\chi}(v) \in [(i-1)\beta + 1, i\beta]$. Since every vertex joins X_i independently with probability β, w.h.p., every X_i has diameter at most two in the sparsified almost-clique (see Lemma 4.9 or Claim 7.7). Each vertex broadcasts its color $\varphi(v)$, and its message is received by a neighbor of X_i such that $\varphi(v) \in [(i-1)\beta + 1, i\beta]$. In $O(\log \Delta)$ rounds, each group can aggregate for each of its $\beta = O(\log n)$ colors the three smallest identifier of vertices with that color. Here, we assume that vertices of K were relabeled using unique identifiers in $\{1, 2, \dots, |K|\}$, which can be done in $O(\log \Delta)$ rounds once, say, after computing the decomposition. Each color group thus aggregates $O(\beta \log \Delta)$ bits over a depth two BFS-tree, which requires $O(\log \Delta)$ rounds. Once the aggregation in each X_i is finished, by broadcasting one $O(\log n)$ bit message, they inform vertices of how many times their color is used in K (i.e., once, twice or more). By aggregating over a BFS-tree spanning K in the sparsified almost-clique, we compute in $O(\log \Delta)$ rounds the number of repeated colors in K. ∎

By Lemma 4.21, when $a(K) \geqslant \beta$, Algorithm 14 computes a colorful matching of size $c \cdot a(K)$ in $O(c + \log \Delta)$ rounds with high probability.

When $a(K)$ is small. If $a(K) \leqslant O(\log n)$, Lemma 4.21 does not guarantee a high probability of success. We explain how to compute a large enough matching in almost-cliques with small anti-degree. Note that nodes can count the number of anti-edges in $G[K]$ or in the colorful matching in $O(\log \Delta)$ rounds.

Intuitively, since $a(K) \geqslant 1/(2\alpha)$, if we repeat the previous procedure $\Theta(\alpha^{-1} \log n)$ times, the probability that they all fail to find a large enough colorful matching is at most $1/\text{poly}(n)$. This implies that even when $a(K)$ is a small constant, a colorful matching of size $\Theta(a(K))$ exists. This was, in fact, already shown in the first palette sparsification theorem.

Lemma 7.22 ([ACK19, Lemma 3.2])**.** *Let K be a ε-almost clique, $D \subseteq [\Delta+1]$ and F a subset of anti-edges in K. Fix any partial coloring φ where nodes of K are uncolored and $Avail_{\varphi,D}(F) \geqslant a(K)\Delta^2/3$. Suppose each node sample colors in $[\Delta+1]$ independently with probability $q \stackrel{def}{=} \gamma_{7.22}\frac{\beta}{\Delta}$ for some constant $\gamma_{7.22} = \gamma_{7.22}(\varepsilon) > 0$ (depending on ε but not n nor Δ). Then there exists a colorful matching of size at least $a(K)/(414\varepsilon)$ with high probability.*

Again, our definition of Avail (as in Eq (7.5)) is stronger than the one of [ACK19] because it removes colors sampled by active external neighbor. Regardless of that, Lemma 7.20 shows that we have a large number of available colors, which is the only requirement for the proof of [ACK19].

ALGORITHM 21. MATCHING

Input: for almost-cliques K with $a(K) < \beta$ in parallel.

Output: a colorful matching of size $ca(K)$.

(1) Each $v \in K$ samples each $\chi \in [\Delta+1]$ into $\mathbf{L}^*_2(v)$ independently with probability $q \stackrel{\text{def}}{=} \gamma_{7.22}\frac{\beta}{\Delta}$.

(2) Each node computes $S_v \stackrel{\text{def}}{=} \mathbf{L}^*_2(v) \setminus \mathbf{L}^*_2(V \setminus K)$, the set of colors that do not collide with those of external neighbors. Let

$$\widehat{S}_v \stackrel{\text{def}}{=} \{\chi \in S_v : \exists u \in A(v) \text{ such that } \chi \in S_u\}$$

be the colors of v sampled by at least one anti-neighbor of v.

(3) We count the number of candidate anti-edges in K: the number of pairs (uv, χ) where uv is an anti-edge and $\chi \in \widehat{S}_v \cap \widehat{S}_u$. If there are more than $U \stackrel{\text{def}}{=} 4\gamma^2_{7.22}ca(K)\beta^2$ candidate anti-edges, we select a set of colors D such that the number of candidate edges with that color is at least D and at most $O(\beta^3)$. If there are less than U candidate edges, we let $D = [\Delta+1]$.

(4) Each node v forms messages $(\mathsf{ID}(v), \chi)$ for each $\chi \in \widehat{S}_v \cap D$. Use RANDOMPUSH to disseminate all messages $(\mathsf{ID}(v), \chi)$ within K. Each node v with $\chi \in \widehat{S}_v \cap D$ forms a message $(\mathsf{ID}(u), \mathsf{ID}(v), \chi)$ for each anti-neighbor u with $\chi \in \widehat{S}_u \cap D$.

Use RandomPush to disseminate all messages $(\mathsf{ID}(u), \mathsf{ID}(v), \chi)$ within K.

(5) Compute locally the colorful matching using anti-edges disseminated in the previous step.

Lemma 7.23. *Let $K \in (0, 1/(18\varepsilon))$ be a constant. Consider all almost-cliques with $1/(2\alpha) \leqslant a(K) \leqslant \beta$. If nodes sample each color independently with probability $q = \gamma_{7.22}\frac{\beta}{\Delta}$ where $\gamma_{7.22}$ is the constant from Lemma 7.22, then in each clique K, with high probability, there exists a colorful matching that does not conflict with nodes on the outside. Moreover, Algorithm 21 finds this matching in $O(\log \Delta)$ rounds.*

Proof. We first explain how nodes compute $\widehat{S}_v$. Each node v starts by broadcasting $\mathbf{L}_2^*(v)$ in $O(\log \Delta)$ rounds (since $|\mathbf{L}_2^*(v)| = O(\log n)$). We run a BFS for each color; two hops suffice by Claim 7.7. To learn $\widehat{S}_v$, we count the number of nodes that sample each color using the BFS trees. A node needs to communicate over an edge only if both endpoints sampled the same color. Hence, we send at most $O(\log n \cdot \log \Delta)$ bits on an edge for each round of the BFS. In $O(\log \Delta)$ rounds, all nodes $v \in K$ know for each $\chi \in S_v$ how many other nodes $u \in K$ have $\chi \in S_u$. If this is more nodes than they know from their neighborhood, they must have an anti-neighbor with that color.

A candidate edge is a pair (uv, χ) where u and v are anti-neighbor and χ is a color such that $\chi \in \widehat{S}_v \cap \widehat{S}_u$. Namely, we could add edge uv to the colorful matching using color χ. For each color, we elect a leader amongst nodes that sampled that color. Using aggregation on 2-hops BFS trees, each leader learns the number of candidate edges for its color in $O(\log \Delta)$ rounds. We then run a BFS in the whole clique K and aggregate the total number of candidate edges.

Suppose that the number of candidate edges is at most $U \stackrel{\text{def}}{=} 4\gamma_{7.22}^2 \cdot ca(K)\beta^2 = O(\beta^3)$. For each candidate edge (uv, χ), we craft two messages $(\mathsf{ID}(u), \chi)$ and $(\mathsf{ID}(v), \chi)$. Note that a node can be in $O(\beta^2)$ anti-edges. The total number of messages is $O(\beta^3)$; hence, can be disseminated to all nodes in $O(\log \Delta)$ rounds by RandomPush (Lemma 7.8). After this

step, a node v knows all candidate edges it belongs to. We run one extra RandomPush for all nodes to know all candidate edges. By Lemma 7.22, a colorful matching of size $ca(K)$ must exist, and nodes can find it with local computations.

Suppose now that the number of candidate edges is more than U. By the same argument as in Claim 7.4, each node is contained in at most $2\gamma_{7.22}^2\beta^2$ candidate edges with high probability. Therefore, the colorful matching can be computed by a simple greedy algorithm from any set F of at least U candidate edges: start with an empty matching; as long as the matching has not size $ca(K)$, insert an arbitrary edges from F into the matching and remove adjacent candidates edges from F. When we add an edge to the matching, we remove at most $4\gamma_{7.22}^2\beta^2$ edges from F. Since we assumed F contained at least $U = 4\gamma_{7.22}^2 \cdot ca(K)\beta^2$ anti-edges, the algorithm always finds a colorful matching of $ca(K)$ edges. To select and disseminate a set F of anti-edges, we select a subset D of the colors, enough to have U candidate edges but small enough to be able to disseminate the candidate edges with RandomPush. Using the same process as when the number of candidate edges is small, but using only colors of D, we can disseminate all selected candidate edges in $O(\log \Delta)$ rounds and compute the colorful matching locally.

A simple recursive algorithm on the BFS tree spanning K selects a subset D of the colors such that the number of candidate edges with these colors is at least U and at most $O(\beta^3)$. Recall that each color has a unique leader which knows the number of candidate edges for its color. We say a subtree holds candidate edges with color χ if the leader for color χ belongs to this subtree. Note that when we compute the total number of candidate edges, each node learns the number of candidate edges held their subtree. Let v be the root of the BFS tree spanning K and $x_1, \dots, x_t$ the number of candidate edges held by each subtree. Let i be the smallest index in $[t]$ such that $\sum_{j \leqslant i} x_j \geqslant U$. We select all colors whose leaders are in subtrees 0 to $i-1$. We selected $\sum_{j<i} x_j$ candidate edges. We recursively run the algorithm on the i-th subtree to find $U - \left(\sum_{j<i} x_j\right)$ candidate edges. It is clear that we select at least U candidate edges. We do not select more than $O(\beta^3)$ edges because each color group contains at most $O(\beta^2)$ edges.

It is easy to see that the algorithm explore the tree top to bottom once as information can propagate independently in each subtree. In $O(\log \Delta)$ rounds each leader knows if its color was selected. Each leader relays the information to nodes of its group in $O(\log \Delta)$ rounds. At this point each $v \in K$ knows which color belongs to a selected candidate edge, i.e., colors such that $\chi \in \widehat{S}_v \cap D$. ∎

7.5. Augmenting Paths

This section is dedicated to the central argument of Theorem 6.1.

THEOREM 7.16. *Assume all almost-cliques have at most $\Delta/(\alpha\beta)$ uncolored nodes and at most Δ/α unpromising nodes. There is a $O(\log \Delta)$-round algorithm* AUGMENTINGPATH *that colors a constant fraction of the nodes in each almost-clique with high probability.*

We first give a high-level description of the complete algorithm in Section 7.5.1. Section 7.5.2 contains the proofs related to the first part of the algorithm: growing the augmenting trees. We call the phase of recoloring augmenting paths *harvesting the trees* and address it in Section 7.5.3.

The coloring of the graph before we call the algorithm is called φ. The analysis focuses on a single almost-clique K and shows that a constant fraction of its uncolored vertices get colored with high probability. Theorem 7.16 follows by union bound. We use the letter k to denote the number of uncolored vertices in K.

Remark 7.24. We can assume without loss of generality that the colorful matching, i.e., the number of repeated colors in K, is at most $3\varepsilon\Delta$. Indeed, since every vertex has $a(v) \leqslant 2\varepsilon\Delta$, by the accounting lemma (Lemma 4.19), every vertex in K then has $\varepsilon\Delta$ slack when $3\varepsilon\Delta$ colors are repeated in K. As such, vertices of K can be colored by SLACKCOLOR like sparse vertices. Observe that vertices can test for that eventuality in $O(\log \Delta)$ rounds by Lemma 7.21.

7.5.1. Detailed Description of the Algorithm.

ALGORITHM 22. AUGMENTINGPATH.
Input: A partial coloring φ such that each almost-clique has $|K \setminus \operatorname{dom}\varphi| < \Delta/(\alpha\beta)$ uncolored nodes, at most Δ/α unpromising nodes, and each $v \in K$ is connected to at most e_{max} nodes in other almost-cliques.

For each almost-clique K, do *in parallel*:

(1) Count the number of uncolored nodes $k = |K \setminus \operatorname{dom}\varphi|$.
(2) $F =$ GROWTREE (Algorithm 23).
(3) if $k \geqslant \beta$, run HARVEST(k, F) for high k (Algorithm 24).
(4) if $k < \beta$, run HARVEST(k, F) for small k (Algorithm 25).

Definition 7.25 (Augmenting Path)**.** Let $P = u_1u_2, \ldots, u_t$ be a path in K where u_1 is uncolored and u_i has color χ_i for each $2 \leqslant i \leqslant t$. We say it is an *augmenting path* if u_t, the colored endpoint of P, knows a color $\chi \neq \chi_t$ such that if we recolor every node u_i using color χ_{i+1} for $i \in [t-1]$ and u_t using χ, the coloring of the graph remains proper.

From an uncolored node $u = u_1$ in an almost-clique with one uncolored nodes. Start with the path $P = u_1$ and as long as $P = u_1, \ldots, u_i$ is not augmenting, do the following: u_i samples a color $\chi \in [\Delta + 1]$, if χ is not used in the almost-clique P is an augmenting path; if χ is used by a node $u_{i+1} \in C$, add u_{i+1} to the end of P and repeat this process. Unfortunately, this algorithm is not fast enough as each time we extend P, we find an augmenting path with probability $1/\Delta$. Hence, we need to spend $\Omega(\Delta)$ rounds exploring the almost-clique before finding the one available color. To speed-up this process, we grow a tree of many augmenting paths.

Definition 7.26 (Augmenting Tree/Forest)**.** An *augmenting tree* is a tree such that each root-to-leaf path is augmenting, provided the leaf finds an available color. An *augmenting forest* is a set of disjoint augmenting trees.

Say we computed an augmenting forest such that all trees have $\Omega(\Delta/k)$ leaves. Since a leaf finds an available color in $L_\varphi(K)$ with probability $\Omega(k/\Delta)$, each tree contains an augmenting path with constant probability.

Technical challenges. This process can fail in several ways.

(1) We need to show a constant probability of progress *for each uncolored node*. It is not enough to have that all leaves in K recolor their path with probability $\Omega(k/\Delta)$. We need to show that (with constant probability) *each tree* finds a leaf with which it can recolor its root. Moreover, trees connecting to different roots must be disjoint.
(2) Consider a tree T and one of its leaves $v \in T$. If v has a high anti-degree, it has few available colors among the ones available in the almost-clique (Lemma 4.19). Similarly, it is possible that all external neighbors of v block the k colors it has available. We call such nodes *spoiled* and must ensure that they only represent a small fraction of every tree.
(3) Assuming all trees have $\Theta(\Delta/k)$ *unspoiled* leaves, the leaves used to recolor augmenting paths in each tree have to use different colors. We say that we *harvest* the trees. When k is $\Omega(\log n)$, if each leaf try one color, w.h.p., the number of conflicts between trees is small, so the issue is merely to detect them. When k is $O(\log n)$, as leaves try $\Theta(\log n)$ colors (to ensure to be successful w.h.p.), many conflicts may arise.

Growing balanced augmenting trees. To overcome Technical Issue 1, when growing the trees, we ensure *they all grow at the same speed*. More precisely, the GrowTree algorithm (Algorithm 23) takes as input a forest F and finds *exactly* β children for each leaf in F for $\lfloor \log_\beta(\Delta/(\alpha k)) \rfloor$ rounds so that each tree has $\Omega(\Delta/\beta k)$ leaves. Nodes then sample a precise number of colors to ensure that w.h.p. the majority of them finds enough leaves to grow trees to $\Theta(\Delta/k)$ leaves (Lemma 7.31).

Bounding the number of spoiled nodes. When a leaf v attempts to recolor its path to some uncolored node, it must sample colors in $L_\varphi(K) \cap L_\varphi(v)$. This might be a problem for two reasons. First, if v is unpromising (Lemma 4.19). The second possibility is it that its k colors in $L_\varphi(K) \cap L_\varphi(v)$ are blocked by external neighbors. The latter eventuality demands more caution. In particular, if we allow adversarial behavior on the outside, it might be that external neighbors block the one remaining color in $L_\varphi(K)$ for all nodes. The random colors for the ℓ-th iteration of AugmentingPath

are made of two independent random lists $\mathbf{L}^g_{3,\ell}$ and $\mathbf{L}^h_{3,\ell}$, where the former if used in the growing steps and the former in the harvesting phase. During the growing phase, nodes have $\Omega(\Delta)$ colors available even with adversarial recolorings outside. So we call vertices spoiled if they would be unlikely to find an available color in the clique palette given the color sampled outside K, i.e., as leaves they would be unlikely to find an available color.

Definition 7.27 (Spoiled Node)**.** We say a node $v \in K$ is *spoiled* during the ℓ-iteration of AugmentingPath if

$$|L_\varphi(K) \cap L_\varphi(v) \setminus \mathbf{L}^h_{3,\ell}\left(\bigcup_{K' \neq K} N(v) \cap K'\right)| \leqslant k/2 \ .$$

We deal with Technical Issue 2 in two ways. We previously computed a colorful matching of size $\Theta(a(K))$ for a large enough constant to reduce the number of unpromising nodes to a sufficiently small fraction of K (Corollary 7.14). Second, we show that it is very unlikely that nodes outside of K block more than $k/2$ colors for a large fraction of K. It stems from the two following observations

- external neighbors in other almost-cliques try $\Theta(\log n)$ *uniform colors* in $[\Delta + 1]$ at each iteration, and
- the preconditioning of almost-cliques reduced the external degree to $e_{max} = O(\Delta/\log n)$. (Theorem 7.11)

So, with high probability, $\Omega(\Delta)$ nodes in K have $\Omega(k)$ available colors in $L_\varphi(K)$. More precisely, in Lemma 7.28, we show that with high probability *over the randomness outside of* K, at most $3\Delta/\alpha$ nodes are spoiled in K. Then, Lemma 7.31 show that with high probability *over the randomness inside* K, all trees have $\Theta(\Delta/k)$ unspoiled leaves.

Harvesting trees. While Technical Issue 2 was about the conflict with external neighbors, Technical Issue 3 is about the conflicts inside the almost-clique. Say leaves sample one color. For a fixed tree T_u and one of its unspoiled leaves v, the expected number of colors blocked in $L_\varphi(K) \cap L_\varphi(v)$ by sampling in other trees is $(k-1) \cdot \Theta(\Delta/\alpha k) \cdot k/\Delta = \Theta(k/\alpha)$. When $k = \Omega(\log n)$, we get concentration and show that w.h.p. a constant fraction of the leaves still have $\Theta(k)$ colors available, even after revealing the

randomness in other trees. Therefore, as long as $k = \Omega(\log n)$, HARVEST colors a constant fraction of the uncolored nodes with high probability (Lemma 7.33).

When $k = O(\log n)$, leaves sampling $\Theta(\log n)$ colors ensure w.h.p. that every leaf has $\Theta(\log n)$ colors to choose from. The drawback to such intensive sampling is that we must resolve conflicts between trees. Using the high expansion property of the sparsified graph, it is possible to deliver to every node in the almost-clique the list of $\Theta(\log n)$ available colors to each of the k uncolored nodes. Conflicts are then resolved locally (by every node).

7.5.2. Growing the Trees.

ALGORITHM 23. GROWTREE (for the ℓ-th iteration of AUGMENTING-PATH).

Input: An almost-clique K with $|K \setminus \operatorname{dom}\varphi| = k < \Delta/(\alpha\beta)$ uncolored nodes, a colorful matching M of size at most $2\alpha \cdot a(K)$ and at most Δ/α unpromising nodes.

Define $d = \left\lfloor \log_\beta \frac{\Delta}{\alpha k} \right\rfloor$ and $U_0 = K \setminus \operatorname{dom}\varphi$.
For $i = 0$ to d,

(**G**1) If $i < d$, let $x = 5\beta$.
Else if $i = d$, nodes compute the exact size of $|U_d|$ in $O(d)$ rounds and set $x = \left\lfloor \frac{6\Delta}{\alpha|U_d|} \right\rfloor$.
Each $u \in U_i$ picks a set S_u of x fresh random colors from $\mathbf{L}_{3,\ell}(u)$.

(**G**2) For each active node $u \in U_i$, let $S'_u \subseteq S_u$ be the colors χ that are
i) unused by colored external neighbors of u and $\chi \notin \mathbf{L}_{3,\ell}(\bigcup_{K' \neq C} K' \cap N(u))$;
ii) unused by nodes of $B_i = U_{\leqslant i} \cup M$;
iii) uniquely sampled: $\chi \notin \bigcup_{v \in U_i \setminus \{u\}} S_v$; and
iv) used by a colored node in $N(u) \cap K$

(**G**3) For each $u \in U_i$, choose y_u arbitrary colors $\chi_1, \ldots, \chi_{y_u} \in S'_u$ where

$$y_u = \begin{cases} \beta & \text{if } i < d \\ |S'_u| & \text{if } i = d \end{cases}.$$

For all $j \in [y_u]$, define $v_{u,j}$ as the node of $(N(u) \cap K) \setminus B_i$ with color χ_j. Let $U_{i+1} = \bigcup_{u \in U} \{v_{u,1}, \ldots, v_{u,y_u}\}$ and $\pi(v_{u,j}) = u$ for all $j \in [y_u]$ and $u \in U_i$.

Bounding Conflicts with the Outside. In the next lemma, we bound the number of spoiled nodes in K (Definition 7.27). Note that the probability is taken only over the randomness of nodes *outside of* K.

Lemma 7.28. *Consider an almost-clique K. With high probability over $\mathbf{L}_{3,\ell}(V \setminus K)$, almost-clique K contains at most $3\Delta/\alpha$ spoiled nodes during the ℓ-th call to* GrowTree *in* AugmentingPath.

Proof. By Corollary 7.14, the almost-clique contains at least $|K| - \Delta/\alpha \geqslant (1 - \varepsilon - 1/\alpha)\Delta \geqslant (1 - 2/\alpha)\Delta$ promising nodes, and each such node v has at least k colors in $L_\varphi(K) \cap L_\varphi(v)$. Let us focus on a set S of exactly $(1 - 2/\alpha)\Delta$ promising nodes. For each selected promising node $v \in S$, we focus on exactly k colors $L'(v) \subseteq L_\varphi(K) \cap L_\varphi(v)$. We consider the $k|S|$ pairs (χ, v) where $v \in S$ is selected promising node and χ is a color $\chi \in L'(v)$ from its selected colors.

Let us denote by $N_{ext}(v) \stackrel{\text{def}}{=} N(v) \cap \bigcup_{K' \neq K} K'$, the external neighbors of v that might get (re)colored. Let $B = \bigcup_{v \in S} N_{ext}(v)$ the set of vertices that are external neighbors of at least one node in S. Recall that, by Theorem 7.11, for each $v \in K$, $|N_{ext}(v)| \leqslant e_{max}$. For each $u \in B$ and color $\chi \in [\Delta + 1]$, let $\mathbf{X}_{u,\chi}$ be defined as:

$$\mathbf{X}_{u,\chi} = \frac{1}{e_{max}} |\{v \in S \cap N(u) \text{ such that } \chi \in L'(v)\}| \cdot \mathbb{I}[\chi \in \mathbf{L}^h_{3,\ell}(u)] \ ,$$

counting how many times a $u \in B$ is in conflicting with the selected nodes S over a selected color χ, re-scaled by $1/e_{max}$ so $\mathbf{X}_{u,\chi}$ is distributed in $[0, 1]$.

Let $\mathbf{X} = \sum_{u \in B} \sum_{\chi \in [\Delta+1]} \mathbf{X}_{u,\chi}$. Each edge between $v \in S$ and $u \in B$ contributes $1/e_{max}$ to $\mathbf{X}$ for each color $\chi \in L'(v)$ such that $\chi \in \mathbf{L}^h_{3,\ell}(u)$. There are at most $|S|e_{max}$ such edges, and each color $\chi \in L'(v)$ is sampled in $\mathbf{L}^h_{3,\ell}(u)$ with probability at most $20\beta/\Delta$. By linearity of expectation, $\mathbb{E}[\mathbf{X}] \leqslant |S| \cdot k \cdot 20\beta/\Delta$. Note e_{max} cancelling itself.

Random variables $\mathbf{X}_{u,\chi}$ are 1-negatively correlated, since each vertex samples β colors in $\mathbf{L}^h_{3,i}$ with repetition and independently of other vertices. As the Chernoff Bound also applies to such variables (see [Doe20, Theorem 1.10.23]), we get

$$\mathbb{P}[\mathbf{X} \geqslant 2\,\mathbb{E}[\mathbf{X}]] \leqslant \exp(-\,\mathbb{E}[\mathbf{X}]/3) = \exp(-k\beta) \leqslant 1/\mathrm{poly}(n)\ ,$$

where we use that $|S| \geqslant \Delta/2$. Therefore, there are at most $e_{max} \cdot \frac{40\beta k|S|}{\Delta} \leqslant k|S|/(4\alpha)$ conflicts between the selected colors of nodes in S and the colors sampled by their external neighbors (by Eq (7.4)). Therefore, at most $\frac{k|S|/(4\alpha)}{k/2} \leqslant \Delta/\alpha$ node are left with fewer than $k/2$ colors in $L_\varphi(v) \cap L_\varphi(K)$. ∎

Growing the Forest. Henceforth, we fix the random lists $\mathbf{L}_{3,\ell}(V \setminus C)$ such that K contains at most $3\Delta/\alpha$ spoiled nodes, which holds w.h.p. by Lemma 7.28. Let $U = \bigcup_i U_i$ and π be the values produced by Algorithm 23, the graph $F = (U, E(F))$ with $E(F) = \{u\pi(u) : u \in U_{>0}\}$ is a forest. Suppose that at each Step (**G3**), each u satisfies $|S_u| \geqslant y_u$. Then F is a forest of k trees of arity β, depth $d+1$ and at most $6\Delta/(\alpha k)$ leaves each. We reveal the randomness inside K as we grow the tree, conditioning at each growing step on arbitrary randomness from nodes that are already in the forest. The following lemma shows that it is unlikely that nodes sample bad colors.

Lemma 7.29. *Let $0 \leqslant i < d$. Then, for any lists in $\mathbf{L}^g_{3,\ell}(U_{\leqslant i})$, if a node u samples a fresh color $\chi \in [\Delta+1]$, we have*

$$\mathbb{P}_{\chi\in[\Delta+1]}[\chi \textit{ violates a condition in Step (}\mathbf{G2}\textit{)} \mid U_{\leqslant i}] \leqslant 14/\alpha\ .$$

Proof. For a fixed $i < d$, we have $|U_i| \leqslant k\beta^i$. We bound the number of colors χ that might be conflicting with for each item in Step (**G2**):

i) Node u has at most $\varepsilon\Delta$ colored external neighbors. The number of colors in $\mathbf{L}_{3,\ell}(\bigcup_{K'\neq K} K' \cap N(u))$ is at most $80\beta e_{max} < \Delta/\alpha$ by Claim 7.2 and Eq (7.4).

ii) For $i < d$, the number of nodes in $U_{\leqslant i}$ is at most $\sum_{j=0}^{d-1} |U_j| \leqslant k\beta^d \leqslant \Delta/\alpha$. Along with the colorful matching, at most $\Delta/\alpha +$

$2\alpha a(K) \leqslant (1/\alpha + 2\varepsilon\alpha)\Delta \leqslant 3\Delta/\alpha$ nodes are colored in B_i (by Eq (7.2)).

iii) The number of colors sampled (and thus blocked) by active nodes is $x|U_i| = xk\beta^i \leqslant 6k\beta^d \leqslant 6\Delta/\alpha$ where the first inequality comes from $i < d$.

iv) The number of uncolored nodes in K is at most $\Delta/\alpha\beta$, and at most $3\varepsilon\Delta$ colors are repeated in K (Remark 7.24); hence, the number of non-repeated colors used in $N(u) \cap K$ is be at least $(1-\varepsilon)\Delta - \Delta/\alpha\beta - 3\varepsilon\Delta \geqslant (1 - 4/\alpha)\Delta$ by Eq (7.2).

Summing all failure probabilities with an union bound, we get the claimed bound. ∎

Since nodes sample $\Omega(\log n)$ colors when $i < d$, Lemma 7.29 implies the following corollary.

Corollary 7.30. *With high probability over $\mathbf{L}^g_{3,\ell}(U_{<d})$, for each $i < d$ and $u \in U_i$, $|S'_u| \geqslant \beta \overset{def}{=} y_u$.*

Because of rounding in d, trees might not contain enough leaves after d growing steps. Furthermore, we also need to show that most leaves are unspoiled.

Lemma 7.31. *Let U_d be the set given by Algorithm 23. With high probability over $\mathbf{L}^g_{3,\ell}(U_d)$, the number of unspoiled nodes in U_{d+1} is at least Δ/α.*

Proof. For each node $u \in U_d$, define a random variable $\mathbf{X}_{u,i}$ for each of its sampled color $i \in [x]$. Let $\mathbf{X}_{u,i}$ be one if and only if the i-th color it samples 1) has no conflict in Step (**G**2) and 2) the corresponding $v_{u,\chi}$ is unspoiled. The analysis is similar to Lemma 7.29 but has to be a bit more careful. Namely, failures caused by i), ii) and iv) remain unchanged but the number of colors sampled by active nodes is different. Furthermore, we now have to filter out spoiled nodes.

- Spoiled nodes are easily dealt with by Lemma 7.28. Indeed, there are at most $3\Delta/\alpha$ spoiled nodes in K. Since each such node blocks one color, they only amount to a small fraction.
- Since nodes try $x = \lfloor 6\Delta/(\alpha|U_d|)\rfloor$ colors, the total number of colors sampled by active nodes is $x|U_d| \leqslant 6\Delta/\alpha$.

If we union bound over all possible failures for a random color $\chi \in [\Delta+1]$, we get $\mathbb{P}[\mathbf{X}_{u,\chi} = 1 \,|\, U_d \setminus \{u\}] \leqslant 15/\alpha \leqslant 1/25$. By Markov inequality, a node $u \in U_d$ samples more than $x/5$ bad colors w.p. at most $\frac{x}{25} \cdot \frac{5}{x} = 1/5$. Giving priority to nodes of lowest ID, the martingale inequality (Lemma 2.3) shows that, w.e.h.p. in $|U_d|$, at most $|U_d|/4$ nodes sample more than $x/5$ bad colors. Note that, by definition of d,

$$|U_d| = k\beta^d \geqslant k\beta^{\log_\beta(\Delta/\alpha k)-1} \geqslant \frac{\Delta}{\alpha\beta} .$$

Therefore, $1 - e^{-\Omega(|U_d|)} \geqslant 1 - e^{-\Omega(\Delta/\beta)} \geqslant 1 - \text{poly}(n)$ (because $\Delta \in \Omega(\log^4 n)$) and the previous claim holds with high probability. That means that w.h.p. the number of unspoiled nodes in U_{d+1} is at least

$$\begin{aligned} \text{(because } x \stackrel{\text{def}}{=} \left\lfloor \tfrac{6\Delta}{\alpha|U_d|} \right\rfloor\text{)} \qquad & \frac{|U_d|}{4} \cdot \frac{4x}{5} \geqslant \frac{|U_d|}{5}\left(\frac{6\Delta}{\alpha|U_d|} - 1\right) \\ \text{(because } |U_d| \leqslant \Delta/\alpha\text{)} \qquad & = \frac{6\Delta/\alpha - |U_d|}{5} \geqslant \frac{\Delta}{\alpha} , \end{aligned}$$

which concludes the proof of the Lemma. ∎

7.5.3. Harvesting the Trees. In the previous section, we argued that there were Δ/α unspoiled leaves. In this section, we argue that enough of these leaves find good colors to color a constant fraction of uncolored nodes. While in Lemma 7.31, we bound from below the total number of unspoiled nodes, because all trees have roughly the same size (at most $6\Delta/\alpha k$ each), a simple counting argument gives the following claim.

Claim 7.32. *There are at least* $0.9k$ *trees with* $\Delta/(2\alpha k)$ *unspoiled leaves.*

Henceforth, we will be focusing on those trees with many unspoiled leaves. Note that at Step (**G**2) of Algorithm 23, we ensure that if a leaf finds a color, each node on its path to the root can change its color without creating conflicts. Hence, this section focuses on counting successful leaves in each tree.

When k is large. Assume first that $k \geqslant \Omega(\log n)$.

ALGORITHM 24. HARVEST (for k greater than $\Omega(\log n)$).
Input: the forest F of augmenting trees computed by Algorithm 23.

(**H**1) Leaves $v \in F$ try *one* fresh color $\boldsymbol{\chi}(v) \in \mathbf{L}_3^h(v)$. We call v *successful* if it can adopt $\boldsymbol{\chi}(v)$, and thereby recolor the path in F connecting v to its uncolored root. Leaves can learn if they are successful in $O(d)$ rounds.

(**H**2) Each leaf can learn in $O(1)$ rounds if a leaf from another tree sampled the same color (by Claim 7.7). We call a tree *successful* if it has at least one successful leaf that is not conflicting with leaves from other trees.

(**H**3) If a tree is successful, it recolors the path from its root to its successful leaf in $O(d)$ rounds.

Lemma 7.33. *Let K be an almost-clique with $\beta \leqslant k \leqslant \Delta/(\alpha\beta)$ uncolored nodes. In Step (**H**3), with high probability over $\mathbf{L}_{3,\ell}^h(K)$, we color at least $\Omega(k/\alpha)$ uncolored nodes.*

PROOF. Let $k' = 0.9k$ and $T_1, \dots, T_{k'}$ be k' trees with at least $\Delta/(2\alpha k)$ paths to unspoiled leaves. For each such tree, let us only consider exactly $\Delta/(2\alpha k)$ selected unspoiled leaves. For each selected leaf v, let us focus on a subset of size $k/2$ of its palette which is available and not conflicting with colors sampled by external neighbors. We call them its selected colors.

For each $i \in [k']$, let $\mathbf{X}_i$ be the indicator random variable for the event that (1) tree i contains exactly one selected leaf that samples a selected color, and (2) no other node in the almost-clique tried the same color as this selected leaf. Note that $\mathbf{X}_i$ may be expressed as the difference between two random variables $\mathbf{Y}_i$ and $\mathbf{Z}_i$, where $\mathbf{Y}_i$ corresponds to the event that at least one selected leaf of T_i tries one of its selected colors, and $\mathbf{Z}_i$ corresponds to at least two selected leaves trying a selected color or a selected leaf trying a selected color but failing to keep it. We have:

$$\mathbb{P}[\mathbf{X}_i = 1] \geqslant \frac{\Delta}{2\alpha k} \cdot \frac{k}{2\Delta} \cdot \left(1 - \frac{k/2+1}{\Delta}\right)^{\Delta/(2\alpha k)-1} \cdot \left(1 - \frac{1}{\Delta}\right)^{6\Delta/\alpha}.$$

The first term $(\Delta/(2\alpha k))$ corresponds to doing a sum over all selected nodes in T_i of their probability of being the selected node that succeeds.

The second term $(k/(2\Delta))$ corresponds to the probability that each selected node samples one of its selected colors. Call this color χ. The third term corresponds to all other selected leaves of T_i sampling neither one of their selected colors nor χ. The last term corresponds to all leaves not trying χ. Using $1 - x/2 \geqslant e^{-x}$ for $x \in [0,1]$, we get $\mathbb{P}[\mathbf{X}_i = 1] \geqslant e^{-12/\alpha}/(4\alpha)$. Let $\mathbf{X} = \sum_{i=1}^{k'100} \mathbf{X}_i$, by linearity $\mathbb{E}[\mathbf{X}] \geqslant k' \cdot e^{-12/\alpha}/(4\alpha)$.

Consider similarly the random variables $\mathbf{Y}_i$ and their sum $\mathbf{Y}$. We have

$$\mathbb{P}[\mathbf{Y}_i = 1] = 1 - \left(1 - \frac{k}{2\Delta}\right)^{\Delta/(2\alpha k)} \geqslant \frac{1}{4\alpha + 1} \ .$$

(using $(1-x)^y \leqslant 1/(1+xy)$ for $x \in [0,1]$ and $y > 0$)

Therefore, $\mathbb{E}[\mathbf{X}]$ and $\mathbb{E}[\mathbf{Y}]$ are both of order $\Theta(k/\alpha)$. Additionally, $\mathbf{Y}$ is 1-Lipschitz and 1-certifiable, and $\mathbf{Z}$ is 2-Lipschitz and 2-certifiable. Set $t = k'/(4\alpha+1)$ and n large enough to have $t/16 \geqslant 30\sqrt{k} \geqslant 30\sqrt{\mathbb{E}[\mathbf{Y}]}$, by Lemma 2.7,

$$\begin{aligned}\mathbb{P}\Big[|\mathbf{Y} - \mathbb{E}[\mathbf{Y}]| \geqslant t/8\Big] &\leqslant 4\exp\left(-\frac{\Big(t/8 - 30\sqrt{\mathbb{E}[\mathbf{Y}]}\Big)^2}{8\,\mathbb{E}[\mathbf{Y}]}\right) \\ &\leqslant 4\exp\big(-\Omega(t^2/k)\big) \leqslant 1/\mathrm{poly}(n) \ ,\end{aligned}$$

since $k \geqslant \beta \gg \alpha^{-1}\log n$. By the same reasoning $|\mathbf{Z} - \mathbb{E}[\mathbf{Z}]| \leqslant t/8$ with high probability. Since $\mathbf{X} = \mathbf{Y} - \mathbf{Z}$ we have that $|\mathbf{X} - \mathbb{E}[\mathbf{X}]| \leqslant |\mathbf{Y} - \mathbb{E}[\mathbf{Y}]| + |\mathbf{Z} - \mathbb{E}[\mathbf{Z}]| \leqslant t/4$ and thus $\mathbf{X} \geqslant \mathbb{E}[\mathbf{X}] - t/4 \geqslant \Omega(k/\alpha)$ with high probability. Since $\mathbf{X}$ is a lower bound on the number of trees that successfully recolor their root, at least $\Omega(k/\alpha)$ uncolored nodes get colored, with high probability. ∎

Coloring the last nodes. Assume now that only $k = O(\log n)$ uncolored nodes remain.

ALGORITHM 25. HARVEST (for k smaller than $O(\log n)$).
Input: the forest F of augmenting trees computed by Algorithm 23.

(**L**1) Leaves try β fresh colors. A leaf keep its color if it not used by any colored neighbor nor tried by external neighbors.

(**L2**) Via simple aggregation on the augmenting tree, each uncolored node v learns a list S_v of up to $\Theta(k)$ *candidate* colors that its leaves could use to recolor themselves.

(**L3**) After some routing, the whole almost-clique knows about all of $\{S_v, v \in K \setminus \operatorname{dom}\varphi\}$. All nodes then locally compute the same matching of size $\Omega(k)$ in the bipartite graph with vertices $(K \setminus \operatorname{dom}\varphi) \cup [\Delta+1]$ and edges (v, χ) iff $\chi \in S_v$.

Sending the sets $\{S_v, v \in K \setminus \operatorname{dom}\varphi\}$ in Step (**L3**) is done by RandomPush (Algorithm 18).

Lemma 7.34. *Let K be an almost-clique with $k \leqslant \beta$ uncolored nodes. At the end of Step (**L3**), with high probability over $\mathbf{L}^h_{3,\ell}(K)$, we color at least $\Omega(k/\alpha)$ uncolored nodes.*

Proof. There exist $k' = 0.9k$ uncolored nodes with at least $\Delta/(2\alpha k)$ paths to unspoiled leaves. Let us now argue that an uncolored node v with this many paths in its tree has them find at least $k/4$ distinct colors, w.h.p. Let us associate to the i-th such leaf u_i a random variable $\mathbf{X}_i$ such that:

- If previous leaves discovered $k/4$ distinct colors already, then $\mathbf{X}_i = 1$ w.p. 1,
- If previous leaves discovered fewer than $k/4$ distinct colors, then $\mathbf{X}_i = 1$ iff leaf number i discovers a new color.

Note that $\mathbf{X}_i$ is a function of the random trials $\boldsymbol{\chi}(u_1), \dots, \boldsymbol{\chi}(u_i)$.

When trials $\boldsymbol{\chi}(u_1), \dots, \boldsymbol{\chi}(u_{i-1})$ from previous leaves have discovered fewer than $k/4$ distinct colors, since the i-th leaf is unspoiled, it still has at least $k/4$ colors it can discover. The probability that it finds one of them is at least:

$$\mathbb{E}[\mathbf{X}_i \mid \boldsymbol{\chi}(u_1), \dots, \boldsymbol{\chi}(u_i)] = 1 - \left(1 - \frac{k}{4\Delta}\right)^{\beta} \geqslant 1 - \frac{1}{1 + k\beta/(4\Delta)} = \frac{k\beta}{4\Delta + k\beta} \geqslant \frac{k\beta}{5\Delta}$$

where the first inequality uses that $(1-x)^y \leqslant \frac{1}{1+xy}$ for $x \in [0,1]$ and $y > 0$, and the last inequality comes from $\Delta \geqslant \beta^2 \geqslant k\beta$.

Therefore, $\mathbb{E}[\sum_i \mathbf{X}_i] \geqslant \frac{\Delta}{2\alpha k} \cdot \frac{k\beta}{5\Delta} = \beta/(10\alpha)$. The series of $\mathbf{X}_i$ satisfy the conditions of Lemma 2.3, hence it has value at least $\beta/(20\alpha)$, w.e.h.p. in β/α, so w.h.p. in n. By definition of $\{\mathbf{X}_i\}$, if $\beta/(20\alpha) \geqslant k/4$, this gives that at least $k/4$ colors are found in the tree, while if $\beta/(20\alpha) \leqslant k/4$, we only get that at least $\beta/(20\alpha) \geqslant k/(20\alpha)$ colors are found in the tree (as $\beta \geqslant k$). ∎

7.5.4. Proof of Theorem 7.16. Consider almost-clique K. The fact that a constant fraction of the uncolored vertices of K get colored follows by applying successively Lemmas 7.28, 7.31, 7.33 and 7.34 and Corollary 7.30.

We now argue that our algorithm has the claimed runtime.

Algorithm 23 runs in $O(\log\Delta + d) = O(\log\Delta)$ rounds. Beforehand, vertices broadcast lists $\mathbf{L}^g_{3,\ell}(v)$ and $\mathbf{L}^h_{3,\ell}(v)$. With high probability, they each contain at most $O(\beta) = O(\log n)$ colors that can be described in $O(\log\Delta)$ bits each. Steps (**G**1) to (**G**3) for $i < d$ takes $O(1)$ rounds to implement. When $i = d$, computing $|U_d|$ takes $O(\log\Delta)$ rounds and Steps (**G**2) and (**G**3) take $O(1)$ rounds.

We conclude with implementation details for Algorithm 25. In each tree, the root learns a subset of the colors its leaves can pick in $O(\log\Delta)$ rounds as follows: in each round, each node that knows about an available color that it has not yet sent towards the root sends as many such colors that it can towards the root (with a maximum of $\log\Delta/\log n$ colors per round). In $O(\log\Delta + k\cdot\log\Delta/\log n)$ rounds, the root learns about a set S_v of $\Omega(k)$ colors, if that many are available in the tree.

Each root v crafts a message of the form $(\mathsf{ID}(v), \chi)$ for each of the colors $\chi \in S_v$, and selects a subset of k of them if it has more than k. The almost-clique then runs RandomPush with $O(k^2) = O(\beta^2)$ messages for $O(\log\Delta)$ rounds with the selected messages. Note that the bipartite graph with vertices $(K \setminus \operatorname{dom}\varphi) \cup [\Delta+1]$ and edges (v,χ) such that $\chi \in S_v$ is now known to all nodes in K. Moreover, this graph has $\Omega(k^2)$ edges and maximum degree k. Therefore, it has a matching of size $\Omega(k)$ (which can be computed locally by a simple deterministic greedy algorithm). It follows

that all nodes compute the same matching without extra communication and recolor the path corresponding to each edge in the matching in $O(d)$ rounds. Therefore, at least $\Omega(k)$ nodes get colored. ∎

CHAPTER 8

Lower Bound

As discussed in Section 6.2.3, at its core, our lower bound result is based on proving a lower bound for distributed computing a perfect matching in a random bipartite graph. More concretely, let B be a bipartite graph on nodes $V \times C$, where V models n nodes and C models n colors. B contain each of the n^2 possible edges between V and C with probability $\operatorname{poly}\log n/n$. In the following, we show that computing a perfect matching of B by a distributed message passing algorithm on B requires $\Omega(\log n/\log\log n)$ rounds, even in the LOCAL model (i.e., even if the nodes in B can exchange arbitrarily large messages with their neighbors in B). We start with a simple observation regarding the structure of perfect matchings in bipartite graphs.

Lemma 8.1. *Let $H = (V_E, E_H)$ be a bipartite graph, let $v_0 \in V_H$ be a node of H, and for every integer $d \geqslant 0$, define $V_d \subset V_H$ be the set of nodes at distance exactly d from v_0 in H. Then, if H has a perfect matching, for every perfect matching M of H and for every $d \geqslant 0$, the number of edges of M between nodes in V_d and nodes in V_{d+1} is equal to*

$$S_d \stackrel{def}{=} \sum_{i=0}^{i} (-1)^i \cdot |V_d|.$$

Proof. We prove the statement by induction on d. For $d = 0$, we have $V_0 = \{v_0\}$ and thus clearly the number of matching edges between V_0 and V_1 must be $S_0 = |V_0| = 1$. Let us, therefore, consider $d > 0$ and assume that the statement holds for all $d' < d$. First note that for all $d > 0$, we have $S_d = |V_d| - S_{d-1}$. Note that all neighbors of nodes in V_d are either in V_{d-1} or in V_{d+1}. Because every node in V_d must be matched, the number of matching edges between V_d and V_{d+1} must be equal to $|V_d|$ minus the number of matching edges between V_{d-1} and V_d. By the induction

hypothesis, the number of matching edges between V_{d-1} and V_d is equal to S_{d-1}. The number of matching edges between V_d and V_{d+1} is therefore equal to $S_d = |V_d| - S_{d-1}$ as claimed. ∎

Note that Lemma 8.1 essentially states that the bipartite perfect matching problem is always a global problem in the following sense. In order to know the number of matching edges in a perfect matching between two sets V_d and V_{d+1}, one must know the sizes of all the sets $V_0, \dots, V_d$. As sketched in Section 6.2.3, we can use this observation to prove an $\Omega(\log n / \log\log n)$-round lower bound for computing a perfect matching in the random bipartite graph B. The formal details are given by the following theorem.

THEOREM 8.2. *Let $B = (V \cup C, E_B)$ be a random bipartite $2n$-node graph with $|V| = |C| = n$ that is defined as follows. For every $(v, c) \in V \times C$, edge $\{v, c\}$ is in E_B independently with probability p, where $p \leqslant \operatorname{poly}\log(n)/n$ and $p \geqslant \alpha \ln(n)/n$ for a sufficiently large constant $\alpha > 0$. Any distributed (randomized)* LOCAL *algorithm that succeeds in computing a perfect matching of B with probability at least $2/3$ requires at least $\Omega(\log n / \log\log n)$ rounds.*

PROOF. First note that if $p \geqslant \alpha \ln(n)/n$ and the constant α is chosen sufficiently large, then B has a perfect matching w.h.p. This is well-known [Bol02, Section VII.3] and can be seen by verifying Hall's condition.

Let $T \stackrel{\text{def}}{=} \eta \cdot \ln n / \ln\ln n$ for a sufficiently small constant $\eta > 0$ that will be determined later and assume that there exists a T-round randomized distributed perfect matching algorithm for the random graph B. We assume that after T rounds, every node outputs its matching edge such that with probability $\geqslant 2/3$, the outputs of all nodes are consistent, i.e., the algorithm computes a perfect matching of B. Consider some node $v_0 \in V \cup C$ and for every integer $d \geqslant 0$, let V_d be the set of nodes at distance exactly d from v_0. We next fix two parameters $\ell \stackrel{\text{def}}{=} T$ and $h \stackrel{\text{def}}{=} \ell + T + 2$. To prove the lower bound, we concentrate on the nodes in V_h and the computation of their matching edges. In a T-round algorithm, a node v can only receive information from nodes within T hops, and therefore the output of a node v must be a function of the combination of the initial states of the nodes of the T-hop neighborhood of v (when assuming that all the private randomness

used by a node v is contained in its initial state). Assume that the initial state of a node contains its ID, as well as the IDs of its neighbors. Then, v's output of a T-round algorithm is a function of the subgraph induced by the $(T+1)$-hop neighborhood of v. The outputs of the nodes in V_h therefore only depend on nodes in V_d for $d \in \{\ell+1, \dots, h+T+1\}$ and on edges between those nodes. And it in particular means that nodes in V_h do have to decide about their matching edges without knowing anything about nodes in V_ℓ.

In the following, we assume that nodes in V_h can collectively decide about their matching edges. We further assume that to do this, the nodes in V_h have the complete knowledge of the subgraph of B induced by $(V_0 \cup \dots \cup V_{\ell-1}) \cup (V_{\ell+1} \cup \dots \cup V_{h+T+1})$. That is the nodes in V_h have the complete knowledge of the graph induced by the nodes that are within distance $h+T+1 = \ell+2T+3$ of v_0, with the exception of the nodes in V_ℓ and all their edges. Because we want to prove a lower bound, assuming coordination between the nodes in V_h and assuming knowledge of parts of the graph that are not seen by nodes in V_h can only make our result stronger. Note that by Lemma 8.1, the number of matching edges between level V_h and V_{h+1} is equal to $S_h = \sum_{i=0}^{h} (-1)^i |V_{h-i}|$, which is an alternating sum that contains the term $|V_\ell|$ (either positively or negatively, depending on the parity of $h-\ell$). Hence, given the knowledge of the subgraph induced by $(V_0 \cup \dots \cup V_{\ell-1}) \cup (V_{\ell+1} \cup \dots \cup V_{h+T+1})$, the number of matching edges between nodes in V_h and nodes in V_{h+1} is in a one-to-one relation with the number of nodes in V_ℓ. Without knowing $|V_\ell|$ exactly, the nodes in V_h can therefore not compute their matching edges. Therefore, in order to prove the lemma, we need to prove that from knowing the subgraph induced by $(V_0 \cup \dots \cup V_{\ell-1}) \cup (V_{\ell+1} \cup \dots \cup V_{h+T+1})$, the size of V_ℓ can at best be estimated exactly with probability $2/3$.

For this, we define several random variables. Let $X_\ell = |V_\ell|$ be the number of nodes in V_ℓ, i.e., X_ℓ is the random variable that the nodes in V_h need to estimate exactly in order to compute their matching edges. If we define $\mathscr{K}$ to be a random variable that describes the knowledge that is provided to the nodes in V_h to determine X_ℓ, then we intend to estimate

$$p_{\text{est},K} \stackrel{\text{def}}{=} \max_{x_\ell > 0} \mathbb{P}[X_\ell = x_\ell \mid \mathscr{K} = K] \ .$$

Clearly, if K is the actual state of the subgraph induced by $(V_0 \cup \cdots \cup V_{\ell-1}) \cup (V_{\ell+1} \cup \cdots \cup V_{h+T+1})$, the nodes in V_h can determine the exact value of X_ℓ with probability at most $p_{\text{est},K}$. To prove the theorem, we will show that with high probability, $\mathscr{K}$ takes on a "good" state K for which $p_{\text{est},K} \leqslant 1/2 + o(1)$. For estimating X_ℓ correctly, we then either need to have a "bad" K, which happens with probability $o(1)$ or we need to have a "good" K and estimate X_ℓ correctly, which happens with probability $1/2 + o(1)$. Overall, the probability for estimating X_ℓ correctly is then at best $1/2 + o(1) \leqslant 2/3$ for sufficiently large n. In order to estimate the probability of $X_\ell = x$, given the knowledge of the nodes in V_h, we first look at the conditioning on $\mathscr{K} = K$ more closely. First note that by symmetry, the probability $\mathbb{P}[X_\ell = x \mid \mathscr{K} = K]$ only depends on the topology of the subgraph induced by $(V_0 \cup \cdots \cup V_{\ell-1}) \cup (V_{\ell+1} \cup \cdots \cup V_{h+T+1})$ and not on the set of node IDs that appear in the part of the graph known by V_h. Further, the probability also does not depend on the edges of the induced subgraph known by V_h. The size X_ℓ of V_ℓ only depends on the additional edges of the nodes in $V_{\ell-1}$ and $V_{\ell+1}$. The probability $\mathbb{P}[X_\ell = x \mid \mathscr{K} = K]$ therefore only depends on the sizes of the sets $V_0, \ldots, V_{\ell-1}$ and $V_{\ell+1}, \ldots, V_{h+T+1}$.

We first introduce the necessary random variables and some notation to simplify our calculation. For each $d \in \{0, \ldots, h + T + 1\}$, we define a random variable $X_d = |V_d|$. For convenience, for every d, we define $V_{\geqslant d} = V_d \cup V_{d+1} \cup \ldots$ to be the set of nodes at distance at least d from v_0. Throughout the calculations, we will concentrate on some fixed knowledge of the nodes in V_h. We therefore consider some values $x_0, x_1, \ldots, x_{h+T+1}$ and for each d, we define $\mathcal{X}_d$ as a shortcut for the event $\{X_d = x_d\}$ that the random variable X_d takes the value x_d. For convenience, we also define $\mathcal{X}_{<d} \stackrel{\text{def}}{=} \mathcal{X}_0 \cap \cdots \cap \mathcal{X}_{d-1}$, $\mathcal{X}_{\leqslant d} = \mathcal{X}_{<d} \cap \mathcal{X}_d$, $\mathcal{X}_{>d} \stackrel{\text{def}}{=} \mathcal{X}_{d+1} \cap \cdots \cap \mathcal{X}_{h+T+1}$, as well as $x_{<d} \stackrel{\text{def}}{=} x_0 + \cdots + x_{d-1}$ and $x_{\leqslant d} = x_{<d} + x_d$. Note that for every $d > 0$, if $V_0, \ldots, V_{d-1}$ are fixed and no randomness of the edges connecting the remaining nodes $V_{\geqslant d}$ to $V_{d-1} \cup V_{\geqslant d}$ is revealed, then the size X_d of V_d is binomially distributed with parameters $|V_{\geqslant d}|$ and $q_d \stackrel{\text{def}}{=} 1 - (1-p)^{x_{d-1}}$.

For all $d \geqslant 1$, we therefore have

$$\mathbb{P}[X_d = x_d \mid \mathcal{X}_{<d}] = \binom{n - x_{<d}}{x_d} \cdot q_d^{x_d} \cdot (1 - q_d)^{n - x_{\leqslant d}}\,, \quad \text{where } q_d \stackrel{\text{def}}{=} 1 - (1-p)^{x_{d-1}}. \tag{8.1}$$

Let us first look at the probability of seeing a concrete assignment of values $x_0, \ldots, x_{h+T+1}$ to the random variable X_d, including the value of x_ℓ for the random variable X_ℓ the nodes in V_h need to estimate. By applying (8.1) iteratively, we obtain

$$\begin{aligned}
\mathbb{P}[\mathcal{X}_{\leqslant h+T+1}] &= \mathbb{P}[\mathcal{X}_{<\ell} \wedge X_\ell = x_\ell \wedge \mathcal{X}_{>\ell}] \\
&= \mathbb{P}[\mathcal{X}_{<\ell}] \cdot \prod_{i=\ell}^{h+T+1} \mathbb{P}[X_i = x_i \mid \mathcal{X}_{<i}] \\
&= \mathbb{P}[\mathcal{X}_{<\ell}] \cdot \prod_{i=\ell}^{h+T+1} \binom{n - x_{<i}}{x_i} \cdot q_i^{x_i} \cdot (1 - q_i)^{n - x_{\leqslant i}}. \qquad (8.2)
\end{aligned}$$

We next analyze what happens to the above probability if the number of nodes in V_ℓ is only $x_\ell - 1$ instead of x_ℓ. Our goal is to show that this only changes the probability by a $1 - o(1)$ factor. If this is true for the most likely value x_ℓ, this will imply that the nodes in V_h can exactly estimate the value of X_ℓ at best with probability $1/2 + o(1)$. To analyze the above probability if $X_\ell = x_\ell - 1$, for all $d \geqslant 1$, we define the even $\mathcal{X}'_{<d}$ as follows. If $d \leqslant \ell$, we have $\mathcal{X}'_{<d} \stackrel{\text{def}}{=} \mathcal{X}_{<d}$ and if $d > \ell$, we have $\mathcal{X}'_{<d} \stackrel{\text{def}}{=} \mathcal{X}_{<\ell} \cap \{X_\ell = x_\ell - 1\} \cap \bigcap_{i=\ell+1}^{d-1} \{X_i = x_i\}$. We also define $\mathcal{X}'_{\leqslant d}$ analogously. We then have

$$\mathbb{P}[\mathcal{X}'_{\leqslant h+T+1}] = \mathbb{P}[\mathcal{X}_{<\ell}] \cdot \mathbb{P}[X_\ell = x_\ell - 1 \mid \mathcal{X}_{<\ell}] \cdot \prod_{i=\ell+1}^{h+T+1} \mathbb{P}[X_i = x_i \mid \mathcal{X}'_{<i}].$$

In order to compare $\mathbb{P}[\mathcal{X}_{\leqslant h+T+1}]$ and $\mathbb{P}[\mathcal{X}'_{\leqslant h+T+1}]$, we therefore need to compare $\mathbb{P}[X_\ell = x_\ell \,|\, \mathcal{X}_{<\ell}]$ and $\mathbb{P}[X_\ell = x_\ell - 1 \,|\, \mathcal{X}_{<\ell}]$, as well as $\mathbb{P}[X_i = x_i \,|\, \mathcal{X}_{<i}]$

and $\mathbb{P}[X_i = x_i \mid \mathcal{X}'_{<i}]$ for all $i \geqslant \ell + 1$. We have

$$\begin{aligned}
\mathbb{P}[X_\ell = x_\ell - 1 \mid \mathcal{X}_{<\ell}] &= \binom{n - x_{<\ell}}{x_\ell - 1} \cdot q_\ell^{x_\ell - 1} \cdot (1 - q_\ell)^{n - x_{\leqslant \ell} + 1} \\
&= \frac{\binom{n - x_{<\ell}}{x_\ell - 1} \cdot q_\ell^{x_\ell - 1} \cdot (1 - q_\ell)^{n - x_{\leqslant \ell} + 1}}{\binom{n - x_{<\ell}}{x_\ell} \cdot q_\ell^{x_\ell} \cdot (1 - q_\ell)^{n - x_{\leqslant \ell}}} \cdot \mathbb{P}[X_\ell = x_\ell \mid \mathcal{X}_{<\ell}] \\
&= \frac{x_\ell \cdot (1 - q_\ell)}{(n - x_{\leqslant \ell} + 1) \cdot q_\ell} \cdot \mathbb{P}[X_\ell = x_\ell \mid \mathcal{X}_{<\ell}]. \qquad (8.3)
\end{aligned}$$

For the following calculation, we define $q'_{\ell+1} = 1 - (1 - p)^{x_\ell - 1}$, i.e., $q'_{\ell+1}$ is the probability that a node outside $V_0 \cup \cdots \cup V_\ell$ is connected to V_ℓ, if we assume that $X_\ell = x_\ell - 1$ (instead of $X_\ell = x_\ell$). We obtain

$$\begin{aligned}
&\mathbb{P}[X_{\ell+1} = x_{\ell+1} \mid \mathcal{X}'_{<\ell+1}] = \mathbb{P}[X_{\ell+1} = x_{\ell+1} \mid \mathcal{X}_{<\ell} \wedge X_\ell = x_\ell - 1] \\
&= \binom{n - x_{\leqslant \ell} - 1}{x_{\ell+1}} \cdot \left(q'_{\ell+1}\right)^{x_{\ell+1}} \cdot (1 - q'_{\ell+1})^{n - x_{\leqslant \ell+1} + 1} \\
&= \frac{\binom{n - x_{\leqslant \ell} - 1}{x_{\ell+1}}}{\binom{n - x_{\leqslant \ell}}{x_{\ell+1}}} \cdot \left(\frac{q'_{\ell+1}}{q_{\ell+1}}\right)^{x_{\ell+1}} \cdot \left(\frac{1 - q'_{\ell+1}}{1 - q_{\ell+1}}\right)^{n - x_{\leqslant \ell+1} + 1} \cdot \\
&\qquad\qquad \mathbb{P}[X_{\ell+1} = x_{\ell+1} \mid \mathcal{X}_{<\ell} \wedge X_\ell = x_\ell] \\
&= \frac{n - x_{\leqslant \ell} - x_{\ell+1}}{n - x_{\leqslant \ell}} \cdot \left(\frac{q'_{\ell+1}}{q_{\ell+1}}\right)^{x_{\ell+1}} \cdot \left(\frac{1}{1 - p}\right)^{n - x_{\leqslant \ell+1} + 1} \cdot \qquad (8.4) \\
&\qquad\qquad \mathbb{P}[X_{\ell+1} = x_{\ell+1} \mid \mathcal{X}_{<\ell+1}].
\end{aligned}$$

Finally, for $i > \ell + 1$, we have

$$\begin{aligned}
\mathbb{P}[X_i = x_i \mid \mathcal{X}'_{<i}] &= \binom{n - x_{<i} + 1}{x_i} \cdot q_i^{x_i} \cdot (1 - q_i)^{n - x_{\leqslant i} + 1} \\
&= \frac{n - x_{<i} + 1}{n - x_{\leqslant i} + 1} \cdot (1 - q_i) \cdot \mathbb{P}[X_i = x_i \mid \mathcal{X}_{<i}]. \qquad (8.5)
\end{aligned}$$

Recall that our goal is to show that if the random variables $X_1, \ldots, X_{\ell-1}$ and $X_{\ell+1}, \ldots, X_{h+T+1}$ that are known to the nodes in V_h are close enough to their expectation, then for all reasonable values x_ℓ, when conditioning on the values of $X_1, \ldots, X_{\ell-1}$ and $X_{\ell+1}, \ldots, X_{h+T+1}$, $X_\ell = x_\ell$ and $X_\ell = x_\ell - 1$ have almost the same probability. We call an instance of the random graph in which the values of $X_1, \ldots, X_{\ell-1}$ and $X_{\ell+1}, \ldots, X_{h+T+1}$ are close enough to their expectation *well-behaved* and we formally denote this by an event $\mathscr{W}$. We next define the event $\mathscr{W}$ that specifies what it means that an instance is well-behaved.

We assume that the probability p that determines the presence of the individual edges is equal to $p = f(n)/n$, where $f(n) \geqslant 32 \ln n$ and $f(n) \leqslant \text{polylog} n$. The event $\mathscr{W}$ is defined as follows. For all $d \in \{1, \ldots, h+T+1\}$, it must hold that

$$|X_d - f(n) \cdot X_{d-1}| \leqslant \sqrt{7 \cdot f(n) \cdot X_{d-1} \cdot \ln n}. \tag{8.6}$$

Note that condition (8.6) and the assumption that $f(n) \geqslant 32 \ln n$ directly imply that $X_d \leqslant 1.5 f(n) \cdot X_{d-1}$ (even if $X_{d-1} = 1$). Therefore in well-behaved instances, X_d / X_{d-1} is at most $\text{polylog} n$. We choose the parameter $T = \Theta(\ln(n)/\ln\ln(n))$ small enough such that in well-behaved instances, $X_0 + \cdots + X_{h+T+1} \leqslant n^{1/3}$. Similarly, (8.6) implies that $X_d \geqslant 0.5 f(n) \cdot X_{d-1}$ and thus for any $d = \Theta(\log n / \log\log n)$, we have $X_d \geqslant n^{\nu}$ for some constant $\nu > 0$. We next show that a given instance (i.e., the neighborhood of a fixed node v_0 in a given random bipartite graph B) is well-behaved with probability $> 1 - 1/n$.

To see this, consider a given value $d \geqslant 1$. We know that once $X_1 = x_1, \ldots, X_{d-1} = x_{d-1}$ is given, X_d is binomially distributed with parameters $n - x_{<d}$ and $q_d = 1 - (1-p)^{x_{d-1}}$. By a standard Chernoff bound, for any $\delta \in [0,1]$, we therefore know that

$$\mathbb{P}\Big[\big|X_d - \mathbb{E}[X_d \mid \mathcal{X}_{<d}]\big| > \delta \cdot \mathbb{E}[X_d \mid \mathcal{X}_{<d}] \;\mid\; \mathcal{X}_{<d}\Big] \;\leqslant\; 2e^{-\delta^2/3 \cdot \mathbb{E}[X_d \mid \mathcal{X}_{<d}]}. \tag{8.7}$$

We will see that (8.7) implies that for every d, Inequality (8.6) holds with probability at least $1 - 2/n^2$. To achieve this, we first have to understand what the value of $\mathbb{E}[X_d \mid \mathcal{X}_{<d}]$ is. We first have a look at the probability q_d of the binomial distribution underlying X_d. We have

$$\begin{aligned} q_d = 1 - (1-p)^{x_{d-1}} &\leqslant p \cdot x_{d-1} \quad \text{and} \\ q_d = 1 - (1-p)^{x_{d-1}} &\geqslant 1 - e^{-p x_{d-1}} \\ &\geqslant p \cdot x_{d-1} - (p \cdot x_{d-1})^2 \\ &\geqslant p \cdot x_{d-1} \cdot \left(1 - \frac{f(n)}{n^{2/3}}\right). \end{aligned}$$

We used that for any real value $y \in [0,1]$ and any $k \geqslant 1$, we have $(1-y)^k \geqslant 1 - ky$ and $1 - y \leqslant e^{-y} \leqslant 1 - y + y^2$. In the last inequality, we have further used that in well-behaved executions, $x_{d-1} \leqslant n^{1/3}$. We have

$\mathbb{E}[X_d \mid \mathcal{X}_{<d}] = q_d \cdot (n - x_{<d})$. We therefore have

$$\begin{aligned}\mathbb{E}[X_d \mid \mathcal{X}_{<d}] &\leqslant p x_{d-1} \cdot n = f(n) \cdot x_{d-1},\\ \mathbb{E}[X_d \mid \mathcal{X}_{<d}] &\geqslant \left(1 - \frac{f(n)}{n^{2/3}}\right) \cdot p x_{d-1} \cdot (n - n^{1/3}) \geqslant \left(1 - \frac{2f(n)}{n^{2/3}}\right) \cdot f(n) \cdot x_{d-1}.\end{aligned}$$

Let ξ denote the maximum absolute deviation from the expectation that is still guaranteed to be well-behaved by (8.6). We can lower bound ξ as follows

$$\begin{aligned}\xi &\geqslant \sqrt{7 f(n) \cdot x_{d-1} \cdot \ln n} - \frac{2f(n)}{n^{2/3}} \cdot f(n) \cdot x_{d-1}\\ &\geqslant \sqrt{7\,\mathbb{E}[X_d \mid \mathcal{X}_{<d}] \cdot \ln n} - \frac{2f^2(n)}{n^{1/3}}\\ &\geqslant \sqrt{6\,\mathbb{E}[X_d \mid \mathcal{X}_{<d}] \cdot \ln n} = \sqrt{\frac{6 \ln n}{\mathbb{E}[X_d \mid \mathcal{X}_{<d}]}} \cdot \mathbb{E}[X_d \mid \mathcal{X}_{<d}].\end{aligned}$$

The last inequality holds if $n \geqslant n_0$ for a sufficiently large constant n_0. Together with (8.7), this now implies that for all d, (8.6) holds with probability larger than $1 - 2/n^2$. Note that there are less only $O(\log n / \log\log n)$ different d-values. If $n \geqslant n_0$ for a sufficiently large constant n_0, the number of different d-values can therefore for example be upper bounded by $\ln(n)/2$. In this case, a union bound over all d-values implies that the probability that an instance is well-behaved is at least $1 - \ln(n)/n^2$.

Let us now assume that we have a well-behaved instance (i.e., that $\mathscr{W}$ holds). Consider some assignment to the random variables $X_1, \dots, X_{\ell-1}$ and $X_{\ell+1}, \dots, X_{h+T+1}$ that are consistent with (8.6). Given the values of those random variables, the nodes in V_h have to guess the value of X_h. Note that there are extreme cases where the values of $X_1, \dots, X_{\ell-1}$ and $X_{\ell+1}, \dots, X_{h+T+1}$ only allow one single value of X_ℓ such that (8.6) is satisfied. We need to show that even when conditioning on $\mathscr{W}$, this only happens with a very small probability. Let us therefore define an even $\mathscr{W}' \subseteq \mathscr{W}$ as an instance in which replacing X_ℓ by $X_\ell - 1$ or $X_\ell + 1$ still satisfies (8.6). Note that the above analysis has enough slack to ensure that also $\mathbb{P}[\mathscr{W}'] \geqslant 1 - \ln(n)/n^2$ if $n \geqslant n_0$ for a sufficiently large constant n_0. The same analysis for example also works if the fixed constant 7 in (8.6) is replaced by any smaller fixed constant that is larger than 6. We therefore

have

$$\mathbb{P}[\mathscr{W}' \mid \mathscr{W}] = \frac{\mathbb{P}[\mathscr{W}' \cap \mathscr{W}]}{\mathbb{P}[\mathscr{W}]} = \frac{\mathbb{P}[\mathscr{W}']}{\mathbb{P}[\mathscr{W}]} \geqslant \mathbb{P}[\mathscr{W}'] \geqslant 1 - \frac{\ln n}{n^2}.$$

We use $x_1, \dots, x_{\ell-1}$ and $x_{\ell+1}, \dots, x_{h+T+1}$ to denote the concrete values of those random variables. We further define x_ℓ to be an arbitrary value such that the values x_ℓ and $x_\ell - 1$ are both valid values for X_ℓ to make the instance well-behaved. Note that if $\mathscr{W}'$ holds, there is at least one such value x_ℓ and we have seen that even conditioning on $\mathscr{W}$, the probability of $\mathscr{W}'$ is still at least $1 - \ln(n)/n^2$. Because we are assuming $\mathscr{W}$, we know that the value x_ℓ satisfies the following condition.

$$x_\ell = x_{\ell-1} \cdot np + \theta \ ,$$
$$\text{where} \quad |\theta| \in O(\sqrt{x_{\ell-1} f(n) \log n}) = O\left(\sqrt{\frac{f(n) \log n}{x_{\ell-1}}}\right) \cdot x_{\ell-1}.$$

We first look at the ratio between $\mathbb{P}[X_\ell = x_\ell - 1 \mid \mathcal{X}_{<\ell}]$ and $\mathbb{P}[X_\ell = x_\ell \mid \mathcal{X}_{<\ell}]$. By Equation (8.3), this ratio is equal to $\frac{x_\ell \cdot (1-q_\ell)}{(n - x_{\leqslant \ell} + 1) \cdot q_\ell}$. In the following, we denote this ratio by ρ_1. By using that $x_{\leqslant h+T+1} \leqslant n^{1/3}$, we can then bound ρ_1 as follows.

$$\begin{aligned}
\rho_1 &= \frac{x_\ell \cdot (1 - q_\ell)}{(n - x_{\leqslant \ell} + 1) \cdot q_\ell} \\
&\leqslant \frac{f(n) \cdot x_{\ell-1} \cdot \left(1 + O\left(\sqrt{f(n) \log(n)/x_{d-1}}\right)\right)}{\left(1 - \frac{1}{n^{2/3}}\right) \cdot n \cdot p x_{\ell-1} \cdot (1 - p x_{\ell-1})} \\
&\leqslant 1 + \frac{1}{n^{\Omega(1)}}.
\end{aligned}$$

In the last inequality, we used that $x_\ell = x_{\ell-1} \cdot np$, that $x_{\ell-1} \leqslant n^{1/3}$, and that $x_{\ell-1} \geqslant n^\kappa$ for some constant $\kappa > 0$. Similarly, we have

$$\begin{aligned}
\rho_1 &= \frac{x_\ell \cdot (1 - q_\ell)}{(n - x_{\leqslant \ell} + 1) \cdot q_\ell} \\
&\geqslant \frac{x_{\ell-1} \cdot \left(1 - O\left(\sqrt{f(n) \log(n)/x_{d-1}}\right)\right) \cdot (1 - p x_{\ell-1})}{n \cdot p x_{\ell-1}} \\
&\geqslant 1 - \frac{1}{n^{\Omega(1)}}.
\end{aligned}$$

We next look at the ratio between $\mathbb{P}[X_{\ell+1} = x_{\ell+1} \mid \mathcal{X}'_{<\ell+1}]$ and $\mathbb{P}[X_{\ell+1} = x_{\ell+1} \mid \mathcal{X}_{<\ell+1}]$. We denote this ratio by ρ_2. By Equation (8.4), ρ_2 can be

written as

$$\begin{aligned}
\rho_2 &= \frac{n - x_{\leqslant \ell} - x_{\ell+1}}{n - x_{\leqslant \ell}} \cdot \left(\frac{q'_{\ell+1}}{q_{\ell+1}}\right)^{x_{\ell+1}} \cdot \left(\frac{1}{1-p}\right)^{n - x_{\leqslant \ell+1}+1} \\
&\leqslant \left(\frac{p \cdot (x_\ell - 1)}{p \cdot x_\ell \cdot (1 - p \cdot x_\ell)}\right)^{x_{\ell+1}} \cdot \left(\frac{1}{1-p}\right)^{n - x_{\leqslant \ell+1}+1} \\
&\leqslant \left(1 + \frac{O(f(n))}{n^{1/3}}\right) \cdot \left(1 - \frac{1}{x_\ell}\right)^{x_{\ell+1}} \cdot \left(1 + \frac{p}{1-p}\right)^{n} \\
&\leqslant \left(1 + \frac{O(f(n))}{n^{1/3}}\right) \cdot e^{-\frac{x_{\ell+1}}{x_\ell}} \cdot e^{np + O(np^2)} \\
&\leqslant \left(1 + \frac{O(f(n))}{n^{1/3}}\right) \cdot e^{np - np + O(\sqrt{f(n)\log(n)/x_\ell})} \leqslant 1 + \frac{1}{n^{\Omega(1)}}.
\end{aligned}$$

In the last inequality, we used that because $\mathscr{W}$ holds, we have $x_{\ell+1} \leqslant x_\ell \cdot np + O(\sqrt{f(n) x_\ell \log n})$ and that $x_\ell \geqslant n^\nu$ for some constant $\nu > 0$. We can similarly lower bound ρ_2 as follows.

$$\begin{aligned}
\rho_2 &\geqslant \frac{n - n^{1/3}}{n} \cdot \left(\frac{p(x_\ell - 1)(1 - p(x_\ell - 1))}{p \cdot x_\ell}\right)^{x_{\ell+1}} \cdot \left(\frac{1}{1-p}\right)^{n - n^{1/3}} \\
&\geqslant \left(1 - \frac{O(f(n))}{n^{1/3}}\right) \cdot \left(1 - \frac{1}{x_\ell}\right)^{x_{\ell+1}} \cdot (1+p)^n \\
&\geqslant \left(1 - \frac{O(f(n))}{n^{1/3}}\right) \cdot e^{-\frac{x_{\ell+1}}{x_\ell}} \cdot \left(1 - \frac{1}{x_\ell^2}\right)^{x_{\ell+1}} \cdot e^{pn} \cdot (1-p^2)^n \\
&\geqslant \left(1 - \frac{O(f(n))}{n^{1/3}}\right) \cdot \left(1 - \frac{x_{\ell+1}}{x_\ell^2}\right) \cdot (1 - p^2 n) \cdot e^{np - np - O(\sqrt{np\log(n)/x_\ell})} \\
&\geqslant \left(1 - \frac{O(f(n))}{n^{1/3}}\right) \cdot \left(1 - \frac{O(f(n))}{x_\ell}\right) \cdot \left(1 - O\left(\frac{\sqrt{np\log(n)}}{\sqrt{x_\ell}}\right)\right) \\
&\geqslant 1 - \frac{1}{n^{\Omega(1)}}.
\end{aligned}$$

In the third inequality, we use that for $y \in [-1, 1]$, it holds that $1 + y \geqslant e^y \cdot (1 - y^2)$. In the last inequality, we again used that $x_{\ell-1}$ and x_ℓ are both of size $\geqslant n^\nu$ for some constant $\nu > 0$.

Finally, let us look at the ratio between the probabilities $\mathbb{P}[X_i = x_i | \mathcal{X}'_{<i}]$ and $\mathbb{P}[X_i = x_i \mid \mathcal{X}_{<i}]$ for $i > \ell + 1$. We denote this ratio by $\rho_{3,i}$, and by using Equation (8.5), we can bound $\rho_{3,i}$ as follows.

$$1 - O\left(\frac{f(n)}{n^{2/3}}\right) \leqslant \rho_{3,i} = \frac{n - x_{<i} + 1}{n - x_{\leqslant i} + 1} \cdot (1 - q_i) \leqslant 1 + O\left(\frac{f(n)}{n^{2/3}}\right).$$

We therefore have

$$\frac{\mathbb{P}[\mathcal{X}'_{\leqslant h+T+1} \mid \mathscr{W}']}{\mathbb{P}[\mathcal{X}_{\leqslant h+T+1}) \mid \mathscr{W}']} \;=\; \rho_1 \cdot \rho_2 \cdot \prod_{i=\ell+2}^{h+T+1} \rho_i \;=\; 1 \pm o(1).$$

Recall that $\mathcal{X}_{\leqslant h+T+1} = \mathcal{X}_{<\ell} \cap \{X_\ell = x_\ell\} \cap \mathcal{X}_{>\ell}$ and $\mathcal{X}'_{\leqslant h+T+1} = \mathcal{X}_{<\ell} \cap \{X_\ell = x_\ell - 1\} \cap \mathcal{X}_{>\ell}$. We therefore have

$$\mathbb{P}[\mathcal{X}_{\leqslant h+T+1} \mid \mathscr{W}'] = \mathbb{P}[X_\ell = x_\ell \mid \mathcal{X}_{<\ell} \cap \mathcal{X}_{>\ell} \cap \mathscr{W}'] \cdot \mathbb{P}[\mathcal{X}_{<\ell} \cap \mathcal{X}_{>\ell} \mid \mathscr{W}']$$

and

$$\Pr \mathcal{X}_{\leqslant h+T+1} \mid \mathscr{W}' = \mathbb{P}[X_\ell = x_\ell - 1 \mid \mathcal{X}_{<\ell} \cap \mathcal{X}'_{>\ell} \cap \mathscr{W}'] \cdot \mathbb{P}[\mathcal{X}_{<\ell} \cap \mathcal{X}'_{>\ell} \mid \mathscr{W}']$$

and thus

$$\frac{\mathbb{P}[X_\ell = x_\ell - 1 \mid \mathcal{X}_{<\ell} \cap \mathcal{X}'_{>\ell} \cap \mathscr{W}']}{\mathbb{P}[X_\ell = x_\ell \mid \mathcal{X}_{<\ell} \cap \mathcal{X}_{>\ell} \cap \mathscr{W}']} \;=\; \frac{\mathbb{P}[\mathcal{X}'_{\leqslant h+T+1} \mid \mathscr{W}']}{\mathbb{P}[\mathcal{X}_{\leqslant h+T+1} \mid \mathscr{W}']} \;=\; 1 \pm o(1).$$

However, this means that if $\mathscr{W}'$ holds, the interval of possible values for X_ℓ contains at least two values and for any two adjacent values, the conditional probability that this is the correct guess is equal up to a $1 \pm o(1)$ factor. Hence, even for the possible value x^*_ℓ that maximizes $\mathbb{P}[X_\ell = x^*_\ell \mid \mathcal{X}_{<\ell} \cap \mathcal{X}_{>\ell} \cap \mathscr{W}']$, we have $\mathbb{P}[X_\ell = x^*_\ell \mid \mathcal{X}_{<\ell} \cap \mathcal{X}_{>\ell} \cap \mathscr{W}'] \leqslant 1/2 + o(1)$. Thus, if $\mathscr{W}'$ holds, the nodes in V_h exactly estimate X_ℓ with probability better than $1/2 + o(1)$. Because $\mathscr{W}'$ holds with probability $1 - o(1)$, even if the nodes in V_h always succeed in case $\mathscr{W}'$ does not hold, the probability that V_h can correctly guess X_ℓ is still at best $1/2 + o(1)$. If the number of nodes $n \geqslant n_0$ for a sufficiently large constant n_0, this is at most $2/3$, which proves the claim of the theorem. ∎

We note that the success probability of $1/3$ could be boosted significantly in several ways. First, note that it would not be hard to adapt the proof so that for some constant $\nu > 0$, $\mathscr{W}$ allows n^ν different values for X_ℓ and that the probabilities for the n^ν most likely values are all approximately the same. This reduces the success probability to $n^{-\Omega(1)}$. Further, instead of looking at one neighborhood in the graph, we could look at polynomially many independent and disjoint neighborhoods and thus make the success probability even exponentially small in n^ν for some constant $\nu > 0$.

Given the lower bound on computing a perfect matching in a random bipartite graph, our main lower bound theorem now follows in a relatively

straightforward fashion. The following is a more precisely phrased version of Theorem 6.2.

THEOREM 8.3. *Assume that each node v of a complete graph K_n on n nodes uniformly and independently computes a subset S_v of the colors $\{1, \dots, n\}$ as follows. Each color x is included in S_v independently with probability $f(n)/n$, where $f(n) \geqslant c \ln(n)$ for a sufficiently large constant c and $f(n) \leqslant \operatorname{poly} \log n$. Let G be the subgraph of K_n defined by all n nodes and the set of edges between nodes u and v with $S_u \cap S_v \neq \emptyset$. Any randomized* LOCAL *algorithm on G to properly color K_n with colors from the sets S_v requires $\Omega(\log n / \log \log n)$ rounds. The lower bound holds even if the algorithm only has a success probability of $2/3$.*

PROOF. We define a bipartite graph B between the set of nodes $V = \{1, \dots, n\}$ and the set of color $C = \{1, \dots, n\}$. There is an edge between $v \in V$ and $x \in C$ iff $x \in S_v$. We note that since for every color x and every node v, $\Pr x \in S_v = f(n)/n$ and those probabilities are independent for different pairs (v, x), the bipartite graph on V and C contains each possible edge between V and C independently with probability $p = f(n)/n$. Further, a valid n-coloring of K_n is a one-to-one assignment between nodes and colors. Therefore, each valid n-coloring of K_n that respects the sampled color set corresponds to a perfect matching in the bipartite graph B between V and C and vice versa. Also note that clearly, in the LOCAL model, any LOCAL algorithm on B can be run on G with only constant overhead, and vice versa (when simulating B on G, each color node x has to be simulated by one of the nodes v for which $x \in S_v$). Hence, any distributed coloring algorithm for K_n that runs on the sampled graph G implies a perfect matching algorithm on B with the same asymptotic round complexity. The theorem therefore directly follows from Theorem 8.2. ∎

Part III

Coloring Virtual Graphs

CHAPTER 9

Virtual Graphs

Most distributed graph algorithms assume communication on the input graph. Namely, that the graph that forms the input to the computational problem at hand is equivalent to the communication network infrastructure. 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 [GK13; GH16; GKKLP18; RG20; GGR21; FGLPSY21; GZ22; RGHZL22; GGHIR23]) that is salient when using a CONGEST algorithm as a subroutine (e.g., local rounding [FGGKR23] used in [GGHIR23]) or when modifying the input graph (e.g., contracting edges [GKKLP18; FGLPSY21]). Part III attempts to study *how bandwidth constraints affect distributed algorithms solving problems on graphs whose description is itself distributed on a communication network*. We focus on symmetry breaking and thus ask

> *How efficiently can H be $(\Delta+1)$-colored when distributed on a network G?*

We devise a notion of embedding to formalize what it means for a graph H to be known only distributively on a communication network G. Albeit technical, it naturally arises from the aforementioned examples from the literature.

As we have seen earlier, in LOCAL, graphs can be $(\Delta + 1)$-colored in $\text{poly}(\log \log n)$ rounds and even in $O(\log^* n)$ rounds when Δ is large enough.

In this part, we extend these results to embedded graphs in nearly the same number of rounds. We devise poly($\log \log n$)-round $(\Delta+1)$-coloring algorithms. For $(\Delta+1)$-coloring, we are even able to obtain a $O(\log^* n)$-round algorithm when Δ is large enough. Conversely, we show that algorithms designed for such situations are inherently limited by the *dilation* and *congestion* of the embedding.

Organization of this Chapter. Before we formalize definitions of embedding, congestion and dilation, we make a detour by a simpler type of embedding widely studied in the distributed graph literature called *cluster graphs*. While being conceptually simpler than the more general notion of *virtual graph* introduced in Section 9.2, they already capture the main challenges we will encounter in this part of the thesis.

We then present the formal results of Part III and list some direct corollaries for problems that have been studied in the literature. We then give a high-level overview of the challenges of coloring virtual graphs compared to Parts I and II and how we solve them.

9.1. A Simpler Case: Cluster Graphs

Throughout Part III, $G = (V_G, E_G)$ refers to the communication *network* with $n = |V_G|$ vertices called *machines* and edges called *links* with b bandwidth. We assume each machine $v \in V_G$ is provided a $O(\log n)$-bits unique identifiers $\mathsf{ID}(v)$ 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 $\mathsf{b} = \Theta(\log n)$. We wish to color a *cluster graph* defined over this communication network.

> A *cluster graph* is a graph $H = (V_H, E_H)$ defined over a communication network $G = (V_G, E_G)$ by partitioning the machines of G into disjoint connected *clusters* of machines. Each node v of H corresponds to a cluster $V(v)$ in G, and two nodes in H are connected if and only if their respective clusters are adjacent in G.

Figure 1. A communication graph G with 4 clusters (left), and the associated cluster graph H (right).

Cluster graphs appear in several places, from maximum flow algorithms [GKKLP18; FGLPSY21] to network decomposition [RG20; GGR21]. See Figure 1 for an example. They arise naturally when algorithms contract edges, for instance in [GKKLP18; FGLPSY21]. Vertices of the input graph then become connected *sets of vertices* in the communication graph. They can then be seen as low-diameter trees on the communication graphs and each step of the algorithm must be simple enough to be implemented through aggregation.

Notice that this strictly generalizes the CONGEST model where the input and communication graph coincide $H = G$ and all the clusters $V(v) = \{v\}$ are reduced to a single machine. We also emphasize that communication on cluster graphs is more permissive than in the setting of Part II since it allows for efficient aggregation over messages from *all* neighbors. Nonetheless, communication remains too restricted to naively simulate a CONGEST algorithm on H.

Importantly, the clusters might be internally poorly connected. Imagine, for instance, that each cluster is a tree with Δ leaves in the network. In that case, communications between different parts of the tree must go through one $O(\log n)$-bandwidth link. In particular, this means that any local computation by vertices of H must admit an efficient communication protocol, which is generally not true of CONGEST algorithms.

Two natural primitives that illustrate this issue are determining a node's degree and finding a free color. In CONGEST, the degree of a node is simply its number of incident edges. A simple aggregation within a cluster suffices to count its incident links. But this quantity can grossly overestimate the cluster's actual degree, given that two clusters can be connected through many links (as in Figure 1). In fact, determining which edges connect to

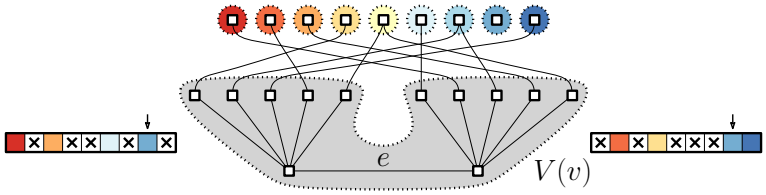

Figure 2. An uncolored cluster $V(v)$ with a tree-topology (in gray) and edges to colored neighbors on both sides of a bridge link e. Finding the one available color *by communicating in* $V(v)$ *only* is at least as hard as solving a set-intersection instance where Alice (resp. Bob) encodes her input on the left (resp. right) inter-cluster links and all communication must go through the central $O(\log n)$-bit bandwidth link.

the same cluster amounts to a costly set intersection problem. Finding a color available to v by local computation in $V(v)$ alone requires $\Omega(\Delta/\log n)$ rounds in the worst case by a similar reduction (see Figure 2).

However, the aforementioned two tasks are easily performed for a node v with the dedication of its neighbors. Indeed, the neighbors of v can each perform an aggregation to cut off all but one link to $V(v)$ to prevent double-counting. This allows v to compute its degree exactly in one aggregation, and to find a free color using binary search. Where the difficulty lies is in performing these tasks in parallel across the whole network. This observation is useful to our algorithm in several parts, as we perform computations within disjoint subgraphs of the cluster graph.

Coloring Cluster Graphs with the DPS Theorem. Recall that in Part II, we provided an algorithm based on the DPS Theorem to $(\Delta+1)$-color cluster graphs in $\text{poly}(\log n)$ rounds of CONGEST (see Section 6.3.2). However, this approach cannot be improved beyond $\widetilde{\Omega}(\frac{\log n}{\log\log n})$ as we proved in Chapter 8. As we shall see in this part of the thesis, using the DPS is too restrictive as it allows vertices to communicate with only $O(\log^4 n)$ neighbors while cluster graphs allow aggregation over the whole neighborhood.

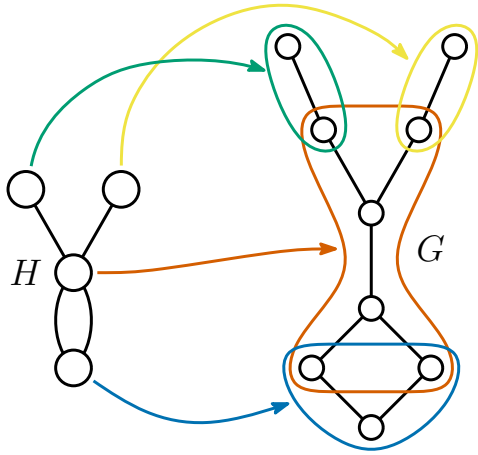

Figure 3. A virtual graph H (on the left) embedded on a network G (on the right). On this example, there is a unique choice of support trees; they have congestion $\mathsf{c} = 1$ and dilation $\mathsf{d} = 3$.

9.2. The More General Framework: Virtual Graphs

Our notion of virtual graphs generalizes cluster graphs by allowing clusters to overlap, i.e., they do not partition machines anymore. Before giving the formal definition, we begin with a high-level discussion of the concept.

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 3 exemplifies such an embedding.

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

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

all vertices on the support; second, machines at intersections of supports perform local computations; third, converge-cast the 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 [FGLPSY21], 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 3, 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.

Formally Defining Virtual Graphs. 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 9.1 (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 9.1. 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 9.2 (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 frequently say vertices do something (e.g., try a color) to refer to such an aggregation process. It can also sometimes be useful to root support trees at some machine called the *leader*.

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,

$$\mathsf{c} \stackrel{\text{def}}{=} \max_{e \in E_G} |T^{-1}(e)| \quad \text{and} \quad \mathsf{d} \stackrel{\text{def}}{=} \max_{v \in V_H} \left(\max_{u,u' \in T(v)} \text{dist}_{T(v)}(u, u') \right). \tag{9.1}$$

Congestion and dilation are natural bottlenecks for virtual graphs. In Chapter 10, 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, Chapter 12 shows that we can $(\Delta+1)$-color virtual graphs in $O(\mathsf{cd}\cdot\log^* n)$ rounds when $\Delta \geqslant \text{poly}(\log n)$, and Chapter 13 shows that coloring in $\mathsf{cd} \cdot \text{poly}(\log\log n)$ rounds is feasible for any embedding.

Degree vs Pseudo-Degree. Before we state our results, some remarks are in order on the way degrees are defined and by which vertices can access them. In multi-graphs, there are two ways to define the degree of a vertex: by counting the number of incident edges or by counting the number of adjacent vertices. In this thesis, *degrees always refer to the number of adjacent vertices*. However, as explained in Section 9.1, counting exact degrees in cluster graphs — thus in virtual graphs — is expensive. So we give a name to the number of incident edges:

Definition 9.3. The *pseudo-degree* of v, denoted by $\widehat{\deg}(v)$, is the number of edges incident to v.

Contrary to degrees, pseudo-degrees can be computed by aggregation on support trees as

$$\widehat{\deg}(v) = \sum_{w \in T(v)} |m^{-1}(w)| \ ,$$

which is why we ask that edges have designated handlers.

We emphasize that 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. We refer readers to Section 11.1 for more details on this issue.

9.3. Our Contributions

We can now state the results of Part III.

Lower Bounds. We show that the congestion c and the dilation d essentially capture the hardness of the coloring problem. On one hand, they limit the efficiency of any $(\Delta+1)$-coloring algorithm:

THEOREM 9.4. *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 arbitrary embeddings. However, they do not necessarily follow the three-step process described in Section 9.2.

Sub-Logarithmic $(\Delta+1)$-Coloring. We give an affirmative answer to our research question with Theorem 9.5. We provide an algorithm to $(\Delta+1)$-color virtual graphs nearly as fast as the best known LOCAL algorithm (Theorem 3.1). It also improves exponentially on the only existing algorithm for $(\Delta+1)$-coloring virtual graphs (Corollary 6.6).

THEOREM 9.5. *Let H be a virtual graph on network G with n machines, bandwidth* $\mathsf{b} = O(\log n)$*, congestion* $\mathsf{c} \leqslant n$ *and dilation* d*. Suppose that Δ is the maximum degree of H and that it is known to all machines. There is a* $O(\mathsf{cd} \cdot \log^7 \log n)$*-round algorithm to* $(\Delta+1)$*-color H, with high probability.*

Our result can be seen as a win for *decentralization* in a distributed context. In more powerful distributed models like LOCAL, the distributed aspect often revolves around *gathering* the information, which is then computed and acted on by a single processor. In virtual graphs, the operation of each vertex is dispersed between multiple machines and these machines can only share a synopsis of the messages they receive. Computation in virtual graphs must therefore be collaborative and primarily based on aggregation (especially when Δ is large).

Ultrafast $(\Delta+1)$-Coloring High-Degree Graphs. When Δ is larger than some $\text{poly}(\log n)$, the LOCAL (and CONGEST) algorithm colors all the vertices in $O(\log^* n)$ rounds with high probability. We are able to obtain the same for virtual graphs of high degree.

THEOREM 9.6. *Let H be a virtual graph on network G with n machines, bandwidth $\mathsf{b} = O(\log n)$, congestion $\mathsf{c} \leqslant n$ and dilation d. Suppose that Δ is the maximum degree of H and that it is known to all machines. There is a $O(\mathsf{cd} \cdot \log^* n)$-round algorithm to $(\Delta+1)$-color virtual graphs of maximum degree $\Delta \geqslant \Delta_{low} = \Omega(\log^{21} n)$, with high probability.*

This complexity is essentially tight given the slow growth of the log-star function and the $\Omega(\log^* n)$ round lower bound for $(\Delta+1)$-coloring constant degree graphs by Linial.

Sub-Logarithmic $(\widehat{\deg}+1)$-Coloring. In [FHN24], we also provide an algorithm using lists of local sizes. While the pseudo-degrees can be larger than Δ, is can also be much smaller at some vertices.

THEOREM 9.7 ([FHN24]). *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 $\widehat{\deg}+1$-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 $\varphi(v) \in \{1, 2, \ldots, \widehat{\deg}(v)+1\}$ where $\widehat{\deg}(v)$ is the number of edges incident to v in H.*

A key reason for considering the $\widehat{\deg}+1$-coloring problem is that we forgo the frequently assumed global knowledge about the maximum degree Δ. This is the source of substantial technical challenges compared to

Theorem 9.5 — albeit the general framework remains the same — and we chose not to include it in this thesis in for the sake of conciseness. We refer readers to [FHN24] for details on Theorem 9.7.

9.4. Applications of Virtual Graphs

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]

Application 1: Cluster Graphs. Clearly, cluster graphs (recall the definition from Section 6.3.2 or Section 9.1) are captured by our definition of virtual graphs. For each cluster $C_x \in V_H$, 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 $V(C_x)$. Theorems 9.5 and 9.6 imply we can color cluster graphs fast:

Corollary 9.8. *Cluster graphs can be $(\Delta+1)$-colored, w.h.p., in $O(\mathsf{d} \cdot \log^7 \log n)$ CONGEST rounds where d is the maximum (strong) diameter of a cluster, i.e., of $H[C_x]$ over all C_x. When $\Delta \geqslant \text{poly}(\log n)$, it can be colored in $O(\log^* n)$ rounds.*

Recall that prior to Corollary 9.8, the only non-trivial algorithm for $(\Delta+1)$-coloring cluster graphs was the $\text{poly}(\log n)$-round algorithm described in Section 6.3.2.

[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.

Meanwhile, Theorem 9.7 gives the first non-trivial distributed algorithm for $(\Delta+1)$-coloring cluster graphs:

Corollary 9.9. *Cluster graphs can be $(\Delta+1)$-colored, w.h.p., in $O(\mathsf{d} \cdot \text{poly}(\log\log n))$ CONGEST rounds where d is the maximum (strong) diameter of a cluster, i.e., of $H[C_x]$ over all C_x. When $\Delta \geqslant \text{poly}(\log n)$, it ends in $O(\mathsf{d}\log^* n)$ rounds with high probability.*

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 $\text{dist}_G(u,v) \leqslant t$. 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 $t \geqslant 1$. [3]

Lemma 9.10. *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 Δ^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 $dist_G(u,v) > t$. 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 Δ^t+1 colors and from the definition of edges in H, a proper coloring on H is also proper on G^t.

[3] Another way to generalize both distance-1 and distance-2 coloring is through the *relay model*. The communication graph G is bipartite with vertices (of H) on one side and *relay* nodes on the other side. Each vertex v has as support tree all the incident edges in G. This is a star graph (i.e., of radius 1). The congestion on an edge is 1 and dilation is 2. All pairs of nodes whose support trees intersect form an edge in H. Thus, both the support trees and edge handling are implicitly given.

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 $(\mathsf{ID}(u), i)$ 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 9.7 and Lemma 9.10 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 9.11. *For any $t \geqslant 1$, there is a randomized* CONGEST *algorithm for Δ^t+1-coloring G^t that runs in $O(t\Delta^{\lfloor (t-1)/2 \rfloor}\mathrm{poly}(\log\log n))$ rounds with high probability. When $\Delta \geqslant \mathrm{poly}(\log n)$, then it ends in $O(t\Delta^{\lfloor (t-1)/2 \rfloor}\log^* n)$ rounds with high probability.*

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 9.11 can be improved to $O(\Delta^{\lfloor (t-1)/2 \rfloor}\mathrm{poly}(\log\log n))$ 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 [FHN20]. However, no super-constant lower bounds are known for computing distance-t colorings in CONGEST when $t \geqslant 3$ and $\Delta \gg \log n$.

9.5. Challenges of Virtual Graphs

Our algorithm on virtual graph has the same general structure as the BCONGEST algorithm from Chapter 4. First, we generate slack by having each vertex try a random color in $[\Delta + 1]$, in almost-cliques we perform additional color trials to increase the number of repeated color — i.e., we compute a colorful matching — and in the densest almost-cliques we find put-aside sets. Sparse vertices and certain outliers in each almost-cliques can then be colored in $O(\log^* n)$ rounds by SlackColor; most dense vertices are colored by the synchronized color trial. For each of these steps, we make sure to avoid certain reserved colors in each almost-clique, where the number of reserved colors depends on the sparsity/density of that almost-clique. After some careful setup, we can run SlackColor with reserved colors to color almost-all remaining dense vertices in $O(\log^* n)$ rounds. The only remaining vertices are in put-aside sets.

With this in mind, let us review the main challenges in virtual graphs.

Inaccurate Degrees. The first challenge in virtual graphs is that vertices cannot efficiently compute their degrees (external degree, anti-degree, etc) exactly — at least not all at the same time. This is a problem since the number of reserved colors (Eq (4.10)) or definition of outliers (Eq (4.9)) in Chapter 4 is based on precise counting arguments over such quantities.

Using properties of geometric random variables, we design a technique we call *fingerpringing* to approximate such quantities up to a small constant factor. Still, anti-degree cannot be approximated well because, roughly speaking, it would require aggregation over anti-edges. As a result, we set the number of reserved colors and define outliers more intricately than in Chapter 4.

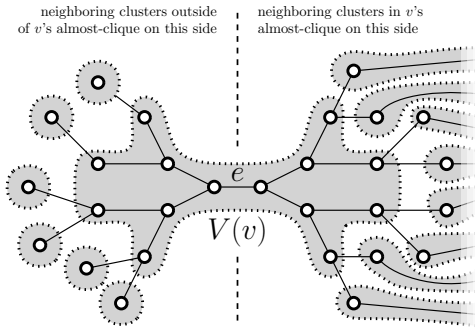

Figure 4. All machines linked to the inside, on the right, are linked to the outside, on the left, by a unique link e.

Setting-Up SlackColor. In Chapter 4, right after the synchronized color trial and just before run SlackColor in almost-cliques, we run some extra color trials to take care of vertices that have too few available reserved colors. Albeit it mostly consists of trying random colors — which can easily be done on virtual graph — it also requires that vertices know if they have sufficiently many available reserved colors. Like for degrees, estimating this on virtual graphs is hard because of double counting. We use the fingerprinting technique and show that, by computing the number of colored neighbors precisely enough, the steps of the BCONGEST algorithm can be replicated on virtual graphs.

Coloring Put-Aside Sets. Surprisingly, the most challenging part in virtual graphs is coloring the put-aside vertices. For succinctness, we call *cabals* the almost-cliques in which we compute a put-aside set. Those are the ones with average external degree $O(\log^{1.1} n)$.

In LOCAL, this step is trivial because each put-aside set has diameter at most two and put-aside sets from different cabals are independent (i.e., no edge connects them). In CONGEST, one can observe that it suffices to learn the set of colors used by external neighbors of put-aside vertices as well as the clique palette (the colors not used in the almost-clique). Because cabals are so dense, this fits in $O(\log^{2.1} n)$ bits, which with some routing tricks can be disseminated efficiently in cabals, even with $\Theta(\log n)$-bit messages.

In virtual graphs however, the machines inside a cabal can be linked to the outside only through a single $O(\log n)$-bandwidth link, making routing

near-impossible (see Figure 4). As explained in Section 9.1, the mere task of finding a color for a single vertex is hard (recall Figure 2). Finally, observe that to obtain the $O(\log^* n)$ runtime claimed in Theorem 9.6, we must color all put-aside vertices ultrafast.

To solve this issue, we do not attempt to extend the coloring to put-aside sets (like in LOCAL and CONGEST), but will use the help of already colored vertices. This leads to a novel and non-trivial parallel color swapping scheme outlined in the following section.

Finding Anti-Edges in Cabals. Since vertices cannot learn their palettes, we approximate them by the *clique palette*: the set of colors not used in the almost-clique. The issue is that in almost-cliques larger than $\Delta + 1$, the clique palette may run out of colors before coloring all its vertices. As for Parts I and II, we remedy this by using some colors multiple times such almost-cliques. This step was referred to as computing a colorful matching. As observed in Section 4.6, when the average anti-degree in the almost-clique is $\Omega(\log n)$, repeating $O(1)$ random color trials works with high probability. Additionally, if the vertices of the almost-clique are sufficiently sparse, they receive enough slack from slack generation (i.e., a single random color trial). In the densest cabals, with fewer than $O(\Delta \log n)$ anti-edges, this approach does not work.

We resort to a non-conventional sampling technique based on *geometric variables*. It allows us to perform $O(\log n)$ parallel random trials, where each trial costs 2 bits on average. Conveniently, geometric variables are easy to aggregate even in the presence of redundant paths and, surprisingly, this can be used to find anti-edges in cabals with high probability. Once we found enough anti-edges, they can be colored in $O(\log^* n)$ rounds using SlackColor.

Low-Degree Graphs. Like for the LOCAL and BCONGEST algorithm, when $\Delta \leqslant \text{poly}(\log n)$, we use a slightly different approach based on shattering, for we cannot ensure that SlackColor is likely enough to color all vertices. Previously, for low-degree graphs, we could rely on random color trials to shatter the graph and the deterministic algorithm by Ghaffari and Kuhn to extend to coloring to the few remaining vertices (see Section 3.7).

There are two difficulties with this scheme in virtual graphs: first, the shattering technique of [BEPS16] requires that vertices know their palette; second, some parts of the Ghaffari-Kuhn algorithm assume that vertices can compute certain degree-like quantities, for which naive aggregation fails because of possible double counting.

To solve the former issue, we use the techniques of the high-degree graph algorithm to reduce degrees to $O(\log n)$. From then on, vertices can learn their palette in $O(\mathsf{cd})$ rounds, hence run [BEPS16]. To solve the former, we show that we can use the fingerpringing technique designed to approximate degrees to implement the Ghaffari-Kuhn with only a multiplicative $\text{poly}(\log \log n)$ round overhead.

9.6. Technical Overview

In this overview, we focus on the most novel ideas of Part III: the use of fingerprinting to find anti-edges in cabals, and a novel color swapping technique for coloring put-aside sets. Recall that cabals are almost-cliques for which the average external degree is $O(\log^{1.1} n)$.

Finding Anti-Edges in Cabals. Consider a cabal in which the average anti-degree is $O(\log n)$, so that Algorithm 14 does not find a large enough colorful matching with high enough probability. We explain how to find a matching of α anti-edges, where α is the median average anti-degree. Given such an anti-matching, we can same-color its anti-edges in $O(\log^* n)$ rounds with SlackColor, thereby providing enough slack to at least half of the vertices of the cabal (the half that has anti-degree at most α).

To detect anti-edges, vertices will aggregate the maximum of certain geometric variables. In this part, we call such aggregates *fingerprints* and use them extensively in our algorithm. Primarily, we employ them to approximate fundamental quantities, such as the degree of vertices within a specific set (see Lemma 11.17). As discussed in the previous section, vertices in virtual graphs cannot compute such basic quantities efficiently. To the best of our knowledge, approximate counting or similarity estimations based on geometric variables is novel in distributed graph algorithms; however, the idea has been used and studied extensively in other contexts, e.g., in [FM85; DF03; Pat07; KNW10; Bla20].

To illustrate how we use fingerprints in this context, let us consider a simplified setting where K is a cabal such that at least $\Delta/2$ of its vertices are incident to an anti-edge and we only care to identify one of them. Every vertex samples a random geometric variable $\mathbf{X}_u$ and we compute their maximum $\mathbf{Y}^K$. Additionally, each vertex $v \in K$ computes the maximum $\mathbf{Y}^v$ of the variables in its neighborhood (in K). Observe that if $\mathbf{Y}^v \neq \mathbf{Y}^K$, we know that v is the endpoint of an anti-edge. In particular, it happens when $\mathbf{Y}^K = \mathbf{X}_u$ for a *unique* u in K and that u has some anti-neighbor. It is not very difficult to verify that the maximum is unique with constant probability (Lemma 11.13) and that it occurs at a uniform vertex in K (Lemma 11.14). Thus, this scheme identifies an anti-edge with constant probability. Analysis shows that repeating this random experiment $\Theta(\alpha^{-1} \log n)$ time in parallel yields an α-sized matching of anti-edges with high probability (Lemma 12.15).

The core advantage of using fingerprints resides in that they can be compressed very efficiently (Lemma 11.16). It might seem that since $\mathbf{X}_v \leqslant O(\log n)$ w.h.p., we represent each fingerprint in $O(\log \log n)$ bits, resulting in a total bandwidth usage of $O(\log n \cdot \log \log n)$ bits. However, we leverage the fact that $\mathbf{Y}^v$ values are highly concentrated and encode *their deviation* from some baseline value rather than their actual value. Since we only expect each trial to diverge by a constant amount, a concentration argument shows that the total deviation over all trials is $O(\log n)$ with high probability (Lemma 11.15).

Coloring Put-Aside Sets. We now describe the most novel and technically involved idea in our coloring algorithm. To convey intuition without overloading the reader with technical details, we consider a simplified setting that illustrates the key ideas: let $\Delta \geqslant \Omega(\log^{21} n)$, set $r = \Theta(\log^{1.1} n)$ and suppose H consists of $(\Delta + 1 - r)$-cliques, where each vertex has r external neighbors (i.e., neighbors outside their clique).

We suppose that we already computed a coloring φ of almost all nodes: in each clique K, only a *put-aside* set P_k of exactly r vertices is uncolored. Recall that no edges connect put-aside sets from different cliques. Our claim is that we can fully color H in constant rounds.

However, searching for available colors is computationally expensive (recall Figure 2). Instead, we ask the colored vertices to assist by offering to *donate their color* to put-aside vertices. Intuitively, this approach is more efficient because it shifts the burden from a few vertices searching for available colors to many vertices finding ways to recolor themselves and donate their colors. Nonetheless, donations are feasible only if

(1) each donated color is not used by an external neighbor of the vertex receiving it;
(2) each *donor* has a *replacement color* (or *replacement*, for short); and
(3) donors have different replacement colors.

Our coloring algorithm thus solves a three-way matching problem in each clique: uncolored vertices are matched to donors and donors to replacements (see Figure 5). We also need to ensure that recolorings do not create conflicts between different cliques.

Let us review the steps for computing this three-way matching.

Step 1: Finding Candidate Donors. We first compute a set $Q_K \subseteq K \setminus P_K$ of $\Omega(\Delta/r)$ candidate donors in each clique K such that no edges connect Q_K to candidate & put-aside sets in other cliques. This ensures that each K can be recolored independently. The sets Q_K are constructed similarly to put-aside sets (see Algorithm 33). We henceforth focus on a fixed clique K with uncolored vertices $u_1, u_2, \ldots, u_r$.

Step 2: Finding Replacement Colors. While it can be expensive to learn the colors available to a given node, we can quickly identify which colors are not used in K. Extending a technique of [FHN23], we show that a node can query the i-th color in the clique palette $L(K) \stackrel{\text{def}}{=} [\Delta + 1] \setminus \varphi(K)$, i.e., the set of colors not used in K. Observe that, in our example of $(\Delta + 1 - r)$-cliques with r uncolored vertices, the clique palette contains $2r$ colors. As such, every vertex has r colors available in $L(K)$ because it has at most r external neighbors. So if each vertex samples one color from $L(K)$ and check if it is available, we expect at least $|Q_K|/2 \geqslant \Omega(\Delta/r)$ of them find a valid replacement color.

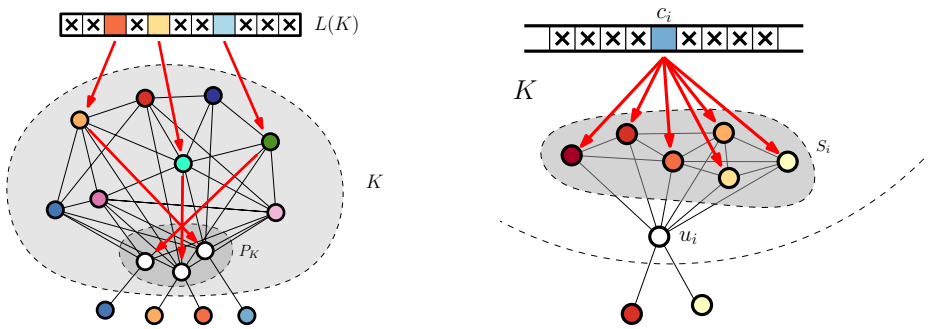

Figure 5. On the left, the 3-way matching from replacement colors to colored vertices to put-aside vertices. Note that a put-aside vertex cannot be matched to some colored vertices because their colors are used by neighbors outside the cabal. On the right, the intermediate set S_i of donors with the same replacement c_i. Observe that all donors from S_i are colored with shades of red: their colors come from the same block. To describe colors used in S_i to external neighbors, u_i sends a message containing the first color or the block — here, for instance, the color red — and for each color the offset from that color.

Step 3: Finding Safe Donors. Surprisingly perhaps, we match the uncolored vertices to replacements colors directly instead of matching them to donors. In other words, each u_i decides on a different c_i and will accept only donations from vertices with c_i as their replacement color. We proceed this way to eliminate recoloring conflicts within K, as by construction, donors for different u_i use different replacements.

For the final step, u_i needs to represent the colors of its donors succinctly. So it will restrict itself further to only a subset S_i of safe donors with similar colors to ensure they can be compressed into small messages. More precisely, we split the color space into contiguous *blocks* $B_1, B_2, \ldots, B_{\frac{\Delta+1}{b}}$ of $b = \text{poly}(\log n)$ colors each. Once vertex u_i settles on a block B_{j_i}, it only accepts donations from vertices with colors in that block. By choosing the block size large enough, each vertex u_i can find a block containing a large set S_i of vertices with colors from B_{j_i} (possibly, and safely, sharing it with other vertices).

Step 4: Donating Colors. In the last step, we match uncolored vertices to donors. The only criteria remaining for matching u_i to a donor $v \in S_i$ is that $\varphi(v)$ is not used by an external neighbor of u_i. By choosing $|S_i|$ large enough compared to r, the external degree of u_i, at least one of $k = \Theta(\frac{\log n}{\log \log n})$ random donations in S_i will be feasible with high probability.

To enable u_i to test all k donations using $O(\log n)$ bandwidth, we use that all nodes in the donor set S_i use colors from the *same block*. Hence k donations from S_i can be described by concatenating the index of the block and the offsets of the donations within the block. Overall, this scheme uses $O(\log \Delta + k \log b) = O(\log n)$ bits, for our choice of k and $b \leqslant \text{poly}(\log n)$ (see Eq (12.7)).

Roadmap of Part III. We begin with our negative result in Chapter 10. Then, in Chapter 11, we introduce general techniques for virtual graphs. This includes aggregation primitives, adaption of color trials based techniques from earlier part of this thesis and, most importantly, the fingerpringing technique to approximate degrees and other similar quantities.

Chapter 12 contains the the proof of Theorem 9.6. In particular, it contains the core definitions and technical results from this part: the adaption of Chapter 4 in Sections 12.1 to 12.3, the fingerpringing technique for finding anti-edges in Section 12.4, the coloring of put-aside sets in Section 12.5 and the prepation of SlackColor in Section 12.6.

Chapter 13 contains the adaption of Chapter 12 to low-degree graphs. The adaption of the Ghaffari-Kuhn algorithm can be found on Section 13.3.

CHAPTER 10

Bottlenecks in Virtual Graphs

In this chapter, we prove the following theorem involving the congestion, dilation, and bandwidth:

THEOREM 9.4. *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 claim little novelty here, and provide these lower bounds chiefly to paint a fuller picture of the relevance of our algorithms.

The dilation lower bound is straightforward, following directly from the seminal $\Omega(\log^* n)$ lower bounds on 3-coloring [Lin92; NS95]. We refer readers to Section 10.2.

As the congestion lower bound makes lengthy use of technical tooling from communication complexity literature largely unrelated to the rest of the thesis, we give here the high-level argument 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., $\Delta+1$-coloring the virtual graph solves the $\mathsf{c}/8$ instances.
- We observe: any T-round algorithm for $\Delta+1$-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 $\Delta+1$-coloring must necessarily be at least $\Omega(\mathsf{c}/\mathsf{b})$.

The idea to study information complexity through zero-communication protocols has been in the literature for a while now [KLLRX15], and recently, lower bounds were proved in a different model[1] using essentially the same arguments as our congestion lower bound [KRZ21, Theorem 1]. The type of argument deployed in the dilation lower bound has also appeared multiple times in the past, e.g., to connect the complexity of computing ruling sets to that of computing maximal independent sets.

10.1. Lower-bound with Respect to the Congestion

We prove the congestion-related part of our lower bound through a reduction from communication complexity.

The Communication Complexity Gadget. We define the communication complexity task we use in Definition 10.1. 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 1 for a illustration of our gadget.

Definition 10.1 (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

[1] The model consists of k machines with a shared blackboard, that receive an input graph whose description is randomly distributed among them.

– 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 ith left node is connected to the ith 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.

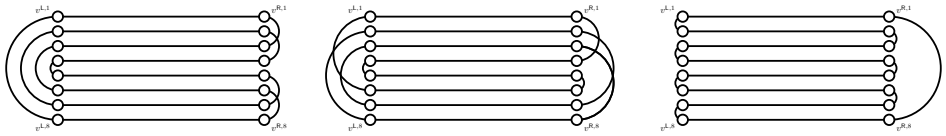

Figure 1. Three possible inputs to the communication complexity task.

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 [2] .

Lemma 10.2. *Any zero communication protocol for* $\mathsf{M3COL}_4$ *fails with probability at least* $\frac{1}{196}$ *over the uniform input distribution.*

Proof. Let us introduce notation first. Alice has a set of $2k$ nodes $\{v^{\mathsf{L},1}, \ldots, v^{\mathsf{L},2k}\}$, and Bob has a set of nodes $\{v^{\mathsf{R},1}, \ldots, v^{\mathsf{R},2k}\}$, such that for each i there is an edge between $v^{\mathsf{L},i}$ and $v^{\mathsf{R},i}$. Alice and Bob each receive as input a perfect matching over their nodes, and must output a 3-coloring of their nodes such that the overall coloring is proper. We denote by X and

[2] 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.

Y the sets of input of Alice and Bob, i.e, each element $x \in X$ represents a unique perfect matching over the nodes $v^{\mathsf{L},1}, \dots, v^{\mathsf{L},2k}$.

Consider any deterministic protocol for this task – since the error is measured w.r.t. the input distribution, for any randomized protocol achieving error ε, there exists a deterministic protocol achieving the same (or lower) error. We show that there necessarily is an index $i \in [8]$ such that both Alice and Bob do not always output the same color on their ith node. Since there are only 3 colors to choose from, this means that the players necessarily output the same color at this index on some pairs of inputs, and thus fail to properly color the graph.

Without loss of generality, we can assume that Alice and Bob always assign colors s.t. the coloring is at least valid w.r.t the matchings they received, and the error only comes from the edges $v^{\mathsf{L},i}v^{\mathsf{R},i}$ for each $i \in [8]$. Indeed, for any protocol without this property, a protocol with the property that has at most the same error probability can always be constructed.

Let us denote by $p_{\mathsf{A}}(i,c)$ the probability (over her random input X) that Alice outputs c at a given index i. For any input $x \in X$ of Alice, let $p_{\mathsf{A}}(i,c \mid x)$ be this same probability conditioned on Alice's input being x. Note that since the protocol is deterministic, $p_{\mathsf{A}}(i,c \mid x) \in \{0,1\}$. We have $p_{\mathsf{A}}(i,c) = \sum_{x\in X} p_{\mathsf{A}}(i,c \mid x)\mathbb{P}[X = x] = \frac{1}{|X|}\sum_{x\in X} p_{\mathsf{A}}(i,c \mid x)$. Let $p_{\mathsf{B}}(i,c)$ and $p_{\mathsf{B}}(i,c\,|\,y)$ be similarly defined from Bob's behavior in the protocol. For any pair of indices $i \neq j$, the vertices $v^{\mathsf{L},i}$ and $v^{\mathsf{L},j}$ are connected in $1/7$ of input matchings of Alice. Therefore, the probability that Alice outputs the same color on both indices is bounded as follows

$$\forall i,j \in \binom{[8]}{2}, \qquad \frac{1}{|X|}\sum_{x\in X}\sum_{c=1}^{3} p_{\mathsf{A}}(i,c \mid x)\cdot p_{\mathsf{A}}(j,c \mid x) \leqslant \frac{6}{7}\,. \tag{10.1}$$

Suppose there exists 4 distinct indices i s.t. $\max_{c\in[3]} p_{\mathsf{A}}(i,c) > \frac{13}{14}$. This would imply that there exist two indices $i \neq j$ and a color c s.t. $p_{\mathsf{A}}(i,c) > \frac{13}{14}$ and $p_{\mathsf{A}}(j,c) > \frac{13}{14}$. But then, for those two indices, we would have $\frac{1}{|X|}\sum_{x\in X}\sum_{c=1}^{3} p_{\mathsf{A}}(i,c \mid x)\cdot p_{\mathsf{A}}(j,c \mid x) > \frac{6}{7}$, contradicting Eq (10.1).

Consequently, there exists at least 5 indices s.t. $\max_{c\in[3]} p_{\mathsf{A}}(i,c) \leqslant \frac{13}{14}$. The same argument can be done on Bob's side, yielding that on at least 5 indices, we have $\max_{c\in[3]} p_{\mathsf{B}}(i,c) \leqslant \frac{13}{14}$. Since there are only 8 indices, there

are at least 2 indices for which both $\max_{c\in[3]} p_{\mathsf{A}}(i,c)$ and $\max_{c\in[3]} p_{\mathsf{B}}(i,c)$ are bounded by $\frac{13}{14}$.

Consider such an index i. The probability of a color conflict at this index is $\sum_{c=1}^{3} p_{\mathsf{A}}(i,c)\cdot p_{\mathsf{B}}(i,c)$. For an inner product of positive terms like this, the sum is minimized if the terms p_{A} and p_{B} in reverse order with respect to each other, i.e., if $p_{\mathsf{A}}(i,c+1) \geqslant p_{\mathsf{A}}(i,c)$ and $p_{\mathsf{B}}(i,c+1) \leqslant p_{\mathsf{B}}(i,c)$ for each $c \in [2]$, and if the weights are as unequal as possible. In our setting, this means the value is minimized with $(p_{\mathsf{A}}(i,c))_{c\in[3]} = (0, \frac{1}{14}, \frac{13}{14})$ and $(p_{\mathsf{B}}(i,c))_{c\in[3]} = (\frac{13}{14}, \frac{1}{14}, 0)$, which gives a probability of error of at least $\frac{1}{196}$. ∎

Non-negligible information complexity. The study of interactive information complexity in communication complexity has intuitively been an enterprise of generalizing seminal results for non-interactive communication (notably Shannon's [Sha48]), and an attempt to understand the flow of information in interactive protocols. Information complexity intuitively captures the amount of information that is revealed about the inputs in an interactive protocol, either between players (internal information cost) or to the outside world (external information cost). For an introduction to information theory concepts and their relevance to the study of communication complexity protocols, we recommend [RY20, Chapter 6]. We list the important definitions and properties in Section 2.5.

Definition 10.3. Consider a protocol π accepting inputs from $\mathcal{X}\times\mathcal{Y}$. Let μ be the distribution of $\mathbf{X}$ and $\mathbf{Y}$ and call $\mathbf{\Pi} = \pi(\mathbf{X},\mathbf{Y})$. Then, the *internal information cost* of π over μ is

$$\mathsf{IC}^{\mathsf{int}}_{\mu}(\pi) \stackrel{\text{def}}{=} I(\mathbf{\Pi} : \mathbf{X} \mid \mathbf{Y}) + I(\mathbf{\Pi} : \mathbf{Y} \mid \mathbf{X}) \ ,$$

and the *external information cost* of π over μ is

$$\mathsf{IC}^{\mathsf{ext}}_{\mu}(\pi) \stackrel{\text{def}}{=} I(\mathbf{\Pi} : \mathbf{XY}) \ .$$

Proposition 10.4 ([Bra17])**.** *For a protocol π and distribution of input μ,*

- $\mathsf{IC}^{\mathsf{int}}_{\mu}(\pi) \leqslant \mathsf{IC}^{\mathsf{ext}}_{\mu}(\pi)$.
- *When μ is a product distribution,* $\mathsf{IC}^{\mathsf{int}}_{\mu}(\pi) = \mathsf{IC}^{\mathsf{ext}}_{\mu}(\pi)$.
- *Information cost lower bounds communication cost, i.e., the maximum number of bits that can be sent in an execution of π.*

Definition 10.5. For a communication complexity task T, its *internal information complexity* $\mathsf{IC}^{\mathsf{int}}(T)$ is the infimum taken over all internal information costs of protocols performing T. Its *external information complexity* $\mathsf{IC}^{\mathsf{ext}}(T)$ is the infimum taken over all external information costs of protocols performing T.

When the task can be performed with error ε over an input distribution μ, the quantities $\mathsf{IC}^{\mathsf{int}}_{\mu}(T,\varepsilon)$ and $\mathsf{IC}^{\mathsf{ext}}_{\mu}(T,\varepsilon)$ are infimums taken over the internal/external information costs of protocols performing T with error at most ε on input distribution μ.

Note that as internal and external information costs coincide when inputs taken from a product distribution, internal and external information complexities also coincide in that situation. As a result, we omit the superscripts int and ext and simply write $\mathsf{IC}_{\mu}(T,\varepsilon)$ when μ is a product distribution.

Lemma 10.6. *Let T be some communication complexity task for some product distribution of input $\mu = \mu_X \times \mu_Y$, and $\varepsilon, \rho > 0$ be s.t. $\mathsf{IC}_{\mu}(T,\varepsilon) < \rho$. Then, there exists a zero communication protocol for performing T with error at most $\varepsilon + \sqrt{\frac{\ln(2)}{8} \cdot \rho}$ over μ.*

Proof. There exists a protocol π s.t. $\mathsf{IC}_{\mu}(\pi) \leqslant \rho$. From the definition of external information cost, there is little mutual information between the inputs $\mathbf{XY}$ and the protocol's transcript Π. Consider the following protocol: Alice and Bob sample random inputs from public randomness $\mathbf{X}'$ and $\mathbf{Y}'$, completely independent from their actual inputs $\mathbf{XY}$. Then, using again public randomness to sample random bits as needed, Alice and Bob locally execute π over the sampled inputs $\mathbf{X}'$ and $\mathbf{Y}'$ without any communication, and use the transcript of this protocol to decide on their output for their communication complexity task T. Let Π' be the distribution of transcripts from the execution of π without any communication using inputs and private randomness sampled from public randomness. Π' has the same distribution as Π, and from Corollary 2.15 (Pinsker's inequality), we have

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathbf{XY}}\left[\left\|p_{\Pi} - p_{\Pi|\mathbf{XY}=(x,y)}\right\|_1\right] \leqslant \sqrt{\frac{\ln(2)}{2} \cdot I(\Pi : \mathbf{XY})}\ .$$

For each input (x, y), let $S_{x,y}$ be the subset of transcripts that lead to correctly performing T. From the definition of total variation distance, the probability of hitting this subset of the transcripts is lowered by at most $\frac{1}{2}\left\|p_{\Pi} - p_{\Pi|\mathbf{XY}=(x,y)}\right\|_1$ when using the locally computed Π' as transcript instead of running the protocol on the actual input (x, y). Hence, when sampling inputs over μ, the error probability of this new protocol is $\varepsilon + \sqrt{\frac{\ln(2)}{8} \cdot \rho}$. ∎

Corollary 10.7. *Computing* $\mathsf{M3COL}_4$ *with error at most* $1/1000$ *over the uniform distribution has information complexity* $\mathsf{IC}^{\mathsf{int}}_{\mu}(\mathsf{M3COL}_4, 1/1000) > 0.0001942$.

Proof. From Lemma 10.2 and Lemma 10.6, we get that $\mathsf{IC}^{\mathsf{int}}_{\mu}(\mathsf{M3COL}_4, \varepsilon) \geqslant \frac{8}{\ln(2)}\left(\frac{1}{196} - \varepsilon\right)^2$. The result follows from plugging in $\varepsilon = 1/1000$. ∎

Increasing the size of the communication task. For a set of input distributions $\mathcal{M}$ and a communication complexity task T (possibly allowing for some error), define $\mathsf{IC}^{\mathsf{int}}(T, \mathcal{M})$ as the infimum of the information costs of all communication complexity protocols that solve task T on any distribution from the set $\mathcal{M}$.

Lemma 10.8 (Direct sum for internal information complexity [Bra17, Theorem 4.2])**.** *Let* T_1 *and* T_2 *be two tasks over input spaces* $\mathcal{X}_1 \times \mathcal{Y}_1$ *and* $\mathcal{X}_2 \times \mathcal{Y}_2$, $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *be two sets of distributions over* $\mathcal{X}_1 \times \mathcal{Y}_1$ *and* $\mathcal{X}_1 \times \mathcal{Y}_2$. *Let* $T = T_1 \times T_2$. *Then*

$$\mathsf{IC}^{\mathsf{int}}(T, \mathcal{M}_1 \times \mathcal{M}_2) = \mathsf{IC}^{\mathsf{int}}(T_1, \mathcal{M}_1) + \mathsf{IC}^{\mathsf{int}}(T_2, \mathcal{M}_2) \ .$$

Lemma 10.9. *Solving* k *independent instances of* $\mathsf{M3COL}_4$ *with error at most* $1/1000$ *for each instance has information complexity* $\Omega(k)$, *and a fortiori, communication complexity* $\Omega(k)$.

Proof. Let T be the task of solving an individual instance of $\mathsf{M3COL}_4$ with error at most $1/1000$. From Corollary 10.7, T has information complexity $\Omega(1)$ on inputs from the uniform distribution. From Lemma 10.8, solving k instances of $\mathsf{M3COL}_4$ given k inputs from the uniform distribution has information complexity $\Omega(k)$. As information complexity lower

bounds communication complexity, this gives the stated lower bound on communication complexity. ∎

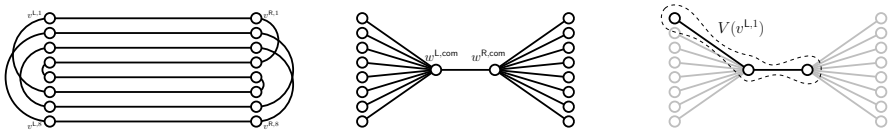

Figure 2. Examples of a virtual graph $H_{1,x,y}$ (left), a communication network $G_{1,x,y}$ (middle) in which it can be embedded, and the support of the top left virtual node (right).

Putting the communication problem into a virtual graph. We now map our communication complexity problem into our virtual graph framework. Intuitively, the graphs involved in our communication complexity gadget problem become our virtual graph, while the communication graph is simply a graph with the same perfect matchings on both sides with a single edge connecting all the nodes in the matching of Alice to all the nodes in the matching for Bob, resulting in all the congestion happening over this central edge. For any integer k, $x \in [105]^k$ and $y \in [105]^k$, consider the following virtual and communication graphs $H_{k,x,y}$ and $G_{k,x,y}$:

Virtual graph: $H_{k,x,y}$ contains $16k$ nodes and $16k$ edges. Each node is indexed by two numbers $(i,j) \in [k] \times [8]$ and is either in the left or the right part of the graph, i.e., we have nodes $v_1^{\mathsf{L},1}, \ldots, v_1^{\mathsf{L},8}, v_2^{\mathsf{L},1}, \ldots, v_k^{\mathsf{L},8}$ and $v_1^{\mathsf{R},1}, \ldots, v_1^{\mathsf{R},8}, v_2^{\mathsf{R},1}, \ldots, v_k^{\mathsf{R},8}$. For each $(i,j) \in [k] \times [8]$, there is an edge between $v_i^{\mathsf{L},j}$ and $v_i^{\mathsf{R},j}$. For each $i \in [k]$, $x_i \in [105]$ describes a perfect matching over the nodes $v_i^{\mathsf{L},1}, \ldots, v_i^{\mathsf{L},8}$, and similarly, $y_i \in [105]$ describes a perfect matching over the nodes $v_i^{\mathsf{R},1}, \ldots, v_i^{\mathsf{R},8}$ (recall that there are $\frac{(2x)!}{2^x \cdot x!}$ perfect matchings over a set of $2x$ nodes, which is 105 in the case of 8 nodes).

Communication graph: $G_{k,x,y}$ contains $16k + 2$ machines. Two central machines, $w^{\mathsf{L,com}}$ and $w^{\mathsf{R,com}}$, are each the root of a star with $8k$ leaves. The leaves are $(w_i^{\mathsf{L},j})_{i \in [k], j \in [8]}$ for the star rooted at $w^{\mathsf{L,com}}$, and $(w_i^{\mathsf{R},j})_{i \in [k], j \in [8]}$ for the other one. $w^{\mathsf{L,com}}$ and $w^{\mathsf{R,com}}$ are linked together.

Embedding: For each $(i,j) \in [k] \times [8]$, the virtual node $v_i^{\mathsf{L},j}$ has support $V(v_i^{\mathsf{L},j}) = \{w_i^{\mathsf{L},j}, w^{\mathsf{L},\mathsf{com}}, w^{\mathsf{R},\mathsf{com}}\}$, while $v_i^{\mathsf{R},j}$ has the smaller support $V(v_i^{\mathsf{R},j}) = \{w_i^{\mathsf{R},j}, w^{\mathsf{R},\mathsf{com}}\}$. Each edge of the form $v_i^{\mathsf{L},j} v_i^{\mathsf{R},j}$ is handled by $w^{\mathsf{R},\mathsf{com}}$. Edges between left nodes (resp., right nodes) are handled by $w^{\mathsf{L},\mathsf{com}}$ (resp., $w^{\mathsf{R},\mathsf{com}}$). That is, $w^{\mathsf{L},\mathsf{com}}$ and $w^{\mathsf{R},\mathsf{com}}$ each know one of the two perfect matchings.

For each $i \in [k]$, we refer to the virtual subgraph over the nodes $v_i^{\mathsf{L},1}, \ldots, v_i^{\mathsf{L},8}$ and $v_i^{\mathsf{R},1}, \ldots, v_i^{\mathsf{R},8}$ as the ith gadget. Figure 2 shows a virtual graph and a communication graph with $k = 1$, i.e., with a single gadget.

Lemma 10.10. *$\Delta+1$ coloring the virtual graph $H_{k,x,y}$ with $G_{k,x,y}$ as communication graph with error at most $1/1000$ requires $\Omega(\mathsf{c}/\mathsf{b}+1)$ communication rounds of bandwidth b.*

Proof. Any T-round distributed protocol for this problem immediately implies a $T \cdot \mathsf{b}$ communication complexity protocol for $\mathsf{M3COL}_4^{\otimes \mathsf{c}}$, which we have shown requires $\Omega(\mathsf{c})$ communication (Lemma 10.9), even with a less stringent error requirement. With the other constraint that 0 communication is insufficient (Lemma 10.2), i.e., at least 1 distributed round is needed, we get the $\Omega(\mathsf{c}/\mathsf{b}+1)$ lower bound. ∎

10.2. Lower-bound with Respect to the Dilation

We obtain the dilation component of our lower bound as an easy consequence of the seminal $\Omega(\log^* n)$ lower bound for 3-coloring paths by Linial [Lin92], and its extension to randomized algorithms by Naor [NS95].

Lemma 10.11. *A $T(n)$-round algorithm for $\Delta+1$-coloring or $(\widehat{\deg}+1)$-coloring virtual graphs of dilation d implies a $O(T(n)/\mathsf{d})$-round algorithm for 3-coloring paths.*

Corollary 10.12 (Lemma 10.11 + $\Omega(\log^* n)$ lower bound for 3-coloring paths [Lin92; NS95])**.** *Any algorithm for $(\Delta+1)$-coloring or $(\widehat{\deg}+1)$-coloring virtual graphs of dilation d has round complexity at least $\Omega(\mathsf{d} \cdot \log^* n)$, both for deterministic and randomized algorithms.*

Proof. Let $\mathscr{A}$ be the $T(n)$-round algorithm for $\Delta+1$-coloring (or $\widehat{\deg}+1$-coloring) virtual graphs of dilation d. Let $P = u_1, \ldots, u_n$ be the

path that we will 3-color using $\mathscr{A}$, and let its nodes have infinite bandwidth, i.e., we are in the LOCAL model. The path is taken as the virtual graph H in the context of our $\Delta + 1$-coloring algorithm, and the nodes in the path simulate $\mathscr{A}$ on an imaginary communication graph G. G is constructed as follow: each node u_i in the path P is replaced by a path of $2\mathsf{d}+1$ machines (of edge-length $2\mathsf{d}$), that acts as the support tree $T(u_i)$ of u_i, with the root in the middle of the path. The support trees are made to overlap at their extremities, i.e., one of the 2 leaves of $T(u_2)$ is also a leaf of $T(u_1)$, and the other is also a leaf of $T(u_3)$. The space of IDs of G is slightly increased from that of P (by a factor $\Theta(\mathsf{d})$) to give each machine a unique ID. Each u_i holds the information about the machines in $T(u_i)$.

After x rounds of communication over the original path P, each node in P has learned everything within distance x of itself. This means it knows everything within distance $x \cdot 2\mathsf{d}$ from any imaginary machine $v \in T(u_i)$, and in particular, it can simulate the full behavior of these machines in algorithm $\mathscr{A}$ for $x \cdot 2\mathsf{d}$ rounds. With $x \geqslant T(n)/(2\mathsf{d})$, u_i can simulate an entire run of $\mathscr{A}$ for the machine $v \in T(u_i)$, which means that the machines of the imaginary communication graph must reach a state in which they correctly $\Delta + 1$-colored their virtual graph. Since the virtual graph of the imaginary communication graph is the path we started with, in which the maximum degree is 2, this yields the claimed $O(T(n)/\mathsf{d})$ algorithm for 3-coloring a path. ∎

CHAPTER 11

General Tools on Virtual Graphs

In this chapter, we introduce several techniques to design algorithms on virtual graphs. We use then use them extensively in Chapter 12. Before we dive into the technical details, let us list informally the main results from this chapter.

Aggregation on Support Trees. As explained in Section 9.1, computing the exact degree of all vertices in parallel is prohibitively expensive. However, counting the degree of *one* vertex is easy with a simple breadth-first search on the communication network. This generalizes to multiple vertices when the corresponding BFS run in disjoint subgraphs (Lemma 11.1). This tree on the communication network can then be used to avoid double counting: for instance, we use this to count the exact size of each almost-clique as well as aggregate the average pseudo-external degree over each almost-clique. A slightly more complicated type of aggregation is prefix sums (Lemma 11.2). It allows us, for instance, to number vertices of an almost-clique K with unique indices in $\{1, 2, \dots, |K|\}$. More importantly, prefix sums and random groups allow dense vertices to learn in $O(\mathsf{cd})$ rounds which is the i-th used/unused color on their almost-clique — we henceforth refer to this as to *querying the clique palette* (Lemma 11.3).

Random Color Trials. As we have seen in earlier parts of this thesis, random color trials are the backbone of fast distributed coloring algorithms. They take multiple forms: a simple random color trial to generate slack (Section 3.3), several trials to compute a large enough colorful matching (Section 4.6), a synchronized color trial in almost-cliques (Section 3.5.1) or multi-color trials to take advantage of slack (Section 3.4). A key difference between color trials in virtual graphs compared to LOCAL or even

BCONGEST is that vertices do not know their palettes. So even the simple $O(\log n)$ round algorithm consisting of trying one random color from the palette until every vertex is colored cannot be implemented on virtual graphs.

We show that the random color trials performed by the broadcast congest algorithm (from Chapter 4) can be adapted to virtual graphs. Intuitively, this works because vertices try colors from "easily accessible sets" like $[\Delta+1]$, a set of reserved colors, or the clique palette. Since the broadcast congest algorithm requires knowledge of quantities like the external- and anti-degrees, implementing this algorithm leads to technical complications in Chapter 12.

Approximate Counting. To choose parameters for our color trials, we must be able to approximate external degrees accurately for all dense vertices. This is infeasible with BFS so we use a sketching technique we refer to as *fingerprinting*. Roughly speaking, fingerprinting allows us to estimate in constant rounds and up to a small constant error factor the number of neighbors with a certain property for each vertex (Lemma 11.17), e.g., estimate external degrees. The technique is based on properties of geometric variables and similar to cardinality-estimation in streaming algorithm.

We use this technique is an unconventional way in Section 12.4, which is why we study properties of fingerprints unrelated to approximate counting (Lemmas 11.13 and 11.14).

11.1. Aggregation on Virtual Graphs

On virtual graphs, a breadth-first search tree on H, the conflict graph, which induces a tree in G, the communication graph, allows for concise aggregation. Indeed, recall that two vertices in H can be adjacent through multiple paths in the communication graph G, so aggregation over all paths in G leads to double counting. In contrast, aggregation on a tree $T_G \subseteq E_G$ ensures each vertex contributes exactly once to the aggregation. Basic flooding construct such a tree. We emphasize, however, that running a BFS from multiple vertices at the same time could create congestion, and we only perform parallel BFS in vertex-disjoint subgraphs.

Lemma 11.1. *Let $t \geqslant 1$. Let $H_1, \ldots, H_k \subseteq H$ be a collection of vertex-disjoint subgraphs of H locally known to machines, each H_i containing a single source node s_i. In $O(\mathsf{cd} \cdot t)$ rounds of communication on G, we can simulate a t-hop BFS in each H_i with source vertex s_i in parallel for each $i \in [k]$. Let $T_{H,i} \subseteq E_{H_i}$ be the resulting BFS tree in H_i. Each $T_{H,i}$ induces a tree $T_{G,i} \subseteq E_G$ on the communication graph G such that each machine knows the edge leading to its parent in $T_{G,i}$. Moreover, $T_{G,i}$ has height (at most) dt and $T_{G,i} \subseteq \bigcup_{u \in V(H_i)} T(u)$. Every link is contained in at most c tree $T_{H,i}$.*

Proof. We define timestamps τ_v, initially $+\infty$ for all vertices except for source vertices: $\tau_{s_i} = 0$ for each $i \in [k]$. The algorithm repeats for t iteration: each vertex $u \in V_{H_i}$ such that $\tau_u < +\infty$ broadcasts $(\mathsf{ID}(s_i), \mathsf{ID}(u), \tau_u + 1)$. When a vertex $v \in V_{H_i}$ receives a message $(\mathsf{ID}(s_i), \mathsf{ID}(u), \tau_u)$ where $\tau_u + 1 < \tau_v$ from a neighbor $u \in V_{H_i}$, it sets τ_v to $\tau_u + 1$, making the node with identifier $\mathsf{ID}(u)$ its parent in the tree $T_{H,i}$. Timestamps are introduced to handle the delays incurred by communicating in G.

We now detail the implementation on G. For simplicity, we describe the algorithm for $\mathsf{c} = 1$. When $\mathsf{c} > 1$, we can replace every round in the following description by c rounds, one for each support tree links might belong to. Focus on some $v \in H_i$ and suppose a machine in $w \in T(v)$ receives a message of the form $(\mathsf{ID}(s_i), \mathsf{ID}(u), \tau_u)$ from a machine $w' \in T(u)$ such that $u \in H_i$ and w is responsible for an edge $\{u, v\} \in E_H$ (recall the definition on embedded virtual graphs, Definition 9.2). We say that vertex u was the emitter and w the receiver. Multiple emitters can reach different machines in $T(v)$ at concurrent times. Each receiver $w \in T(v)$ crafts a message $m_w = (RECEIVE, \tau_u, \mathsf{ID}(u), \mathsf{ID}(w))$ where $u \in H_i$ is the emitter. We aggregate on $T(v)$ the minimum of these messages according to the lexicographical order. Note that it suffices to learn the message with the minimum timestamp. Moreover, the aggregation produces a *unique receiver identifier* $\mathsf{ID}(w)$. When vertex v receives a message m_w with $\tau_u + 1 < \tau_v$, it updates $\tau_v = \tau_u + 1$ and broadcasts $(UPDATE, \tau_v, \mathsf{ID}(u), \mathsf{ID}(w))$ to all machines in $T(v)$. When machines of $T(v)$ receive this message, if they

belong to the support of a vertex $u \in V_{H_i} \setminus \{v\}$, they emit the message $(\mathsf{ID}(s_i), \mathsf{ID}(v), \tau_v)$ to the neighboring machine in $V(u)$.

Since support trees have diameter d, after $O(t'\mathsf{d})$ rounds in G for all $t' \leqslant t$, all vertices at distance $\leqslant t'$ from s_i in H_i identified a unique receiver in their support. We orient all edges of $T(v)$ toward that selected receiver. Taking all those directed trees together, we obtain the tree T_G. ∎

An *ordered tree* is a rooted tree on which each node knows an (arbitrary) ordering of its children. Ordering the children of every vertex leads to a total order of the vertices in the tree: a vertex is always ordered after its ancestors and, if u and v are two vertices of the tree with w as their lowest common ancestor, then we order u and v the same way w orders its two children on the wu- and wv-path.

Lemma 11.2 is obtained from a basic recursive algorithm on an ordered tree. It can be used, for instance, to give uncolored vertices of some set $S \subseteq V_H$ distinct identifiers in $\{1, 2, \ldots, |S|\}$ by having T be an arbitrary tree spanning S and setting $x_u = 1$ if u is uncolored and zero otherwise.

Lemma 11.2. *Let $T_1, \ldots, T_k$ be a collection of ordered trees of depth at most d on G and such that each link belongs to at most c of them. For each $i \in [k]$, let $S_i \subseteq V_{T_i}$ be a subset of its vertices such that each $u \in S_i$ holds some integer $|x_u| \leqslant \mathrm{poly}(n)$. There exists a $O(cd)$-round algorithm for every $u \in S_i$ to learn $\sum_{w \in S_i : w \prec u} x_w$, where $\prec$ denotes the ordering on S_i induced by T_i.*

Proof. We describe a $O(d)$-round algorithm within a single tree T, hence a single set $S \subseteq V_T$. Given that each link belongs to at most c trees, running the algorithm in parallel across all trees with pipelining ends in $O(cd)$ rounds.

Let r be the root of T. By convergecast, in $O(d)$ rounds, each machine computes the sum of the x_u over machines u in its subtree. In particular, the root knows the sum $\sum_{u \in S} x_u$, which it can broadcast to all machines in $O(d)$ rounds. To compute partial sums, we repeat the following inductive process starting with $w = r$. Let $w \in V_T$. If its subtree contains no machines of S, it has nothing to do. Otherwise, let $u_1 \preceq v_1 \prec u_2 \preceq v_2 \prec \ldots \prec u_k \preceq v_k$ be machines of S such that all $u \in [u_i, v_i]$ belong to the i-th subtree of w.

Call w_i the i-th children of w. By induction on the height of T, suppose that w knows the sum $S_{\prec w} \stackrel{\text{def}}{=} \sum_{u \prec w} x_u$. For the root, this sum is zero because r is the minimum vertex for the order induced by T. Since, during the converge cast, w received each sum $S_{[u_i,v_i]} \stackrel{\text{def}}{=} \sum_{u_i \preceq u \preceq v_i} x_u$, it can send to w_i (its i-th children), $S_{\prec w_i} = S_{\prec w} + \sum_{j<i} S_{[u_j,v_j]} = \sum_{u \prec u_i} x_u$, which concludes the proof. ∎

Query to the Clique Palette. Recall that the clique palette of some almost-clique K is the set of colors not used in K, i.e., $L_\varphi(K) = [\Delta+1] \setminus \varphi(K)$. Contrary to broadcast congest (Section 4.5.1), in virtual graphs, vertices cannot learn the clique palette. Fortunately, they do not need to: it suffices that they can *query* the clique palette. Vertices can either ask about the number number of used/unused colors in a given range $[a_v, b_v]$, or ask for the i-th used/unused color in such a range.

Lemma 11.3. *Suppose $\Delta \gg \log n$. Let φ be any (possibly partial) coloring of almost-clique K and let $\mathscr{C}(v) \in \{\varphi(K), L_\varphi(K)\}$ for every $v \in K$. If each $v \in K$ knows two integers $1 \leqslant a_v \leqslant b_v \leqslant \Delta+1$, then, w.h.p., they can either*

(1) learn $|\mathscr{C}(v) \cap [a_v, b_v]|$; or
(2) if it has $i_v \in [a_v, b_v]$, learn the i_v-th color in $\mathscr{C}(v) \cap [a_v, b_v]$.

The algorithm runs in $O(\mathsf{cd})$ rounds.

Proof. Let $k \stackrel{\text{def}}{=} \left\lceil \frac{\Delta+1}{C \log n} \right\rceil$ where C is some large universal constant. Partition $[\Delta+1]$ into k contiguous ranges $R_i = \{(i-1) \cdot C \log n + 1, \dots, i \cdot C \log n\}$ of colors for $i \in [k]$. Split K into k random groups $X_1, X_2, \dots, X_k$. The i-th group computes the set $R_i \cap \varphi(K)$ by aggregating a bit-wise OR. It takes $O(\mathsf{cd})$ rounds because each X_i has diameter two in H and $N_H(X_i) \supseteq K$ (Lemma 4.9). By taking the complement, vertices of X_i also compute $R_i \setminus \varphi(K) = R_i \cap L_\varphi(K)$, i.e., the free colors in range R_i.

Fix $\mathscr{C} \in \{\varphi(K), L_\varphi(K)\}$. The algorithm works the same for both. To comply with all vertices in K, we run the algorithm once for each value.

First, vertices of X_i learn $S_i \stackrel{\text{def}}{=} \sum_{j<i} |R_j \cap \mathscr{C}|$ as follow. Choose an arbitrary vertex $w \in K$ and run a BFS of depth one in K. It returns a $O(\mathsf{d})$-depth tree $T \subseteq E_G$ connected to at least one machine in each $V(v)$ for

$v \in N_H(w)$. Order neighbors of w in H as $v_1, v_2, \ldots, v_{|N_H(w) \cap K|}$ by using the prefix sum algorithm on T (Lemma 11.2). Each vertex knows its index in the ordering. Since v_i has a neighbor in X_i, it learns $|R_i \cap \mathscr{C}|$ in $O(\mathsf{cd})$ rounds. Using the prefix sum algorithm on T again, each v_i learns S_i. It then broadcasts (i, S_i) to its neighbors, one of which belongs to X_i. Hence, after $O(\mathsf{cd})$ rounds, all machines in $V(X_i)$ know S_i.

Nodes of X_i broadcast the following $O(\log n)$-bit message: $(i, S_i, R_i \cap \mathscr{C})$, where the last part of the message is a *set* of $|R_i| \leqslant O(\log n)$ colors represented as a $O(\log n)$-bitmap.

To learn $|\mathscr{C} \cap [a_v, b_v]|$, vertex v selects exactly one machine in $w_v^a \in V(v) \cap V(X_i)$ and exactly one in $w_v^b \in V(v) \cap V(X_j)$ where $a_v \in R_i$ and $b_v \in R_t$. Because machine w_v^a knows $a_v \in R_i$ and the message shared by X_i, it computes the number of colors strictly smaller than a_v in $\mathscr{C}$ as $S_v^a = S_i + |\{\chi \in R_i \cap \mathscr{C} : \chi < a_v\}|$. Similarly, machine w_v^b computes the number of colors smaller or equal to b_v as $S_v^b = S_t + |\{\chi \in R_t \cap \mathscr{C} : \chi < b_v\}|$. As those are two $O(\log n)$-bit integers, they can be disseminated in $V(v)$ in $O(\mathsf{CONGESTd})$ rounds. Finally, all machines in $V(v)$ know $|\mathscr{C} \cap [a_v, b_v]| = S_v^b - S_v^a$.

To learn the i_v-th color in $[a_v, b_v]$, first we broadcast S_v^a to all machines in $V(v)$. The i_v-th color of $[a_v, b_v]$ is the $(S_v^a + i_v)$-th color of $\mathscr{C}$. Suppose it belongs to R_j. Then machines of X_j detect from i_v, S_j and S_v^a that their group is responsible for the color v is asking about. And so a machine in $V(v) \cap V(X_j)$ can reply to v's query with the name of the color. ∎

Remark 11.4. If all machines have a scheme to compress colors, we can use Lemma 11.3 to learn multiple colors at a time. Indeed, the only moment where the $O(\log n)$-bit description of the colors matters is when they are broadcasted within the cluster at the very end. In general, if nodes have a scheme to encode colors using b bits, they can query up to $O(\log n/b)$ colors in the clique palette in $O(\mathsf{cd})$ rounds. We emphasize the vertex only learns the encoded colors (e.g., the hashes).

11.2. Random Color Trials on Virtual Graphs

We use that trying random colors from the palette decreases degrees by a constant factor (Lemma 4.33). Since vertices do not have access to

their palettes, this does not translate directly to virtual graphs. However, it suffices that vertices can sample one uniform color in a $\mathrm{List}(v)$ with enough available colors. This can be, for instance, $[\Delta + 1]$, the clique palette, or the set of reserved colors.

ALGORITHM 26. TRYCOLOR in virtual graphs

Input: a virtual graph H, lists List, a partial coloring φ, a constant γ.
Output: an extension of φ

Each uncolored vertex does:

(1) Get activated with probability $p = \gamma/4$.
    Let $\mathbf{A}$ be the set of activated vertices.
(2) Sample $\boldsymbol{\chi}(v) \in \mathrm{List}(v)$ uniformly at random
(3) If $\boldsymbol{\chi}(v) \in \mathrm{List}_\varphi(v)$ and $\boldsymbol{\chi}(v) \notin \boldsymbol{\chi}(\mathbf{A} \cap N_<(v))$, then extend φ at v with $\boldsymbol{\chi}$. Otherwise, v remains uncolored.

Lemma 11.5. *Let $c \gg 1$ and $\gamma \in (0, 1)$ be universal constants known to all nodes. Let φ be a coloring, $S \subseteq V \setminus \operatorname{dom}\varphi$ a set of uncolored nodes, and $\mathrm{List}(v)$ a set of colors for each $v \in S$ such that*

(1) v can sample a uniform color in $\mathrm{List}(v)$ in $O(\mathsf{cd})$ rounds,
(2) $|\mathrm{List}(v)| \geqslant \gamma^{-2} c \log n$ for some large constant c, and
(3) $|\mathrm{List}_\varphi(v)| \geqslant \gamma \max\{|\mathrm{List}(v)|, \deg_\varphi(v)\}$.

Let ψ be the partial coloring produced by having each vertex of S run TRYCOLOR. *Then, w.h.p., each $w \in V_H$ has uncolored degree in S*

$$\deg_\psi(w; S) \leqslant \max\big\{(1 - \gamma^2/32)\deg_\varphi(w; S),\ \gamma^{-2} c \log n\big\}\ .$$

The algorithm runs in $O(\mathsf{cd})$ rounds.

PROOF. Let $\mathcal{E}$ be the event that all vertices with $\gamma^{-2}(c/2)\log n$ neighbors in S have between $(\gamma/8)\deg(w, S)$ and $(\gamma/2)\deg(w, S)$ active neighbors in S. Each vertex gets activated independently with probability $p = \gamma/4$, so by the classic Chernoff bound and union bound, $\mathcal{E}$ holds with high probability. We implicitly condition on $\mathcal{E}$ henceforth.

Consider some w with $\deg_\varphi(w, S) \geqslant \gamma^{-2} c \log n$, for otherwise the claim already holds. Call d its active degree in S and $u_1, \dots, u_d$ its active neighbors in S ordered by increasing identifier. From $\mathcal{E}$, we know that $d \geqslant (\gamma/8) \deg_\varphi(w, S)$, so let us prove that each u_i gets colored with constant probability (over the randomness of the sampled colors). Define $\mathbf{X}_i$ as the indicator random variable of the event that vertex u_i gets colored. Note that to get colored, vertex u_i must sample a color from $\text{List}_\varphi(u_i)$ and a color that is not sampled by active neighbors of smaller ID. The former occurs with probability at least γ by assumption (3). Conditioning on $\boldsymbol{\chi}(u_i) \in \text{List}_\varphi(v)$, we get that $\boldsymbol{\chi}(u_i)$ is uniformly distributed in $\text{List}_\varphi(u_i)$. If u_i has $\deg(u_i, S) \leqslant \gamma^{-2}(c/2) \log n$ neighbors in S (so $\mathcal{E}$ does not apply to u_i), it gets colored with probability at least $1/2$ since

$$|\text{List}_\varphi(u_i)| \geqslant \gamma |\text{List}(u_i)| \geqslant \gamma^{-1} c \log n \geqslant 2 \deg(u_i, S) \ ,$$

where the first inequality uses assumption (3) and the second one assumption (2). Otherwise, if $\deg(u_i, S) \geqslant \gamma^{-2}(c/2) \log n$, vertex u_i has at most $(\gamma/2) \deg(u_i, S)$ active neighbors, since $\mathcal{E}$ applies to u_i, hence the probability that its color conflicts with an activated neighbor is at most

$$\frac{(\gamma/2) \deg(u_i)}{|\text{List}_\varphi(u_i)|} \leqslant \gamma^2/2$$

from assumption (3). Also note that this upper bound on $\mathbf{X}_i = 1$ (with the conditioning on $\mathcal{E}$) holds for any conditioning on the colors of sampled by neighbors of lower ID. For all i, we thus have that

$$\mathbb{P}[\mathbf{X}_i = 1 \mid \mathcal{E}, \boldsymbol{\chi}(u_1), \dots, \boldsymbol{\chi}(u_{i-1})] \geqslant \gamma(1 - \gamma^2/2) \geqslant \gamma/2 \ .$$

As $\mathbf{X}_i$ depends only on the colors sampled by lower ID neighbors, we may apply the Chernoff Bound with stochastic domination (Eq (2.4)). We get that at least $(\gamma/4)d \geqslant (\gamma^2/32) \deg_\varphi(w, S)$ neighbors of w get colored with high probability, using the assumption on w's degree in S. By union bound, it holds for all such w in the graph. In particular, it means that the uncolored degree in S of every such w decreases by at least a factor $(1 - \gamma^2/32)$. ∎

One call to TryColor reduces the uncolored degrees only by a small constant factor. Nonetheless, a direct corollary of Lemma 11.5 is that repeated calls reduce uncolored degrees by any desirable constant factor.

We emphasize that 11.6 makes a stronger assumption on the slack of vertices than Lemma 11.5.

Corollary 11.6. *Let $\gamma, \varphi, \text{List}, S$ such that all vertices $v \in S$ have $|\text{List}(v)| \gg \gamma^{-2} \log n$ and*

$$|\text{List}_\varphi(v)| \geqslant \deg_\varphi(v) + \gamma |\text{List}(v)| \ .$$

For any $\delta \in (0,1)$, after $O\left(\frac{\ln(1/\delta)}{\gamma^2}\right)$ iterations of TryColor*, the uncolored degree of every vertex in S has decreased by a factor δ or is at most $O(\gamma^{-2} \log n)$.*

Proof. We argue that assumption for Lemma 11.5 hold before each call. Note that assumption (2) always holds as $\text{List}(v)$ never changes. Moreover, the slack in List does not decrease as we extend the coloring, so assumption (3) always hold. Hence, w.h.p., by Lemma 11.5, the uncolored degrees decrease by a factor $(1 - \gamma^2/32)$ each call, and after $T = \frac{32 \ln(1/\delta)}{\gamma^2}$ calls it decreases by a factor δ. ∎

Ultrafast Coloring. Observe that the version of MultiColorTrial used by the broadcast congest algorithm (in Section 4.3) works by broadcasting one $O(\log n)$-bit message and then aggregating a bit-wise OR over support trees. So it can be implemented directly on virtual graphs. Using Corollary 11.6 for the degree reduction of Phase (1) in SlackColor, we obtain the following variant of Proposition 3.19 for virtual graphs.

Proposition 11.7. *Let $\delta \in (0,1)$ be a real number and H be a virtual graph on network G with congestion c and diation d. Suppose every vertex has a list $\text{List}(v)$ and φ a partial coloring of H such that*

i) $\text{List}(v)$ is known to every machine in $V(v)$,
ii) $|\text{List}_\varphi(v)| \geqslant \deg_\varphi(v) + s(v)$ colors where $s(v) \geqslant \delta |\text{List}(v)|$.

If there exists $\kappa \in [1/n, 1]$ and $c > 0$ (both globally knwon) such that $s(v) \geqslant \max\{O(\delta^{-2} \log n), (2c \log n)^{1+\kappa}\}$ for every vertex, then there exists a $O(\mathsf{cd}/\delta^2(\log^* n + \log 1/\delta + 1/\kappa))$-round randomized algorithm called SlackColor *to* List*-list-colors virtual graphs with high probability.*

Proof. The lists List verify conditions of Corollary 11.6, so after $O(\mathsf{cd} \cdot \log(1/\delta)/\delta^2)$ rounds, w.h.p., uncolored vertices have are at most $(\delta/2)|\text{List}(v)| \leqslant$

$s(v)/2$ uncolored neighbors. By assumption on $s(v)$, this is also true for vertices of degree $O(\delta^{-2}\log n)$. As such, we can use the Algorithm 9 to run MULTICOLORTRIAL in Phases (2) and (3) of Algorithm 4. By Lemma 3.22, every vertex is colored w.h.p. after $O(\log^* n + 1/\kappa)$ calls to MULTICOLORTRIAL, each of which takes $O(\mathsf{cd}/\delta^2)$ rounds to implement on virtual graphs (for the same reason as for Proposition 4.5). ∎

Slice Color. For completeness, we mention a substitute to Proposition 11.7 where vertices need only to try one color per round. Proposition 11.8 assumes that vertices have access to an algorithm $\mathbf{C}_v$ for sampling roughly-uniform colors. The uniformity property is formalized by Eq (11.2) and essentially asks that v has slack $s(v) \geqslant \Omega(\deg_\varphi(v))$. The advantage of this method is that it is easier to setup than Proposition 11.7 — for instance, in [FHN23], it was used to color the square graph by sampling colors from the clique palette — but the downsides are that (1) it requires $O(\mathsf{cd}\cdot\log\log n)$ rounds and (2) it only reduces the problem to $O(\log\log n)$ low-degree instances.

Proposition 11.8 ([FHN23, Lemma 3.2])**.** *Let $C, \alpha, \kappa > 0$ be some universal constants. Suppose that H is a virtual graph with a partial coloring φ such that each node knows an upper bound $b(v) \geqslant \deg_\varphi(v)$ on its uncolored degree. Suppose that for all nodes with $b(v) \geqslant C\log n$, and a value $s(v) \geqslant \alpha\cdot b(v)$, there exists a T-round algorithm that samples a color $\mathbf{C}_v \in L_\varphi(v)\cup\{\perp\}$ (where $\perp$ represents failure) with the following properties:*

$$\mathbb{P}[\mathbf{C}_v = \perp] \leqslant 1/\mathrm{poly}(n)\ , \tag{11.1}$$

$$\mathbb{P}[\mathbf{C}_v = c \mid \mathbf{C}_v \neq \perp] \leqslant \frac{\kappa}{\deg_\varphi(v) + s(v)}\ . \tag{11.2}$$

Then, there is a $T\cdot O(\log\log\Delta + \kappa\cdot\log(\kappa/\alpha))$-rounds algorithm extending the current partial coloring so that uncolored vertices are partitioned into $\ell = O(\log\log\Delta)$ layers $L_1, \dots, L_\ell$ such that each uncolored node knows to which layer it belongs and each $H[L_i]$ has uncolored degree $O(\log n)$.

Since we will not need Proposition 11.8 in this manuscript, we skip the proof and refer readers to [FHN23, Section 5] for details.

Colorful Matching. Recall that K has a colorful matching of size k if at least k colors are repeated in K (Definition 4.18). In Section 4.6, we described a simple random-color trial based algorithm to compute a $\Omega(a(K)/\varepsilon)$-sized colorful matching when $a(K)$ is large enough. Algorithm 14 as well as its BCONGEST implementation (Lemma 4.22) extend naturally to virtual graphs.

Lemma 11.9. *Assume $\Delta \gg \log n$. Algorithm 14 can be implemented in $O(\mathsf{cd}/\varepsilon)$ rounds on virtual graphs. In particular, it extends the coloring from* SlackGeneration *using only colors in $(r, \Delta+1]$ and such that every almost-clique with $a(K) \geqslant \Omega(\log n)$ increases the size of its colorful matching by at least $\Omega(a(K)/\varepsilon)$ with high probability.*

We omit the proof since it is a straightforward adaptation of Lemma 4.22 to virtual graphs. Indeed, it uses only random groups and aggregation on trees, which works on virutal graphs thanks to Lemma 11.1.

Synchronized Color Trial. Recall that, in every almost-clique, Algorithm 5 samples a uniform permutation of the clique palette and distributes colors accordingly. Sampling a truly uniform permutation in virtual graphs is challenging. Instead, we sample a *pseudo-random permutation*, which only affects the success probability by a constant factor. The argument is akin to that of [New91] to reduce the number of public random bits in the two-party communication model to $O(\log n)$. It is also reminiscent of the representative sets construction from [HN23].

To implement Algorithm 5 in LOCAL, vertices learned the entire clique palette — and, in fact, also in BCONGEST, Section 4.5.1. This approach is not feasible in virtual graphs due to bandwidth constraints. It suffices, however, that each vertex learns the one color it has to try, which we implement using the query mechanism (Lemma 11.3).

Lemma 11.10. *Let $\alpha \in (0,1]$ be a globally known universal constant, φ be a partial coloring and, for each almost-clique K, let $S_K \subseteq K$ and $r_K \in \mathbb{N}_{\geqslant 0}$ be such that $|\mathscr{C}_K| \geqslant |S_K| \geqslant \alpha|K|$ where $\mathscr{C}_K = L_\varphi(K) \setminus [r_K]$. Algorithm 5 — where we permute $S_i = S_K$ and try colors $\mathscr{C}_i = \mathscr{C}_K$ in the i-th almost-clique K — can be implemented in parallel in all almost-cliques in $O(\mathsf{cd})$ rounds*

with high probability. The number of uncolored vertices in each K is as described in Lemma 3.25.

PROOF. For a given n, Δ, k, r, and α, the set $\mathscr{L}(n, \Delta, k, r, \alpha)$ is the set of all possible inputs for Algorithm 5: a tuple $(G, \varphi, K, S, \sigma)$ where G is graph on n vertices with maximum degree Δ, K is an almost-clique in G and φ is a partial coloring of G such that K has a subset S of k uncolored vertices, colors $\{1, \ldots, r\}$ are not used in K and $|\mathscr{C}_K| \geqslant \alpha|S|$ and σ is some total ordering of the vertices in S. We also include a $O(\log n)$-bit identifier and poly(n) random bits for all vertices. It is easy to see that the number of such configurations is at most $\exp(\text{poly}(n))$.

We argue that, for all fixed n, Δ, k, r, and α, there exists a set of poly(n) permutations $\mathscr{P} = \mathscr{P}(n, \Delta, k, r, \alpha)$ of $[k]$ such that, for $(G, \varphi, K, S, \sigma) \in \mathscr{L}(n, \Delta, k, r, \alpha)$, if vertices of S run Algorithm 5 for a uniform $\boldsymbol{\pi} \in \mathscr{P}$ while every uncolored vertex outside K tries at most one color (adversarially), with high probability, the number of uncolored vertices in K is as described by Lemma 3.25.

Before proving its existence, let us explain why it implies the lemma. An uncolored vertex in $S \subseteq K$ knows the number of uncolored vertices $|K \setminus \operatorname{dom} \varphi|$, the number of reserved colors r, and its rank in the ordering $\sigma(v)$ — using the prefix sum algorithm from Lemma 11.2. In particular it knows the set $\mathscr{P}(n, \Delta, k, r, \alpha)$ (e.g., by computing it locally in exponential time). One vertex of K can therefore sample $\boldsymbol{\pi} \in \mathscr{P}$ and broadcast it to the rest of the vertices in $O(\mathsf{cd})$ rounds. Since the family $\mathscr{P}$ has poly(n) size, it requires only $O(\log n)$ bits to represent it. Once $v \in S$ knows $\boldsymbol{\pi}(\sigma(v))$, it can learn the corresponding color using the query algorithm (since it knows r_K), thus run the color trial of Algorithm 5 in $O(\mathsf{cd})$ rounds. The bound claimed by Lemma 3.25 holds for all almost-cliques with high probability by union bound.

Construct $\mathscr{P}(n, \Delta, k, r, \alpha)$ by sampling independently t truly uniform permutations of $\{1, 2, \ldots, k\}$. We say permutation π is bad for a fixed configuration $(G, \varphi, K, S, \sigma) \in \mathscr{L}$ if the synchronized color trial using π fails on $(G, \varphi, K, S, \sigma)$, i.e., too many nodes are uncolored in S. By Lemma 3.25, a uniform permutation permutation is bad for $\mathscr{L}$ with probability at most n^{-c} for some constant $c \geqslant 1$. Hence, by Chernoff, w.e.h.p. in t/n^c, the

set $\mathscr{P}(n, \Delta, k, r, \alpha)$ contains at most $2t/n^c$ bad permutations for a fixed local configuration in $\mathscr{L}(n, \Delta, k, r, \alpha)$. By the probabilistic method, for $t = \text{poly}(n)$ a large enough polynomial, there exists a family $\mathscr{P}(n, \Delta, k, r, \alpha)$ of t permutations such that each local configuration has at most $2t/n^c$ bad permutations in $\mathscr{P}(n, \Delta, k, r, \alpha)$. ∎

11.3. Fingerprinting & Approximate Counting

The plan for this section is as follows: in Section 11.3.1, we analyze the behavior of maxima of geometric variables; in Section 11.3.2, we give an efficient encoding scheme for sets of maxima of geometric variables; in Section 11.3.3, we apply those results to derive an efficient approximate counting algorithm in cluster graphs.

11.3.1. Maxima of Geometric Random Variables. We begin with a few simple properties of geometrically distributed random variables. For any $\lambda \in (0, 1)$, we say $\mathbf{X}$ is a geometric random variable of parameter λ when

$$\text{for all } k \in \mathbb{N}_{\geqslant 0}, \quad \mathbb{P}[\mathbf{X} = k] = \lambda^k - \lambda^{k+1} \ .$$

Note that $\mathbb{P}[\mathbf{X} \geqslant k] = \lambda^k$. That is $\mathbf{X}$ counts the the number of trials needed until the first success when each trial is independent and *fails* with probability λ.

Claim 11.11. *Let d be an integer and $\mathbf{X}_1, \dots, \mathbf{X}_d$ be independent geometric random variables of parameter $1/2$. Let $\mathbf{Y} = \max_{i \in [d]} \mathbf{X}_i$. For all $k \in \mathbb{N}_{\geqslant 0}$, we have*

$$\mathbb{P}[\mathbf{Y} < k] = (1 - 2^{-k})^d \ .$$

Proof. The event $\{\mathbf{Y} < k\}$ is the intersection of the d events $\{\mathbf{X}_i < k\}$ with $i \in [d]$, which are independent and have probability $1 - 2^{-k}$. ∎

Standard analysis shows that the expected maximum over d geometric random variables is about $\log_{1/\lambda} d$. There has been work on asymptotic behavior of such variables (e.g., [Eis08; BO90]). Lemma 11.12 shows concentration of measure for maxima of independent geometric variables suited to our use. We will later use this phenomenon to approximate an unknown d from the aggregated maxima.

Lemma 11.12 (Concentration of values)**.** *Consider $t, d \geqslant 1$ integers and $t \times d$ independent geometric random variables $(\mathbf{X}_{i,j})_{i\in[t],j\in[d]}$ of parameter $1/2$. For each $i \in [t]$, let $\mathbf{Y}_i = \max_{j\in[d]} \mathbf{X}_{i,j}$. For each integer k, let $\mathbf{Z}_k = |\{i \in [t] : \mathbf{Y}_i < k\}|$. Let $\boldsymbol{k}^\star = \min\{k : \mathbf{Z}_k \geqslant (27/40)t\}$ and define*

$$\hat{d} \overset{def}{=} \frac{\ln(Z_{\boldsymbol{k}^\star}/t)}{\ln(1-2^{-\boldsymbol{k}^\star})} \ .$$

Then, for any $\xi \in (0,1)$,

$$|d - \hat{d}| \leqslant \xi d \quad \textit{w.p. at least} \quad 1 - 6\exp(-\xi^2 t/200) \ . \tag{11.3}$$

Proof. For each integer k, let $p_k \overset{\text{def}}{=} (1-2^{-k})^d$, such that $p_k = \mathbb{P}[\mathbf{Y}_i < k]$ for all $i \in [t]$, as shown in Claim 11.11. Let us analyze p_k when $k \approx \log d$. First, we bound p_k as

$$1 - d\cdot 2^{-k} \quad \leqslant \quad p_k \quad \leqslant \quad \exp(-d\cdot 2^{-k}) \ . \tag{11.4}$$

(Using that $1 - d\cdot x \leqslant (1-x)^d \leqslant e^{-d\cdot x}$ for all $x \in [0,1]$.)

By definition, the probabilities p_k are strictly increasing as a function of k. We also have that $\mathbf{Z}_{k+1} \geqslant \mathbf{Z}_k$ structurally, since $\{i : \mathbf{Y}_i < k\} \subseteq \{i : \mathbf{Y}_i < k+1\}$. Eq (11.4) implies that $p_k \geqslant 3/4$ for each $k \geqslant \log d + 2$, and $p_k \leqslant e^{-1/2} < 0.607$ for each $k \leqslant \log d + 1$. We argue that $\boldsymbol{k}^\star \in \{\lceil \log d\rceil + 1, \lceil \log d\rceil + 2\}$. For each $k > 0$, we have $\mathbb{E}[\mathbf{Z}_k] = p_k \cdot t$. By the additive Chernoff Bound (Eq (2.3)), we have

$$\mathbb{P}[|\mathbf{Z}_k - p_k t| > (\xi/20)t] \leqslant 2\exp\left(-\frac{\xi^2 t}{200}\right) \ .$$

Thus, by union bound, $|\mathbf{Z}_k - p_k t| \leqslant (\xi/20)t$ holds for all $k \in \{\lceil \log d\rceil, \lceil \log d\rceil + 1, \lceil \log d\rceil + 2\}$, w.p. at least $1 - 6\exp(-(\xi^2/200)t)$. In particular, we have $\mathbf{Z}_{\lceil \log d\rceil+2} \geqslant (3/4 - \xi/20)t > (27/40)t$ and thus $\boldsymbol{k}^\star \leqslant \lceil \log d\rceil + 2$. Additionally, it holds that $\boldsymbol{k}^\star \geqslant \lceil \log d\rceil + 1$ because $\mathbf{Z}_{\lceil \log d\rceil} \leqslant (e^{-1/2} + \xi/20)t < (27/40)t$.

To summarize, the selected $\boldsymbol{k}^\star$ belongs to $\{\lceil \log d\rceil + 1, \lceil \log d\rceil + 2\}$ and verifies that

$$|\mathbf{Z}_{\boldsymbol{k}^\star}/t - p_{\boldsymbol{k}^\star}| \leqslant (\xi/20) \ . \tag{11.5}$$

Eq (11.3) follows from the following computations. Plugging Eq (11.5) into the definition of $\hat{d}$, we obtain

$$\hat{d} \stackrel{\text{def}}{=} \frac{\ln(\mathbf{Z}_{\boldsymbol{k}^\star}/t)}{\ln(1-2^{-\boldsymbol{k}^\star})} \in \frac{\ln(p_{\boldsymbol{k}^\star} \pm \frac{\xi}{20})}{\ln(1-2^{-\boldsymbol{k}^\star})} = \frac{\ln(p_{\boldsymbol{k}^\star}) + \ln(1 \pm \frac{\xi}{20p_{\boldsymbol{k}^\star}})}{\ln(1-2^{-\boldsymbol{k}^\star})}$$
$$= d + \frac{\ln(1 \pm \frac{\xi}{20p_{\boldsymbol{k}^\star}})}{\ln(1-2^{-\boldsymbol{k}^\star})}$$

where the last equality follows from $p_{\boldsymbol{k}^\star} = \exp\big(d \cdot \ln(1-2^{-\boldsymbol{k}^\star})\big)$.

Recall that $x/(1+x) \leqslant \ln(1+x) \leqslant x$ for all $x \in (-1,1)$, and that the three functions are increasing over this interval. Let us bound the denominator of the error term first.

Since $\boldsymbol{k}^\star \leqslant \lceil \log d \rceil + 2$, we have that $2^{-\boldsymbol{k}^\star} \geqslant 1/(8d)$, and so

$$\ln\Big(1-2^{-\boldsymbol{k}^\star}\Big) \leqslant \ln\left(1-\frac{1}{8d}\right) \leqslant -\frac{1}{8d} \ .$$

Let us now bound the numerator, which is equal to $\ln(1+x)$ for some value $x \in [-\xi/(20p_{\boldsymbol{k}^\star}), \xi/(20p_{\boldsymbol{k}^\star})]$. Remark that $\xi/(20p_{\boldsymbol{k}^\star}) \leqslant \xi/10$ since $p_{\boldsymbol{k}^\star} \geqslant 1/2$ (by Eq (11.4)). As $\ln(1+x)$ is increasing over the interval $(-1,1)$ and the bound of $\ln(1+x) \leqslant x$, we get an upper bound for the numerator of

$$\ln\left(1+\frac{\xi}{10p_{\boldsymbol{k}^\star}}\right) \leqslant \xi/(20p_{\boldsymbol{k}^\star}) \leqslant \xi/10 \ .$$

On the other hand, that $\ln(1+x)$ is increasing over the interval $(-1,1)$ and the bound $\ln(1+x) \geqslant x/(1+x)$ lowers bound the numerator by

$$\ln\left(1-\frac{\xi}{20p_{\boldsymbol{k}^\star}}\right) \geqslant \ln\left(1-\frac{\xi}{10}\right) \geqslant -\frac{\xi}{10}\Big/\left(1-\frac{\xi}{10}\right) \geqslant -\frac{\xi}{9} \ .$$

Together, these inequalities yield Eq (11.3). ∎

In Section 12.4, we will use other properties of maxima of geometric variables, which we introduce now.

Lemma 11.13 (Unique maximum)**.** *Consider an integer $d \geqslant 2$ and d independent geometric random variables $(\mathbf{X}_j)_{j\in[d]}$ of parameter $\lambda < 1$. Let $\mathbf{Y} = \max_{j\in[d]} \mathbf{X}_j$. Then there exist $i \neq j \in [d]$ such that $\mathbf{X}_i = \mathbf{X}_j = \mathbf{Y}$ with probability at most $\frac{(1-\lambda)^2}{1-\lambda^2}$.*

PROOF. Call $\mathcal{E}$ the event that the maximum is not unique. Then, for each pair $i < j$, let

$$\begin{aligned}\mathscr{F}_{i,j} = \quad & \{\forall k \in [d], \mathbf{X}_i \geqslant \mathbf{X}_k\} \\ & \cap \{\forall k \in [d] \setminus \{i\}, \mathbf{X}_j \geqslant \mathbf{X}_k\} \\ & \cap \{\forall k \in [j-1] \setminus \{i\}, \mathbf{X}_j > \mathbf{X}_k\}\end{aligned}$$

be the event that the $\mathbf{X}_i$ is a maximum, $\mathbf{X}_j$ is a maximum in $\mathbf{X}_{\neg i}$ (if we ignore $\mathbf{X}_i$) and all $\mathbf{X}_{<j-1}$ (except $\mathbf{X}_i$) are *stricly less* than $\mathbf{X}_j$ (which is itself at most $\mathbf{X}_i$). Intuitively, when $\mathscr{F}_{i,j}$ occurs, $\mathbf{X}_i$ is the first maximum and $\mathbf{X}_j$ the second one when we order first according to $\mathbf{X}$'s and then to indices.

Note that $\mathscr{F}_{i,j}$ are pairwise disjoint events and that $\mathcal{E} \subseteq \bigcup_{i<j} \mathscr{F}_{i,j}$. We emphasize however that event $\mathscr{F}_{i,j}$ do not cover the whole universe (meaning the set $[d] \times \mathbb{N}$ of all outcomes). By disjoint union and bayes rule

$$\mathbb{P}[\mathcal{E}] = \sum_{m \geqslant 0} \sum_{i<j} \mathbb{P}[\mathcal{E}, \mathscr{F}_{i,j}, \mathbf{Y} = m] = \sum_{m \geqslant 0} \sum_{i<j} \mathbb{P}[\mathcal{E}, \mathbf{Y} = m \mid \mathscr{F}_{i,j}] \mathbb{P}[\mathscr{F}_{i,j}] \ .$$

We bound this conditional probability through the following observation. Since $\mathbf{X}_i$ and $\mathbf{X}_j$ are the largest values, the maximum is not unique iff both are equal to m. So,

$$\mathbb{P}[\mathcal{E}, Y = m \mid \mathscr{F}_{i,j}] \leqslant \lambda^{2m}(1-\lambda)^2 \ ,$$

and since $\sum_{i<j} \mathbb{P}[\mathscr{F}_{i,j}] \leqslant 1$, we get

$$\mathbb{P}[\mathcal{E}] = \sum_{m \geqslant 0} \lambda^{2m}(1-\lambda)^2 \left(\sum_{i<j} \mathbb{P}[\mathscr{F}_{i,j}] \right) \leqslant \sum_{m \geqslant 0} \lambda^{2m}(1-\lambda)^2 = \frac{(1-\lambda)^2}{1-\lambda^2} \ .$$

which concludes the proof. ∎

In particular, with a set of geometric random variables of parameter $1/2$, their maximum occurs uniquely with probability at least $2/3$, regardless of the number of random variables. Also, note that the distribution of where the unique maximum occurs is the uniform distribution over the d trials:

Lemma 11.14. *Let d be a positive integer and $(\mathbf{X}_j)_{j \in [d]}$ be a family of independent geometric random variables with the same parameter $\lambda \in (0,1)$.*

Let $Y = \max_{j\in[d]} \mathbf{X}_j$. *Then:*

$$\forall i \in [d], \qquad \mathbb{P}[\mathbf{X}_i = Y \mid \exists! j : \mathbf{X}_j = Y] = \frac{1}{d} \ . \tag{11.6}$$

PROOF. Since variables $\mathbf{X}_1, \dots, \mathbf{X}_d$ are i.i.d., permuting their order does not change their joint distribution. Let $\mathcal{E}$ be the event that the maximum is unique. Observe that Y and $\mathcal{E}$ are invariant under permutations of $\mathbf{X}_i$'s. Hence, then $\mathbb{P}[\mathcal{E} \wedge \mathbf{X}_i = Y] = \mathbb{P}[\mathcal{E} \wedge \mathbf{X}_j = Y]$ for any pair $i, j \in [d]$. By Bayes rule, we infer Eq (11.6). ∎

11.3.2. Efficient Encoding of Maxima. In this section, we show that a set of t maxima of $d \leqslant n$ independent geometric random variables of parameter $1/2$ can be encoded in $\Theta(t+\log\log d)$-bits, with high probability. As we compute such sets of random variables to estimate neighborhood similarities, this ensures our algorithms are bandwidth-efficient.

The intuition for this result is that while each maxima can reach values as high as $\Theta(\log(d) + \log(n))$, requiring $\Theta(\log\log(d) + \log\log(n))$ bits to encode on their own, the values taken together are mostly concentrated around $\Theta(\log(d))$. This high concentration allows for more efficient encoding, by only storing a number $k \approx \log d$ around which the values are concentrated, along with the deviations from k.

Lemma 11.15. *Consider a set of* $t \times d$ *independent geometric random variables* $(\mathbf{X}_{i,j})_{i\in[t],j\in[d]}$ *of parameter* $1/2$ *and their associated maxima, the* t *random variables* $\mathbf{Y}_i = \max_{j\in[d]} \mathbf{X}_{i,j}$ *for each* $i \in [t]$. *We have*

$$\sum_{i=1}^{t} |\mathbf{Y}_i - \lceil \log d \rceil| \leqslant 12t \quad w.p. \quad 1 - 2^{-t/10+1} \ .$$

PROOF. Let $k = \lceil \log d \rceil$. For each $i \in [t]$, let $\mathbf{Y}_i^+ = \max(0, \mathbf{Y}_i - k)$ and $\mathbf{Y}_i^- = \max(0, k - \mathbf{Y}_i)$, such that $|\mathbf{Y}_i - k| = \mathbf{Y}_i^+ + \mathbf{Y}_i^-$. Let $\mathbf{Y}^+ = \sum_{i=1}^t \mathbf{Y}_i^+$ and $\mathbf{Y}^- = \sum_{i=1}^t \mathbf{Y}_i^-$. We prove the lemma by showing that $\mathbf{Y}_i^+$ and $\mathbf{Y}_i^-$ do not exceed $O(t)$ w.p. $1 - \exp(-\Omega(t))$.

By definition of $\mathbf{Y}_i$ (Claim 11.11), for any $x \geqslant 0$, we have that

$$\mathbb{P}[\mathbf{Y}_i^+ \geqslant x] = \mathbb{P}[\mathbf{Y}_i \geqslant x + k] = 1 - (1 - 2^{-x-k})^d \leqslant d \cdot 2^{-x-k} \leqslant 2^{-x} \ .$$

(using that $1 - yd \leqslant (1-y)^d$ for any $y \geqslant 0$)

If the sum $\mathbf{Y}^+$ reaches $4t$ or more, then there are numbers $x_1, \ldots, x_t$ such that $\sum_{i=1}^t x_i = 4t$ and for each $i \in [t]$, $\mathbf{Y}_i^+ \geqslant x_i$. For a fixed combination of x_i, since $\mathbf{Y}_i$'s are independent, the probability that this occurs is upper bounded by

$$\prod_{i=1}^t \mathbb{P}[\mathbf{Y}_i^+ \geqslant x_i] \leqslant \prod_{i=1}^t 2^{-x_i} = 2^{-\sum_{i=1}^t x_i} \leqslant 2^{-4t} .$$

There are $\binom{4t+t}{t-1} \leqslant \left(\frac{5t \cdot e}{t}\right)^t = 2^{t \log(5e)}$ ways to choose numbers $x_1, x_2, \ldots, x_t \geqslant 0$ such that they sum to $4t$. Thus, the probability that $\mathbf{Y}^+$ reaches $4t$ is bounded as follows:

$$\mathbb{P}[\mathbf{Y}^+ \geqslant 4t] \leqslant \binom{5t}{t-1} \cdot 2^{-4t} \leqslant 2^{-(4-\log(5e))t} \leqslant 2^{-t/10} .$$

The bound on $\mathbf{Y}^-$ is obtained in the same way, using that $\mathbb{P}[\mathbf{Y}_i^- \geqslant x_i] = \mathbb{P}[\mathbf{Y}_i \leqslant k - x_i + 1] = p_{k-x_i+1} \leqslant \exp(-2^{x_i-2}) \leqslant 2^{-x_i/2}$. Hence $\mathbb{P}[\mathbf{Y}^- \geqslant 12t] \leqslant 2^{-(6-\log(13e)t)} \leqslant 2^{-t/10}$. ∎

It follows that we can encode a sequence of maxima of geometric values compactly.

Lemma 11.16. *Let $\mathbf{Y}_i = \max_{j \in [d]} \mathbf{X}_{i,j}$ where $(\mathbf{X}_{i,j})_{i \in [t], j \in [d]}$ are independent geometric variables of parameter $1/2$. With probability $1 - 2^{-t/10+1}$, the sequence of values $(\mathbf{Y}_i)_{i \in [t]}$ can be described in $O(t + \log\log d)$ bits.*

Proof. To encode the set of maxima efficiently, compute an integer $k \in O(\log d)$ such that $\sum_{i=1}^t |\mathbf{Y}_i - k| \leqslant O(t)$. The existence of such a k is guaranteed with probability $1 - 2^{-t/10+1}$ by Lemma 11.15. We can take the minimal one, or the one that in general minimizes the size of our encoding. Writing the binary representation of k in the encoding takes $O(\log\log d)$ bits.

To encode the values $(\mathbf{Y}_i)_{i \in [t]}$, we write $|\mathbf{Y}_i - k|$ in unary, prefix it by the sign bit $\operatorname{sign}(\mathbf{Y}_i - k)$, and use 0 as a separator. In total, the encoding takes $O(\log\log d) + \sum_{i \in [t]} (|\mathbf{Y}_i - k| + 2) \leqslant O(t + \log\log d)$ bits. ∎

11.3.3. Approximate Counting from Fingerprinting. In this section, we explain how vertices use fingerprinting to approximates the number of $u \in N(v)$ such that $P_v(u) = 1$, for a binary predicate P_v. When

P_v is the trivial predicate (i.e., $P_v(u) = 1$ for all $u \in N(v)$), vertices approximate their degree, i.e., $|N(v)|$. Other examples of predicates we use are "neighbors outside of K_v" (when approximating external degrees) or "neighbors $u \in N(v)$ colored with $\varphi(u) > r_v$" (when estimating palette sizes in Claim 12.24). Importantly, P_v must be known to the machines of a cluster $V(v)$, as well as being efficiently computable by the machines with the knowledge they have. In the case of predicates related to the ACD, for instance, it suffices that each vertex informs its cluster of the ID of its almost-clique.

Lemma 11.17. *Let $\xi \in (0, 1/4)$ and, for each $v \in V_H$, let $P_v : N(v) \to \{0,1\}$ be a deterministic predicate such that if $P_v(u) = 1$, there exists $w \in V(v) \cap V(u)$ that knows it. There is a $O(\mathsf{cd} \cdot \xi^{-2})$-round algorithm for all nodes to estimate $|N_H(v) \cap P_v^{-1}(1)|$ with high probability within a multiplicative factor $(1 \pm \xi)$.*

Proof. Each vertex v samples $t = \Theta(\xi^{-2} \log n)$ independent geometric variables $\mathbf{X}_{v,1}, \dots, \mathbf{X}_{v,t}$ of parameter $1/2$ and broadcasts them within its support tree $T(v)$. Through aggregation, each vertex v computes $\mathbf{Y}_{v,i} = \max\{X_{u,i} : u \in N(v) \text{ and } P_v(u) = 1\}$ for each $i \in [t]$. Let $d = |N_H(v) \cap P_v^{-1}(1)|$. By Lemma 11.12, w.h.p., each vertex v deduces from $\{\mathbf{Y}_{v,i}\}_{i\in[t]}$ an estimate $\hat{d}_v \in (1 \pm \xi)d$.

We now argue that the maxima $\{\mathbf{Y}_{v,i}\}_{i\in[t]}$ can be aggregated efficiently. Let $\mathbf{Z}_{w,v,i} = \max\{\mathbf{X}_{u,i} : \exists e \in E_H(u,v), P_v(u) = 1 \text{ and } w = m(e)\}$ be the geometric variables that machine $w \in V(v)$ receives from neighbors of v. We aggregate variables $\mathbf{Z}_{w,v,i}$ by order of depth in the support trees. More precisely, we have d phases of $O(\xi^{-2})$ rounds each, such that at the end of phase $i \in [\mathsf{d}]$, each machine w at depth $\mathsf{d} - i$ in $T(v)$ has computed the coordinate-wise maximum of the variables $(\mathbf{Z}_{w,v,j})_{j\in[t]}$ from its subtree. Note that each partially aggregated set $\{\mathbf{Z}_{w,v,j}\}$ is a set of t maxima of independent geometric variables. Thus, to send the maxima to their parents in the support tree, vertices use the encoding scheme in Lemma 11.16. In total, $\mathsf{d}n \leqslant n^2$ aggregates are computed. By union bound over all those aggregates, w.h.p., the root of $T(v)$ learns $\mathbf{Y}_{v,i}$ for each $i \in [t]$ in $O(\xi^{-2})$. ∎

CHAPTER 12

Ultrafast $(\Delta+1)$-Coloring of Virtual Graphs

The goal of this section is to describe our $(\Delta+1)$-coloring algorithm for virtual graphs when $\Delta \geqslant \text{poly}(\log n)$. The algorithm for low-degree graphs is deferred to Chapter 13.

THEOREM 9.6. *Let H be a virtual graph on network G with n machines, bandwidth $\mathsf{b} = O(\log n)$, congestion $\mathsf{c} \leqslant n$ and dilation d. Suppose that Δ is the maximum degree of H and that it is known to all machines. There is a $O(\mathsf{cd} \cdot \log^* n)$-round algorithm to $(\Delta+1)$-color virtual graphs of maximum degree $\Delta \geqslant \Delta_{low} = \Omega(\log^{21} n)$, with high probability.*

The algorithm follows the same high-levels steps as the LOCAL algorithm except that vertices are colored in a carefully chosen order. We begin with the high-level algorithm with emphasis on how we assemble each piece together, while the most involved intermediate steps (coloring put-aside sets, colorful matching in cabals and reserved colors) are deferred to later sections.

ALGORITHM 27. High-Level Coloring Algorithm for Theorem 9.6

Input: A cluster graph H on G such that $\Delta \geqslant \Delta_{low}$

Output: A $\Delta+1$-coloring

(1) Compute the almost-clique decomposition
(2) SLACKGENERATION in $V \setminus V_{cabal}$
(3) SLACKCOLOR in V_{sparse}
(4) Color the Non-Cabals
(5) Color the Cabals

The main coloring steps are Steps 3 to 5. In Step 1, we determine when each vertex will be colored. Step 2 provides slack necessary for Steps 3

and 4. Each step of Algorithm 27 operates *almost* independently of the other ones. We provide pre/post-conditions of each step.

12.1. The High-Level Algorithm

In this section we give the definitions essential to our algorithm and state the properties of each step in Algorithm 27. We conclude the section with a proof of Theorem 9.6.

Global Parameters. We define here the precise values of some parameters used throughout the chapter.

$$\begin{aligned} \varepsilon &= 1/2000 \ , \\ \delta &= \gamma_{12.3}/300 \ , \\ \Delta_{low} &= \Theta\big(\log^{21} n\big) \quad \text{and} \\ \ell_{\min} &= \Theta\big(\log^{1.1} n\big) \ , \end{aligned} \tag{12.1}$$

where $\gamma_{12.3} = \gamma_{12.3}(\varepsilon) \in (0,1)$ is the small universal constant from slack generation such that sparse vertices have at least $\gamma_{12.3} \cdot \zeta(v)$ slack (Proposition 12.3).

Almost-Clique Decomposition. Recall how one computes almost-clique decompositions: compute friendly edges, find vertices incident to many of friendly edges, and run a two step BFS to explore each almost-clique. To detect friendly edges, we can use the same algorithm as in BCONGEST (Section 4.2) since it only requires vertices to broadcast one $O(1/\varepsilon^2)$-bit message and aggregate a bit-wise OR. With fingerprinting (Lemma 11.17), vertices can then estimate up to a small error the number of friendly edges they are adjacent to. As explained in Section 7.1 (see also [ACK19, Section 4.1]), these approximations suffice to find an almost-clique decomposition.

Proposition 12.1. *There exists an algorithm* ComputeACD *that computes an ε-almost-clique decomposition in $O(\mathsf{cd}/\varepsilon^4)$-rounds in virtual graphs.*

Cabals & Non-Cabals. Recall that a vertex has sparsity proportional to $a(v) + e(v)$ (Lemma 3.8). To know exactly how sparse/dense a vertex is, it approximates its external degree $\widetilde{e}(v) \in (1 \pm \delta)e(v)$ using the fingerprinting technique (Lemma 11.17). Observe that by computing a BFS tree in each

almost-clique, we can compute $|K|$ exactly and aggregate the average of the approximate external degrees. We remark that anti-degrees cannot be approximated the same way using fingerprinting. Since the densest almost-cliques play a key role, we give them a special name:

Definition 12.2. A *cabal* is an almost-clique K such that

$$\widetilde{e}(K) \stackrel{\text{def}}{=} \frac{1}{|K|}\sum_{v\in K}\widetilde{e}(v) < \ell_{\min}\ .$$

We denote by $\mathscr{K}_{cabal}$ the set of all cabals and by V_{cabal} the set of vertices v such that $K_v \in \mathscr{K}_{cabal}$.

Reserved Colors. Like in BCONGEST (Chapter 4), we avoid using certain colors in the earlier steps of the algorithm. Each almost-clique K reserves the colors $\{1, 2, \dots, r_K\}$, where

$$r_K = 250 \cdot \max\{\widetilde{e}(K), \ell_{\min}\} \tag{12.2}$$

depends on the density of K. By extension, define $r_v = r_{K_v}$ (and $r_v = 0$ if $v \notin V_{dense}$). Note that in all K the number of reserved colors is $r_K \leqslant 250(1+\delta)\varepsilon\Delta \leqslant 300\varepsilon\Delta$, a small fraction of the color space. Hence, those colors are dispensable in slack generation and in finding a colorful matching.

Slack Generation. Slack generation (Algorithm 2 in Section 3.3) can be implemented in $O(\mathsf{cd})$ in virtual graphs since it only requires that vertices try one random color in $(r, \Delta+1]$. We restate the guarantees of slack generation (see Section 3.3) in a form more appropriate to our needs for this chapter.

Proposition 12.3 (Reformulation of Proposition 3.15)**.** *Suppose V_{sparse}, V_{dense} is an ε-almost-clique decomposition. There exists a constant $\gamma_{12.3} = \gamma_{12.3}(\varepsilon) \in (0,1)$ such that if $\Delta \geqslant \Omega(\gamma_{12.3}^{-1}\log n)$ and φ_{sg} is the (partial) coloring produced by* SlackGeneration*, then $\varphi_{sg}(V_H) \cap [300\varepsilon\Delta] = \emptyset$ and with high probability,*

(1) $s_{\varphi_{sg}}(v) = |L_{\varphi_{sg}}(v)| - \deg_{\varphi_{sg}}(v) \geqslant \gamma_{12.3}\cdot\Delta$ for all $v \in V_{sparse}$;
(2) $|N(v)\cap \operatorname{dom}\varphi_{sg}| - |\varphi_{sg}(N(v))| \geqslant \gamma_{12.3}\cdot e(v)$ for all $v \in V_{dense}$ with $e(v) \geqslant \Omega(\gamma_{12.3}^{-1}\log n)$; and
(3) each K contains $|K \cap \operatorname{dom}\varphi_{sg}| \leqslant |K|/100$ colored vertices.

PROOF. To obtain (3), set the activation probability to $p = 1/200$. Then (1) follows from Proposition 3.15 and $\zeta(v) \geqslant \Omega(\varepsilon^2\Delta)$ for every sparse vertex. For dense vertices, recall that $\zeta(v) \geqslant \Omega(\varepsilon e(v))$ (Part 3 in Proposition 3.6), so Proposition 3.15 implies (2). ∎

Coloring Algorithms. We can now state properties required by each of the coloring algorithms in Algorithm 27. There are three coloring steps: sparse vertices V_{sparse}, non-cabals $V_{dense} \setminus V_{cabal}$, and cabals V_{cabal}. Coloring sparse vertices follows from Propositions 11.7 and 12.3 (SLACKCOLOR), and hence we defer details to the proof of Theorem 9.6. Non-cabal vertices need the slack from SLACKGENERATION, but cabal vertices must remain uncolored for COLORINGCABALS to work. Thus, we run SLACKGENERATION everywhere *except in cabals*, and then color $V_{dense} \setminus V_{cabal}$ first. Only then do we color V_{cabal} (Part (NC-2)). Also note that steps prior to non-cabals cannot use reserved colors (Part (NC-3)).

Proposition 12.4. *Let φ be a coloring such that*

(NC-1) we did slack generation in $V \setminus V_{cabal}$, i.e., $\varphi \succeq \varphi_{sg}$;
(NC-2) cabals are uncolored, i.e., $V_{cabal} \subseteq V \setminus \operatorname{dom}\varphi$;
(NC-3) no reserved color is used in non-cabals, i.e., $\varphi(K) \cap [300\varepsilon\Delta] = \emptyset$ for all $K \notin \mathscr{K}_{cabal}$.

Then, w.h.p., COLORINGNONCABALS colors all vertices in $V_{dense} \setminus V_{cabal}$ in $O(\mathsf{cd} \log^* n)$ rounds.

Finally, we color cabals.

Proposition 12.5. *Let φ be a coloring where cabals are not colored. Then, w.h.p., in $O(\mathsf{cd} \log^* n)$ rounds COLORINGCABALS extends φ such that all cabals are colored.*

PROOF OF THEOREM 9.6. By Proposition 12.1, COMPUTEACD returns an ε-almost-clique decomposition in $O(\mathsf{cd}/\varepsilon^4) = O(\mathsf{cd})$ rounds. Each $v \in V_{dense}$ computes $\widetilde{e}(v) \in (1\pm\delta)e(v)$ in $O(\mathsf{cd}/\delta^2) = O(\mathsf{cd})$ rounds using the fingerprinting technique (Lemma 11.17 with $P_v(u) = 1$ iff $u \notin K_v$). Through aggregation on a BFS tree spanning K, vertices compute $|K|$ exactly and

approximate the average external degree $\widetilde{e}(K) \in (1 \pm \delta)e(K)$. In particular, each $v \in V_{dense}$ knows if $K_v \in \mathscr{K}_{cabal}$.

After slack generation, w.h.p., all sparse vertices have $\gamma_{12.3}\Delta$ slack by Proposition 12.3. By Proposition 11.7 with $\text{List}(v) = [\Delta + 1]$, $\delta = \gamma_{12.3} = O_\varepsilon(1)$, $\kappa = 1/10$ and $s(v) = \gamma_{12.3}\Delta$, every sparse vertex gets colored in $O(\mathsf{cd} \log^* n)$ rounds.

Preconditions for ColoringNonCabals are verified because we ran SlackGeneration in $V \setminus V_{cabal}$ (Part (NC-1)), ColoringSparse and SlackGeneration do not color vertices of V_{cabal} (Part (NC-2)) and SlackGeneration does not use reserved colors (Part (NC-3)). With high probability, ColoringNonCabals colors $V_{dense} \setminus V_{cabal}$ in $O(\mathsf{cd} \log^* n)$ rounds. None of the vertices in V_{cabal} have been colored thus far. By Proposition 12.5, w.h.p., ColoringCabals colors V_{cabal} in $O(\mathsf{cd} \log^* n)$ rounds. ∎

12.2. Coloring Non-Cabals

This section aims at proving Proposition 12.4 by arguing the correctness of Algorithm 28. The overall structure follows the one of the BCONGEST algorithm from Chapter 4 with important internal changes. Inliers are defined differently because vertices cannot approximate anti-degrees (hence $a(K)$) accurately. Step 4 in Algorithm 28 requires careful approximation of palette sizes, which is technical and deferred to Section 12.6.

Proposition 12.4. *Let φ be a coloring such that*

- *(NC-1) we did slack generation in $V \setminus V_{cabal}$, i.e., $\varphi \succeq \varphi_{sg}$;*
- *(NC-2) cabals are uncolored, i.e., $V_{cabal} \subseteq V \setminus \operatorname{dom} \varphi$;*
- *(NC-3) no reserved color is used in non-cabals, i.e., $\varphi(K) \cap [300\varepsilon\Delta] = \emptyset$ for all $K \notin \mathscr{K}_{cabal}$.*

Then, w.h.p., ColoringNonCabals colors all vertices in $V_{dense} \setminus V_{cabal}$ in $O(\mathsf{cd} \log^* n)$ rounds.

Algorithm 28. ColoringNonCabals

Input: A coloring φ such as given in Proposition 12.4

Output: A total coloring of $V \setminus V_{cabal}$

(1) ColorfulMatching
(2) ColoringOutliers
(3) SynchronizedColorTrial
(4) Complete

Inliers & Outliers. Vertices that differ significantly from the average may not receive enough slack from slack generation and colorful matching to be colored later in the algorithm. Those vertices are called *outliers* $O_K \subseteq K$ while their complement in K is called *inliers* $I_K = K \setminus O_K$. It would suffice to guarantee $e(v) \leqslant O(e(K))$ and $a(v) \leqslant O(a(K))$. While external degrees can be approximated (allowing the first condition in Eq (12.4) to be verified), approximating anti-degrees is more challenging. We exploit the following relation (derived from counting neighbors of v inside and outside K)

$$\Delta + 1 = (\Delta - \deg(v)) + \deg(v) + 1 = (\Delta - \deg(v)) + |K| + e(v) - a(v) \ .$$

Hence, vertices can approximate their anti-degree as

$$x_v \stackrel{\text{def}}{=} |K| - (\Delta + 1) + \widetilde{e}(v) \quad \in \quad a(v) - (\Delta - \deg(v)) \pm \delta e(v) \ . \tag{12.3}$$

Intuitively, the error made in Eq (12.3) is compensated for by the slack provided to v. Inliers are then defined as

$$I_K \stackrel{\text{def}}{=} \left\{ u \in K : \widetilde{e}(v) \leqslant 20\widetilde{e}(K) \text{ and } x_v \leqslant \frac{M_K}{2} + \frac{\gamma_{12.3}}{8}\widetilde{e}(K) \right\} , \tag{12.4}$$

where M_K is the size of the colorful matching in K. Henceforth, we focus on coloring inliers and assume all outliers have been colored. Outliers are colored after the colorful matching, while they have $\Omega(\Delta)$ temporary slack from adjacent uncolored inliers. Let us argue that inliers represent a large constant fraction of each almost-clique.

Lemma 12.6. *For $K \notin \mathscr{K}_{cabal}$, the number of inliers is $|I_K| \geqslant 0.85|K| \geqslant 0.8\Delta$.*

Proof. Let Z be the set of vertices v in K with $a(v) \leqslant 20a(K)$ and $e(v) \leqslant 15e(K)$. We claim that all vertices in Z are inliers. The lemma

follows then, since by Markov at most $(1/15 + 1/20)|K| \leqslant 0.15|K| \leqslant 0.15(1+\varepsilon)\Delta$ vertices are outside Z. We first derive a useful bound (letting $M_K = 0$ when $a(K) = O(\log n)$, as no colorful matching is computed in that case):

$$80a(K) \leqslant M_K + \gamma_{12.3} e(K)/8 \ . \tag{12.5}$$

Eq (12.5) holds when $a(K) \gg \log n$, because $M_K \geqslant \Omega(a(K)/\varepsilon) \geqslant 80a(K)$; while when $a(K) = O(\log n)$, then $a(K) \ll e(K)$, since $e(K) = \Omega(\log^{1.1} n)$ in non-cabals. Consider $v \in Z$. Setting $\delta \leqslant \gamma_{12.3}/300$, by the definition of x_v and Z and Eq (12.5),

$$\begin{aligned} x_v \leqslant a(v) + \delta e(v) &\leqslant 20a(K) + \frac{15\delta}{1-\delta}\widetilde{e}(K) \\ &\leqslant \left(\frac{M_K}{2} + \frac{\gamma_{12.3}}{16}\widetilde{e}(K)\right) + \frac{\gamma_{12.3}}{16}\widetilde{e}(K) \\ &= \frac{M_K}{2} + \frac{\gamma_{12.3}}{8}\widetilde{e}(K) \ . \end{aligned}$$

Hence, v is an inlier, as claimed. ∎

As we argue in the next lemma, all vertices classified as inliers received sufficient slack *even when restricted to colors of the clique palette*. Eq (12.6) will be crucial in coloring the inliers remaining after the synchronized color trial.

Lemma 12.7. *There exists a universal constant $\gamma_{12.7} = \gamma_{12.7}(\varepsilon) \in (0,1)$ such that the following holds. Let φ be the coloring produced by running slack generation and colorful matching. Then, w.h.p., for all inliers $v \in I_{K_v}$ in non-cabals $K_v \notin \mathscr{K}_{cabal}$, the number of* non-reserved *repeated colors in $K \cup E(v)$ is at least:*

$$\begin{aligned} &|\{u \in K_v \cup E(v) : \varphi(u) > r_v\}| - |(r_v, \Delta+1] \cap \varphi(K_v \cup E(v))| \\ &\qquad \geqslant \gamma_{12.7} \cdot e(K) + 40a(K) + x_v \ . \end{aligned} \tag{12.6}$$

Proof. Slack generation creates $\gamma_{12.3} \cdot e(v)$ reuse slack in $N(v)$ *when* $e(v) \gg \gamma_{12.3}^{-1} \log n$ (Proposition 12.3) and the colorful matching creates $M_K = \Omega(a(K)/\varepsilon) \geqslant 100a(K)$ repeated colors in K *when* $a(K) \gg \log n$, w.h.p. Neither algorithm uses reserved colors. Recall that $e(K) \geqslant \ell_{\min}/2$ in non-cabals and by definition of inliers $x_v \leqslant \gamma_{12.3}/8 \cdot e(K) + M_K/2$. Eq (12.6) follows from case analysis.

- First, suppose that $a(K) \geqslant e(K)/2$. Since $K \notin \mathscr{K}_{cabal}$, we have $a(K) \geqslant e(K)/2 \geqslant \ell_{\min}/4 \gg \gamma_{12.3}^{-1} \log n$. Hence, the colorful matching provides enough reuse slack: $M_K \geqslant e(K) + 40a(K) + x_v$.
- Next, suppose $e(v) \geqslant e(K)/4$ and $a(K) \leqslant e(K)/2$. Then the reuse slack is $\gamma_{12.3} \cdot e(v) + M_K \geqslant \gamma_{12.3}/4 \cdot e(K) + M_K \geqslant \gamma_{12.3}/8 \cdot e(K) + M_K/2 + x_v$. This is at least $\gamma_{12.3}/16 \cdot e(K) + 40a(K) + x_v$, by Eq (12.5).
- Finally, suppose $e(v) \leqslant e(K)/4$ and $a(K) \leqslant e(K)/2$. Then, the clique must be smaller than Δ because $(\Delta + 1) - |K| \geqslant e(K) - a(K) \geqslant e(K)/2$. In particular, by definition (Eq (12.3)), we have $x_v \leqslant \widetilde{e}(v) - e(K)/2 \leqslant -e(K)/5$ (for $\delta < 1/5$). Then the reuse slack from colorful matching is at least $M_K \geqslant (x_v + e(K)/5) + M_K \geqslant x_v + \gamma_{12.3} e(K) + 40a(K)$, by Eq (12.5) and that $\gamma_{12.3} \leqslant 1/10$.

Combined, Eq (12.6) holds with $\gamma_{12.7} \stackrel{\text{def}}{=} \gamma_{12.3}/16$. ∎

We deduce from Lemma 12.7 an equivalent of the Accounting Lemma from Section 4.6 for our modified definition of inliers.

Lemma 12.8 (Accounting Lemma II)**.** *For any φ extending the coloring produced by slack generation and the colorful matching, w.h.p., in all non-cabals $K \notin \mathscr{K}_{cabal}$, there are at least*

$$|L_\varphi(v) \cap L_\varphi(K)| \geqslant |(N(v) \cup K) \setminus \operatorname{dom} \varphi|$$

colors available to $v \in I_K$ in the clique palette.

Proof. Let $R = |(N(v) \cup K_v) \cap \operatorname{dom} \varphi| - |\varphi(N(v) \cup K_v)|$ be the number of repeated colors in $N(v) \cup K_v$. The number of colors in the clique palette available to v is

$$|L_\varphi(v) \cap L_\varphi(K)| \geqslant \Delta + 1 - |K \cap \operatorname{dom} \varphi| - |E(v) \cap \operatorname{dom} \varphi| + R \,.$$

Then, using $|K \cap \operatorname{dom}\varphi| = |K| - |K \setminus \operatorname{dom}\varphi|$, $|E(v) \cap \operatorname{dom}\varphi| = e(v) - |E(v) \setminus \operatorname{dom}\varphi|$, and $\Delta + 1 - |K| = (\Delta - \deg(v)) + e(v) - a(v)$, this becomes

$$\begin{aligned} &|L_\varphi(v) \cap L_\varphi(K)| - |(N(v) \cup K) \setminus \operatorname{dom}\varphi| \\ &\qquad \geqslant (\Delta - \deg(v)) + R - a(v) \\ &\qquad \geqslant (\Delta - \deg(v)) + R - x_v - (a(v) - x_v) \\ \text{(by definition of } x_v) \qquad &\qquad \geqslant R - x_v - \delta e(v)\ . \end{aligned}$$

By Eq (12.6), the number of repeated colors is $R \geqslant \gamma_{12.7} \cdot e(K) + x_v$ is large. Since $e(v) \leqslant 25e(K)$ and $\delta < \gamma_{12.7}/25$ is small, $R \geqslant x_v + \delta e(v)$ which concludes the proof. ∎

Lemma 12.8 implies that $|L_\varphi(K)| \geqslant |K \setminus \operatorname{dom}\varphi|$ for any coloring after the colorful matching. In particular, if we let $S \subseteq K \setminus \operatorname{dom}\varphi$ be all but r_K of the uncolored inliers, there are enough colors in $L(K)$ for all vertices of S. We refer readers to the proof of Proposition 12.4 at the end of this section for more details.

Preparing MultiColorTrial (Section 12.6). The prior steps of the algorithm produced a coloring where uncolored vertices have slack $\Omega(e(K))$ for a small constant while uncolored degrees are $O(e(K))$ for a large hidden constant. Before we can apply MultiColorTrial, we must reduce uncolored degrees to a small constant factor of the slack (Part ii) of Proposition 11.7). Moreover, we must do so without using too many reserved colors. This step is similar to Step 3 of the BCONGEST algorithm (Section 4.8).

This necessitates detecting vertices with enough slack in reserved colors, which is challenging in cluster graphs because vertices do not have access to their palettes. Hence, some complications ensue that are deferred to Section 12.6. We emphasize that this problem does not occur in cabals because we can easily adjust the size of put-aside sets and colorful matching (contrary to the slack received during slack generation).

Proposition 12.9. *Let φ be a coloring such that reserved colors are unused ($[r_K] \cap \varphi(K) = \emptyset$ in all $K \notin \mathcal{K}_{cabal}$), Eq (12.6) holds, and vertices have uncolored degree $O(e(K))$. There is an algorithm that extends φ to $V_{dense} \setminus V_{cabal}$ in $O(\mathsf{cd} \log^* n)$ rounds with high probability.*

We can now prove that Algorithm 28 indeed colors non-cabal vertices in $O(\mathsf{cd} \log^* n)$ rounds with high probability.

Proof of Proposition 12.4. We argue that dense non-cabal vertices are colored in $O(\mathsf{cd} \log^* n)$ rounds. We go over each step of Algorithm 28.

Colorful Matching (Step 1). We run the colorful matching algorithm in all almost-cliques. Using the query algorithm (Lemma 11.3) to compare the number of colors in $L(K)$ before and after computing the colorful matching, vertices learn M_K. (A vertex is colored in Step 1 iff it provides slack). If $M_K \geqslant 3\varepsilon\Delta$, then all vertices of K have $M_K - a(v) \geqslant \varepsilon\Delta$ slack because $|L(v)| \geqslant \Delta + 1 - |(K_v \cup N(v)) \cap \operatorname{dom}\varphi| + M_K \geqslant \deg_\varphi(v) + M_K - a(v)$. If this occurs, SlackColor colors *all vertices* of K in $O(\mathsf{cd} \log^* n)$ rounds using $\operatorname{List}(v) = [\Delta + 1]$ and $s(v) = \varepsilon\Delta$. We henceforth assume $M_K \leqslant 3\varepsilon\Delta$.

Coloring Outliers (Step 2). Since slack generation colored at most $0.01|K| \leqslant 0.02\Delta$ vertices in K, there are at least $(0.8 - 0.02 - 3\varepsilon)\Delta \geqslant 0.75\Delta$ uncolored inliers for $\varepsilon \leqslant 1/100$. Moreover, each outlier is adjacent to at least $(0.75 - \varepsilon)\Delta \geqslant 0.5\Delta$ uncolored inliers. After removing the at most $300\varepsilon\Delta$ reserved colors, outliers still have 0.25Δ slack. Hence outliers are colored in $O(\mathsf{cd} \log^* n)$ rounds by SlackColor *without using reserved colors* — by Proposition 11.7 with $\operatorname{List}(v) = [\Delta + 1] \setminus [r_K]$ and $s(v) = 0.25\Delta$. We henceforth assume outliers are colored and focus on inliers.

Synchronized Color Trial (Step 3). In each non-cabal, define $S_K \subseteq K \setminus \operatorname{dom}\varphi$ as an arbitrary set of $|K \setminus \operatorname{dom}\varphi| - r_K$ uncolored inliers that participate in the synchronized color trial. There are at least 0.75Δ uncolored inliers and $r_K \leqslant 300\varepsilon\Delta$; hence, the number of vertices participating in the synchronized color trial is $|S_K| \geqslant 0.75\Delta - r_K \geqslant (0.75 - 300\varepsilon)\Delta \geqslant 0.5|K|$ for $\varepsilon < 1/1202$. On the other hand, Lemma 12.8 implies that $|L_\varphi(K)| \geqslant |K \setminus \operatorname{dom}\varphi| = |S_K| + r_K$. Hence, both conditions of Lemma 3.25 with $\mathscr{C}_K = L_\varphi(K) \setminus [r_K]$ are verified and by Lemma 11.10, we implement the synchronized color trial in $O(\mathsf{cd})$ rounds.

Finishing the Coloring (Step 4). After the synchronized color trial, by Lemma 3.25 (with $\alpha = 1/2$), each K contains at most $r_K + 50e(K) \leqslant 300e(K)$ uncolored vertices, with high probability. Adding external neighbors, the maximum uncolored degree is $300e(K) + e(v) \leqslant 350e(K)$ (as

$e(v) \leqslant 20\frac{1+\delta}{1-\delta}e(K) \leqslant 50e(K)$). Recall slack generation and colorful matching do not use reserved colors. We were careful not to use reserved colors during the synchronized color trial. By Lemma 12.7, w.h.p., Eq (12.6) holds. All conditions of Proposition 12.9 are therefore verified and we complete the coloring of $V_{dense} \setminus V_{cabal}$ in $O(\mathsf{cd} \log^* n)$ rounds. ∎

12.3. Coloring Cabals

Once non-cabal vertices are colored, we color cabals. We emphasize that we make no assumption about the coloring computed in ColoringNonCabals besides that it does not color any vertex in V_{cabal}. The task of this section is to prove Proposition 12.5, by arguing the correctness of Algorithm 29. Since there is a significant overlap between Algorithm 28 and Algorithm 29, the exposition focuses on the differences (put-aside sets and colorful matching). A proof going over all steps of Algorithm 29 can be found at the end of the section.

Proposition 12.5. *Let φ be a coloring where cabals are not colored. Then, w.h.p., in $O(\mathsf{cd} \log^* n)$ rounds* ColoringCabals *extends φ such that all cabals are colored.*

Algorithm 29. ColoringCabals

Input: A total coloring φ_0 of $V_H \setminus V_{cabal}$

Output: A total coloring of V_{cabal}

(1) ColorfulMatching
(2) ColoringOutliers
(3) ComputePutAsideSets
(4) SynchronizedColorTrial
(5) MultiColorTrial
(6) ColorPutAside

Reserved Colors. Recall cabals are almost-cliques where $\widetilde{e}(K) \leqslant \ell_{\min}$. All cabals have the same number of reserved colors $r \stackrel{\text{def}}{=} r_K = 250\ell_{\min}$ (see Eq (12.2)).

Finding a Colorful Matching in Cabals (Section 12.4). To color put-aside sets in Step 6, we must have a colorful matching *even when* $a(K) \leqslant O(\log n)$. We introduce a novel algorithm based on the fingerprinting techniques to compute a colorful matching in cabals where $a(K) \leqslant O(\log n)$; see Section 12.4 for more details. We run first the BCONGEST algorithm (Lemma 11.9) and if it results in a matching of size $O(\log n)$, we cancel the coloring and run our new algorithm. This is the distributed algorithm that works in these extremely dense almost-cliques. We emphasize that we do not necessarily find a matching of size $\Omega(a(K)/\varepsilon)$. However, it suffices to find a matching of size M_K such that $M_K \geqslant a(v)$ for almost all vertices v of K.

Proposition 12.10. *Assume $\Delta \gg \log^2 n$. Suppose all vertices in cabals K with $a(K) \leqslant O(\log n)$ are uncolored. Let φ be the coloring produced by* ColorfulMatchingCabals *in $O(\mathsf{cd}(\log^* n))$ rounds. With high probability, in each cabal K such that $a(K) \leqslant O(\log n)$, for at least $(1-10\varepsilon)\Delta$ vertices $v \in K$ the size of the colorful matching exceeds their anti-degrees $a(v) \leqslant M_K \stackrel{def}{=} |K \cap \operatorname{dom}\varphi| - |\varphi(K)|$. Moreover, the algorithm does not use reserved colors.*

Inliers & Outliers. In cabals, it suffices that inliers have external degree $O(e(K))$ because we create slack using put-aside sets. Formally, in each $K \in \mathscr{K}_{cabal}$, inliers are $I_K \stackrel{\text{def}}{=} \{u \in K : \widetilde{e}(v) \leqslant 20\widetilde{e}(K)\}$. The following lemma is clear from Markov inequality.

Lemma 12.11. *For $K \in \mathscr{K}_{cabal}$, the number of inliers $|I_K| \geqslant 0.9\Delta$.*

In cabals, it suffices that for almost all vertices there are as many available colors in the clique palette as uncolored vertices in K. Lemma 4.19 shows that $a(v) \leqslant M_K$ suffices for this to hold. Note that vertices cannot check if $a(v) \leqslant M_K$. Part of the error in the approximation we used in non-cabals (Eq (12.3)) was balanced by slack from slack generation, which we cannot run in cabals. Fortunately, in cabals, we will not need to detect when $a(v) \leqslant M_K$.

Computing Put-Aside Sets. Recall that the put-aside sets P_K should have two properties: *i)* they have size r each (where r is the number of

reserved colors), and *ii*) no two such sets have an edge between them. The aim is to color P_K only at the very end so that we can avoid using colors $\{1, 2, \dots, r\}$ before calling MULTICOLORTRIAL. In this chapter, we introduce the additional guarantee (compared to the LOCAL or BCONGEST algorithms) that each cabal contains only a few vertices adjacent to vertices in put-aside sets from other cabals, which will be necessary for coloring put-aside sets at the end.

Lemma 12.12. *Assume $\Delta \gg \ell^2 \log n$. Let φ be a coloring such that $|I_K \setminus \operatorname{dom}\varphi| \geqslant 0.75\Delta$. There is a $O(\mathsf{cd})$-round algorithm computing sets $P_K \subseteq I_K \setminus \operatorname{dom}\varphi$ such that, w.h.p., for each cabal K,*

(1) $|P_K| = r$,
(2) there are no edges between P_K and $P_{K'}$ for $K' \neq K$,
(3) at most $|K|/100$ inliers in K have neighbors in $\bigcup_{K' \in \mathscr{K}_{cabal} \setminus \{K\}} P_{K'}$.

PROOF. Run LOWDEGREESAMPLE with $p = 1/r$ where each S_i is the set of inliers in the i-th cabal. Since inliers have at most $25\ell_{\min} \leqslant r$ external neighbors and $\Delta \gg \ell^2_{\min} \log n$, by Lemma 3.26, we obtain sets $P_K^{safe} \subseteq I_K$ of size at least $|I_K|p/5$ for each $K \in \mathscr{K}_{cabal}$ such that no edge connects P_K^{safe} to $P_{K'}^{safe}$ for $K' \neq K$. Then, we sample each vertex of P_K^{safe} into P_K independently with probability $q = 1/(5000\ell_{\min})$. By the classic Chernoff Bound (Lemma 2.2), w.h.p., each P_K contains at least $|I_K|p/(50000\ell_{\min}) \geqslant r$ vertices (Lemma 12.11 and $\Delta \geqslant \Delta_{low}$). Part 1 and Part 2 thus hold with high probability.

For the remaining of this proof, we focus on proving Part 3. Consider a cabal K, call its inliers $v_1, \dots, v_{|I_K|}$. Let $\mathbf{X}_i$ be the random variable indicating if v_i has an external neighbor in some put-aside set $P_{K'}$ for $K' \neq K$. Observe that each vertex joins an a put-aside set independently and with probability at most $q = 1/(5000\ell_{\min})$. As inliers have at most $25\ell_{\min}$ external neighbors, we have that $\mathbb{P}[\mathbf{X}_i = 1] \leqslant 1/200$ by Markov inequality. Hence, in expectation, at most $|I_K|/200$ inliers of K have an external neighbor in a put-aside set. To show concentration on $\mathbf{X} = \sum_i \mathbf{X}_i$, we use the read-$k$ bound with $k = \Theta(\ell_{\min})$ (Lemma 2.6 with $n = |I_K|$ and $\delta = 1/200$). Indeed, the variable $\mathbf{X}_i$ depends on two independent binary random variables for each external neighbor $u \in E(v_i)$ that is an inlier in some other

cabal (one random variable for the activation in LowDegreeSample and one for the sampling in $P_{K'}$). Since each such u is an inlier from a cabal, its corresponding variables influence at most k external neighbors of u in K. Thus, w.p. $1 - \exp(-\Theta(|K|/k)) \geqslant 1 - 1/\text{poly}(n)$, we get that at most $|K|/100$ inliers have an external neighbor in a put-aside set. ∎

Coloring Put Aside Sets (Section 12.5). Only put-aside sets remain to color. The following lemma states it can be done in $O(\mathsf{cd})$ rounds, thereby concluding the proof of Proposition 12.5.

Proposition 12.13. *Suppose φ is a coloring such that only put-aside sets are uncolored and at least 0.9Δ vertices in each cabal verify $a(v) \leqslant M_K$. Then there is a $O(\mathsf{cd})$-round algorithm that computes a total coloring of H, with high probability.*

Note that it *does not* extend the coloring produced by earlier steps of the algorithm, but instead *exchanges* colors with a few already colored inliers in K. Recoloring vertices must be done carefully, as it must happen in all cabals in parallel without creating monochromatic edges. As ColorPutAsideSets is quite involved, we defer its description and analysis to Section 12.5.

Proof of Proposition 12.5. We go over the steps of Algorithm 29.

Colorful Matching (Step 1). We say we found a sufficiently large colorful matching M_K if all but 0.1Δ vertices in K have $a(v) \leqslant M_K$. Observe that $M_K \geqslant \Omega(a(K)/\varepsilon)$ is sufficiently large because, by Markov, at most $O(\varepsilon\Delta)$ can have $a(v) > \Omega(a(K)/\varepsilon)$. Run Algorithm 14 (see Section 4.6) in each cabal the colorful matching. If $a(K) \geqslant C \log n$ for some constant $C > 0$, we find a sufficiently large colorful matching with high probability. Assume $a(K) \leqslant C \log n$. If the algorithm produces a matching of size $\Omega(C/\varepsilon \cdot \log n) \geqslant \Omega(a(K)/\varepsilon)$, we found a large enough matching. Otherwise, vertices can compute M_K using the query algorithm (Lemma 11.3) and detect that the matching might be too small. We emphasize that vertices do not know $a(K)$ but it suffices to compare M_K to $\Omega(C/\varepsilon \cdot \log n)$. In that case, all vertices of K drop their colors and run the algorithm of Proposition 12.10. With high probability, we find a sufficiently large matching in those cabals.

As in Proposition 12.4, if $M_K \geqslant 2\varepsilon\Delta$, we can colors all vertices of K in $O(\mathsf{cd} \log^* n)$ rounds with high probability. We henceforth assume *all* cabals contain a sufficiently large colorful matching such that $M_K \leqslant 2\varepsilon\Delta$.

Coloring Outliers (Step 2). Recall that, in cabals, inliers are defined as vertices of low-external degree (verifying only $\widetilde{e}(v) \leqslant 20\widetilde{e}(K)$); hence, $|I_K| \geqslant 0.9\Delta$ (Lemma 12.11). Therefore, as in Proposition 12.4, w.h.p., all outliers get colored in $O(\mathsf{cd} \log^* n)$ rounds and we henceforth focus on inliers.

Computing Put-Aside Sets (Step 3). By Lemma 12.12, we find put-aside sets P_K (which are all uncolored inliers) in $O(\mathsf{cd})$ rounds with high probability.

Synchronized Color Trial (Step 4). Let $S_K = K \setminus (P_K \cup \operatorname{dom}\varphi)$ be the uncolored inliers that are not put-aside vertices. Since $|I_K| \geqslant 0.8\Delta$ and $|P_K| = r = O(\ell_{\min}) \leqslant 0.1\Delta$, the set S_K contains at least $|S_K| \geqslant 0.5|K|$ vertices. On the other hand, by Lemma 4.19, the clique palette contains enough colors to run the synchronized color trial with $\mathscr{C}_K = |L_\varphi(K)| \setminus [r]$ since $|L_\varphi(K)| \geqslant |K \setminus \operatorname{dom}\varphi| = |S_K| + |P_K| = |S_K| + r$. Conditions for the synchronized color trial are verified and afterward at most $16(1+\delta)\ell_{\min} + O(\log n) \leqslant 50\ell_{\min}$ vertices in S_K remain uncolored for each cabal K, by Lemmas 3.25 and 11.10 (with $\alpha = 1/2$), with high probability.

MultiColorTrial (Step 5). Let H' be the graph induced by $\bigcup_{K \in \mathscr{K}_{cabal}} K \setminus (P_K \cup \operatorname{dom}\varphi_{sct})$, the uncolored vertices not in put-aside sets. The maximum degree *in* H' is at most $71\ell_{\min}$ (at most $50\ell_{\min}$ uncolored neighbors in K and at most $e(v) \leqslant 20\frac{1+\delta}{1-\delta}\ell_{\min} \leqslant 21\ell_{\min}$ external neighbors). Since $\varphi_{sct}(K) \cap [r] = \emptyset$, each v loses reserved colors only because of external neighbors. Hence, the number of reserved colors available to a vertex v in H' is at least

$$|[r] \cap L_\varphi(v)| \geqslant r - e(v) \geqslant 3 \cdot 75\ell_{\min} \geqslant \deg_\varphi(v; H') + 179\ell_{\min} \ .$$

By Proposition 11.7, SlackColor with $\mathrm{List}(v) = [r]$ and $s(v) = 179\ell_{\min}$ (recall $r = 250\ell_{\min}$, Eq (12.2)) colors all vertices of H' in $O(\mathsf{cd} \log^* n)$ rounds with high probability. Observe that all vertices know r, hence $\mathrm{List}(v)$, because it is fixed in advance.

Coloring Put-Aside Sets (Step 6). The only vertices left to color are put-aside sets. Recall that, in all cabals, we computed a sufficiently large colorful matching. By Proposition 12.13, they can be colored in $O(\mathsf{cd})$ rounds. ∎

12.4. Colorful Matching in Densest Cabals

Recall that a colorful matching is a partial coloring that uses each color twice (within a given almost-clique K). We wish to find a set of anti-edges and to same-color the nodes of each pair.

There are two regimes for computing a colorful matching: when $a(K)$ is $\Omega(\log n)$ and when $a(K)$ is $O(\log n)$. Computing a colorful matching in almost-cliques of average anti-degree $a(K) \in \Omega(\log n)$ can be done as in BCONGEST (see Algorithm 14 in Section 4.6). In non-cabals, this suffices because vertices also get slack from slack generation. In cabals, we begin by running Algorithm 14, and if it returns a colorful matching $M_K \leqslant O(\varepsilon^{-1} \log n)$, we cancel the coloring in K to run this new algorithm. (See proof of Proposition 12.5 in Section 12.3 for more details.) This section therefore focus exclusively on *low anti-degree cabals*, with $a(K) \leqslant O(\log n)$.

Proposition 12.10. *Assume $\Delta \gg \log^2 n$. Suppose all vertices in cabals K with $a(K) \leqslant O(\log n)$ are uncolored. Let φ be the coloring produced by* ColorfulMatchingCabals *in $O(\mathsf{cd}(\log^* n))$ rounds. With high probability, in each cabal K such that $a(K) \leqslant O(\log n)$, for at least $(1-10\varepsilon)\Delta$ vertices $v \in K$ the size of the colorful matching exceeds their anti-degrees $a(v) \leqslant M_K \stackrel{def}{=} |K \cap \operatorname{dom}\varphi| - |\varphi(K)|$. Moreover, the algorithm does not use reserved colors.*

Throughout this section, we use the constant C to represent the value such that $a(K) \leqslant C \log n$, as assumed in Proposition 12.10.

The heart of the algorithm is to find a large matching of anti-edges in K. Using basic routing and MultiColorTrial, they can then be colored in $O(\log^* n)$ rounds. Contrary to the BCONGEST algorithm, we do not find $\Omega(a(K)/\varepsilon)$ anti-edges, but rather enough anti-edges to "satisfy almost all nodes". To make this formal, let us introduce some quantities and notations:

- let $\tau \stackrel{\text{def}}{=} 4\varepsilon$ be the constant fraction at which we divide nodes for the analysis,
- let $\alpha(K)$ be the largest number such that at least $\tau|K|$ nodes in K have anti-degree $\alpha(K)$ or more,
- call a vertex *low* when $a(v) < \alpha(K)$ and *high* otherwise.

We split vertices according to their position in the distribution of anti-degrees within the almost-clique: high vertices represent the top τ-fraction of anti-degrees in the cabal, and low vertices the bottom $(1-\tau)$-fraction. As anti-degrees are non-negative, $\alpha(K)$ cannot be much larger than the mean.

Fact 12.14. $\alpha(K) \leqslant \frac{1}{\tau} a(K) \leqslant (C/\tau)\log n$.

PROOF. $a(K) = \frac{1}{|K|}\sum_{v\in K} a(v) \geqslant \frac{1}{|K|}\sum_{v\in K : a(v)\geqslant\alpha(K)} a(v) \geqslant \tau\cdot\alpha(K)$. ∎

The bulk of the analysis is the following lemma, which we prove later in this section. It states we can find (at least) $\alpha(K)$ anti-edge in each cabal where we run the algorithm.

Lemma 12.15. *Assume $\Delta \gg \varepsilon^{-3}\log n$. With high probability, Algorithm 31 outputs a matching of $\tau\alpha(K)/(4\varepsilon)$ anti-edges.*

Before we dive into the analysis of Algorithm 31, we show that it implies Proposition 12.10.

Coloring the Matching. We show that Algorithm 30 has the effects claimed in Proposition 12.10 and can be implemented as laid out here.

ALGORITHM 30. Computing a Colorful Matching in Cabals with $a(K) \leqslant C\log n$.

Input: A cabal K of low anti-degree $a(K) \leqslant C\log n$ for some (large) constant $C > 0$.
Output: Cabal K has $2M_K \geqslant 2\alpha(K)$ of its nodes properly colored by M_K non-reserved colors.

(1) Run Algorithm 31 and let F_K be the resulting matching and $M_K = |F_K|$.
(2) Each $v \in K$ joins a random group $j \in [M_K]$. Group j assists the j-th non-edge in F_K with performing MultiColorTrial.
(3) Each $f \in F_K$ runs SlackColor with color space $\text{List}(f) = [\Delta+1] \setminus [300\varepsilon\Delta]$.

Proof of Proposition 12.10. Let $K_1, \ldots, K_k$ be the cabals where we run the algorithm. The algorithm has two phases: first, we run Algorithm 31 and find a matching of anti-edges M_i in each K_i; second, we colors them. By Lemmas 12.15 and 12.16, in $O(\mathsf{cd}/\varepsilon)$ rounds, we find an anti-matching F_i of $M_i \geqslant \tau\alpha(K)/(4\varepsilon) = \alpha(K)$ anti-edges in each K with high probability. For each K, at least $(1-\tau)|K| \geqslant (1-10\varepsilon)\Delta$ vertices $v \in K$ have $a(v) \leqslant \alpha(K) \leqslant M_K$, by definition of $\alpha(K)$ and choice of $\tau = 4\varepsilon$.

It remains to explain how we color the anti-edges. We split each cabal K_i into M_i random groups. Since $\Delta \gg (C^2/\tau)\log^2 n \geqslant \alpha(K) \cdot C \log n$ and $M_i = \alpha(K)$ for each cabal K_i. By Lemma 4.9, w.h.p., both endpoints of the j-th anti-edge are adjacent to the j-th group, and the j-th group has diameter 2. This enables efficient communication and coordination between the endpoints of each anti-edge.

Consider MultiColorTrial as implemented by Algorithm 9. It is easily adapted to our setting where it is anti-edges that are trying multiple colors. For each anti-edge, for every call to MultiColorTrial, we let the endpoint of highest ID pick the (pseudo)random set of colors to try. This set is then sent to the other endpoint of the anti-edge using the anti-edge's designated random group. From there, all anti-edge endpoints try the colors from their sets: each endpoint learns which colors from its set are already taken by a neighbor, or are tried in this round by neighbor. Each anti-edge then uses its group to see which colors are available at its two endpoints, and colors itself with such a color if it exists. Thus all operations of Algorithm 9 can be performed by a cabal's discovered anti-edges.

Now, for each $f \in F_i$, let $\text{List}(f) = [\Delta+1] \setminus [300\varepsilon\Delta]$. Each anti-edge is adjacent to $\leqslant 2\varepsilon\Delta + O(\log n) \leqslant 3\varepsilon\Delta$ other anti-edges in $\bigcup_i F_i$; on the

other hand, they have at least $\Delta + 1 - 300\varepsilon\Delta - 2\varepsilon\Delta \geqslant 0.5\Delta$ available colors. Hence, they have slack 0.1Δ and get colored in $O(\log^* n)$ rounds by SlackColor (Proposition 11.7). ∎

Finding the Matching. The suitable matching is found using fingerprinting (Section 11.3). Each node v in the almost-clique samples $k = \Theta(\log n)$ independent geometric random variables $(\mathbf{X}_{v,1}, \mathbf{X}_{v,2}, \ldots, \mathbf{X}_{v,k})$. For each $i \in [k]$, nodes compute the point-wise maximum of the random variables in their neighborhood $\mathbf{Y}_i^v = \max\{\mathbf{X}_{u,i}, u \in N(v)\}$, and the almost-clique computes $\mathbf{Y}_i^K = \max\{\mathbf{X}_{u,i} : u \in K\}$ the point-wise maximum of all the random variables in contains.

Let us focus on the first random variable sampled by each node in K, and the relevant maxima. Suppose the maximum value of the $|K|$ random variables is unique and let $v \in K$ be the unique node in K at which it occurs. All neighbors of v have the same maximum value in their in-clique neighborhood as in the whole almost-clique. Meanwhile, any anti-neighbor of v has a different value for the two, so they all learn that they have an anti-edge with the node that sampled the unique maximum. This observation is the main idea behind how Algorithm 31 finds anti-edges.

Algorithm 31. FingerprintMatching

Input: A cabal K of low anti-degree $a(K) \leqslant C \log n$ for some (large) constant $C > 0$.

Output: An anti-matching of size $M_K \geqslant \Omega(\tau\alpha(K)/\varepsilon)$.

Let $k = \frac{6C \log n}{\varepsilon\tau} \geqslant 6\alpha(K)/\varepsilon$

(1) Each v computes fingerprints $(\mathbf{Y}_i^v)_{i\in[k]}$ and $(\mathbf{Y}_i^{K_v})_{i\in[k]}$.

(2) By BFS in K, identify the set I of indices $i \in [k]$ such that
- the maximum in K is reached at a unique vertex u_i ($\exists! u_i \in K, \mathbf{X}_{u_i,i} = \mathbf{Y}_i^K$),
- a non-edge $\{u_i, v\}$ incident to u_i was detected ($\exists v \in K, \mathbf{Y}_i^v \neq \mathbf{Y}_i^K$), and

- the maximum-value vertex u_i was not a unique maximum in a previous trial ($\forall j < i, \exists u_j \in K \setminus \{u_i\}, \mathbf{X}_{u_i,j} \leqslant \mathbf{X}_{u_j,j}$).

(3) By BFS in K, give each node $v \in K$ with a local identifier $\mathsf{ID}_v \in \{1, \ldots, |K|\}$.
(4) For each $i \in I \subseteq [k]$, let $A_i = \{v \in K : \mathbf{Y}_i^v \neq \mathbf{Y}_i^K\}$, and u_i the one node such that $\mathbf{X}_{u_i,i} = \mathbf{Y}_i^K$.
(5) Each $v \in K$ samples $i_v \in [k]$ uniformly at random and joins the i_v-th random group.
(6) Group i does computation and communication regarding trial $i \in [k]$.
(7) Each node $v \in K$ informs its neighbors whether $v \in A_i$ for each $i \in I \subseteq [k]$.
(8) Group i picks a random $(1/2, |K|)$-min-wise independent hash function h_i.
(9) Group i learns the node w_i of smallest h_i-hash value in A_i, $w_i = \arg\min_{w \in A_i} h_i(\mathsf{ID}(w))$.
(10) Group i broadcasts $\mathsf{ID}(w_i)$ to nodes in A_i .
(11) Discard trials i such that $\exists j : u_i = w_j$.
(12) For each w such that $|\{i : w = w_i\}| > 1$, w picks one such trial i at random, discards others.
(13) Let $I' \subseteq I$ be the set of remaining trial indices.
(14) Output as matching the anti-edges $(u_i w_i)_{i \in I'}$.

Let us refer to the $\Theta(\log n)$ independent cells of our fingerprints as trials. There is a constant probability that the maximum value in each trial is unique (Lemma 11.13), and since all vertices have the same distribution, independent from each other, the unique maximum occurs at a uniform vertex in K (Lemma 11.14). Notably, it occurs at a high vertex with probability τ. Each trial with a unique maximum is an opportunity to sample a new anti-edge between the unique maximum and its anti-neighbors. As earlier trials have already found anti-edges, later trials run the risk of sampling anti-edges sharing endpoint(s) with a previously sampled anti-edge. We cannot add such anti-edges to our set of anti-edges since it would not result in a matching. We show that until enough anti-edges have been discovered,

each trial has a probability $\Omega(\tau)$ of discovering an anti-edge with both endpoints unmatched in ealier trials. Key to our argument is that until enough anti-edges have been sampled, a majority of the high nodes must remain non-sampled and have a majority of their anti-neighbors non-sampled.

In each trial, we need the unique maximum and its anti-neighbors to communicate. We use random groups for this, aided by the fact that $\Delta \gg k \cdot \log n \in \Theta(\varepsilon^{-2} \log^2 n)$. One subtlety is how we sample the anti-neighbor w_i from the set of anti-neighbors A_i for each trial $i \in [k]$. Each node is possibly the anti-neighbor of multiple unique maxima, i.e., a node can be part of many trials as a samplable anti-neighbor. The random group for the i-th trial performs the sampling of w_i by using a min-wise hash function (Definition 2.12). Min-wise hash functions have the property that, when applying a random such function to a set, which element of the set has the minimum hash value roughly follows a uniform distribution over the set.

First, we argue that the algorithm has the claimed runtime. As we use $k = \Theta(\varepsilon^{-2} \log n)$ random groups, we need that $\Delta \gg \varepsilon^{-2} \log^2 n$. We remark nonetheless that the probabilistic argument behind Lemma 12.15 only requires that $\Delta \gg \varepsilon^{-3} \log n$.

Lemma 12.16. *Let k be as in Algorithm 31 and assume $\Delta \gg k \log n$. Algorithm 31 runs in $O(\mathsf{cd}/\varepsilon^2)$ rounds with high probability.*

Proof. From Lemma 11.16, the maxima $\mathbf{Y}_i^v$ and $\mathbf{Y}_i^K$ need $O(k + \log\log n) = O(\varepsilon^{-2} \log n)$ bits to describe in Step 1 with probability $1 - 2^{-\Theta(k)}$. Hence, w.h.p., each vertex learns their values in $O(1/\varepsilon^2)$ rounds by aggregation on support trees.

Step 2 is performed in $O(\mathsf{cd}/\varepsilon^2)$ rounds by simple aggregations over a BFS tree in K. Since there are $O(\varepsilon^{-2} \log n)$ trials, as long as only $O(1)$ bits are needed per trial for the aggregation, the aggregation is possible. Aggregating whether the maximum is unique is done by counting how many nodes satisfy $\mathbf{X}_{v,i} = \mathbf{Y}_i^K$, but with a sum capped at 2. Whether a non-edge was detected is an OR of Boolean values at the nodes. Eliminating trials in which a node is a unique maximum for the second time is done by aggregating the OR of $O(\log n)$-bitmaps in which the nodes indicate which

trial they want to throw away. The same idea is used again in Steps 10 and 11.

In Step 5, w.h.p., each node is adjacent to $\Omega(|K|/\log n)$ nodes in each random group in expectation (Lemma 4.9). By the Chernoff Bound (Lemma 2.3), this holds with high probability. Thus, as every node announces to its neighbors in which trials i it is an eligible anti-neighbor using a $k = O(\varepsilon^{-2} \log n)$ bitmap in Step 7, it is heard by at least one node of every group.

In Step 8, we let the maximum ID node of each group sample a $(1/2, |K|)$-min-wise hash function and send it to its group. By Lemma 2.13, such hash functions with input and output space of size $\Theta(|K|)$ exist that can be described in only $O(\log|K|)$ bits. Here, we use the fact that we equipped nodes in K with identifiers between 1 to $|K|$ in Step 3. Group i computes the ID of the node participating in trial i that hashes to the smallest value by having all nodes in its support trees compute the hashes of their neighbors in A_i, and aggregating the minimum (Step 9). That minimum is then disseminated in all the support trees of group i during Step 10. Each potential anti-neighbor aggregates on its support tree an $O(\log n)$ bitmap of the trials in which it was sampled.

Like Step 10, Steps 11 and 12 are performed by each node using a $k = O(\varepsilon^{-2} \log n)$ bitmap. In Step 10, the bitmap indicates in which trials the node was selected. In Steps 11 and 12, it indicates which trials it opts out of. Aggregating those bitmaps over the whole cabal means every node knows the set of trials that produced an edge for the matching, and every matched node knows to be so and learns the ID of its relevant anti-neighbor from the relevant random group.

The matching computed as output in Step 14 is only distributively known, but at least for each edge of the matching, the random group that was in charge of the trial that produced the edge knows the edge and can inform both endpoints of the other endpoint. ∎

Lemma 12.15. *Assume $\Delta \gg \varepsilon^{-3} \log n$. With high probability, Algorithm 31 outputs a matching of $\tau\alpha(K)/(4\varepsilon)$ anti-edges.*

PROOF. Let us analyze Algorithm 31 trial by trial. As in the algorithm's pseudocode, let $u_i = \perp$ if $i \notin I$ in Step 2 and let otherwise u_i denote the unique maximum in trial $i \in [k]$ and identified by the algorithm in that step. Similarly, when $u_i \neq \perp$, let w_i be the random anti-neighbor of u_i sampled in Step 9. We define the following (random) sets U_i and W_i for the analysis:

$$\mathbf{U}_i \stackrel{\text{def}}{=} \{v \in K : \exists j \leqslant i, u_j = v\} \setminus \{v \in K : \exists j \leqslant i, w_j = v\} ,$$
$$\mathbf{W}_i \stackrel{\text{def}}{=} \{v \in K : \exists j \leqslant i, u \in \mathbf{U}_i, w_j = v \wedge u_j = u\} .$$

Intuitively, $\mathbf{U}_i$ is the set of unique maxima in the first i trials that have not been removed from the set of useful trials in Step 11 by also being a sampled anti-neighbor. The set $\mathbf{W}_i$ contains the anti-neighbors randomly selected by nodes in $\mathbf{U}_i$. Note that $|\mathbf{W}_i|$ never shrinks: in the case where the randomly selected anti-neighbor w_i was already in $\mathbf{W}_{i-1}$, then $|\mathbf{W}_i| = |\mathbf{W}_{i-1}|$ (the sets are without multiplicity); if w_i happens to be a previous unique maximum u_j, then w_j might exit the set $\mathbf{W}_{i-1}$, but $w_i = u_j$ joins it; if w_i is not in $\mathbf{W}_{i-1}$ and was not a previously sampled unique maximum, $\mathbf{W}_i = \mathbf{W}_{i-1} \cup \{w_i\}$ and $|\mathbf{W}_i| = |\mathbf{W}_{i-1}| + 1$.

Each node in $\mathbf{W}_k$ is a guaranteed anti-edge for the matching. Indeed, consider the bipartite graph between nodes of $\mathbf{U}_k$ and $\mathbf{W}_k$, with an edge between $u \in \mathbf{U}_k$ and $w \in \mathbf{W}_k$ if there exists $i \in [k]$ s.t. $u_i = u$ and $w_i = w$. Edges of this graph corresponds to anti-edges in K. It has maximum degree 1 on $\mathbf{U}_k$'s side, and minimum degree 1 on $\mathbf{W}_k$'s side. The former is true because each $u \in K$ may be a u_i for at most one index in I, by the third condition in Step 2. The latter holds because each $w = w_j \in \mathbf{W}_k$ is an anti-neighbor of $u_j \in \mathbf{U}_k$, by definition of W_k. Taking one edge incident on every node of $\mathbf{W}_k$ thus gives a matching of size $|\mathbf{W}_k|$.

We now show that $|\mathbf{W}_k| \geqslant \Omega(\tau\alpha(K)/\varepsilon)$ w.h.p., i.e., Algorithm 31 finds a matching of this size. For completeness, set $\mathbf{W}_0 = \emptyset$ and define $\mathbf{Z}_i$ for all $i \in [k]$ as follows:

- if $|\mathbf{W}_{i-1}| \leqslant (\tau/4) \cdot \alpha(K)/\varepsilon$, then $\mathbf{Z}_i \stackrel{\text{def}}{=} |\mathbf{W}_i| - |\mathbf{W}_{i-1}|$,
- if $|\mathbf{W}_{i-1}| > (\tau/4) \cdot \alpha(K)/\varepsilon$, then $\mathbf{Z}_i \stackrel{\text{def}}{=} 1$.

Observe that $\mathbf{U}_i$, $\mathbf{W}_i$ and thus $\mathbf{Z}_i$ are functions of the random variables $\mathbf{X}_{v,\leqslant i}$. We claim that the lower bound

$$\mathbb{P}[\mathbf{Z}_i = 1 \mid \mathbf{X}_{v,<i} \text{ for all } v \in K] \geqslant \tau/12$$

holds for any conditioning on $\{\mathbf{X}_{v,<i}, v \in K\}$ on previous trials. Before proving it, we explain how it implies our result. By Chernoff with domination (Lemma 2.3), w.h.p., we have $\sum_{i=1}^{k} \mathbf{Z}_i > \tau/24 \cdot k$. By our choice of k and Fact 12.14, it means that

$$\sum_{i=1}^{k} \mathbf{Z}_i \geqslant \tau/24 \cdot k > (\tau/4) \cdot \alpha(K)/\varepsilon \ .$$

Let $i \in [k]$ be the first index such that $\sum_{j\leqslant i} \mathbf{Z}_j \geqslant (\tau/4) \cdot \alpha(K)/\varepsilon$. It must exist because the total sum is strictly larger. By definition, $\mathbf{Z}_j = |\mathbf{W}_j| - |\mathbf{W}_{j-1}|$ for all $j \leqslant i$ and thus $|\mathbf{W}_i| = \sum_{j\leqslant i} \mathbf{Z}_j \geqslant (\tau/4) \cdot \alpha(K)/\varepsilon$. Since the $\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_k$ are non-decreasing, we found a large matching $|\mathbf{W}_k| \geqslant |\mathbf{W}_i| \geqslant (\tau/4) \cdot \alpha(K)/\varepsilon$.

When $\mathbf{X}_{v,<i}$ is such that $|\mathbf{W}_{i-1}| > (\tau/4) \cdot \alpha(K)/\varepsilon$, we get $\mathbb{P}[\mathbf{Z}_i = 1 \mid \mathbf{X}_{v,<i}] = 1 \geqslant \tau/12$. We assume henceforth that $\mathbf{W}_{i-1} = W_{i-1}$ such that $|W_{i-1}| \leqslant (\tau/4) \cdot \alpha(K)/\varepsilon$.

Consider the anti-edges incident on the nodes in $|W_{i-1}|$, the total of their incidences is $\sum_{v\in W_{i-1}} a_v \leqslant |W_{i-1}| \cdot \varepsilon\Delta \leqslant (\tau/4) \cdot \alpha(K)\Delta$. From the number of anti-edges incident to the nodes of W_{i-1}, we get that at most $(\tau/2)\Delta$ nodes have $\alpha(K)/2$ or more anti-neighbors in W_{i-1}. Since high nodes have anti-degree at least $\alpha(K)$, this means that at most $(\tau/2)\Delta$ high nodes can have half or more of their neighborhood in W_{i-1}. Since there are at least $\tau|K|$ high nodes, and at most $2k$ of them were sampled in U_{i-1} or W_{i-1} by previous trials, at least $\tau|K| - (\tau/2)\Delta - 2k \geqslant \tau|K|/3$ non-previously sampled high nodes have more than half their anti-neighbors outside W_{i-1}. If the i-th trial sampled such a u_i, it would join U_i, thereby increasing its size by one.

Conditioned on the existence of a unique maximum, this maximum occurs at a non-previously sampled high node with a majority of anti-neighbors outside W_{i-1} with probability at least $\tau/3$. The probability of the maximum being unique is at least $2/3$ (Lemma 11.13), and the probability of the random anti-neighbor being outside W_{i-1} is at least $1/4$, where the

probability is smaller than $1/2$ from the parameter chosen for min-hashing (Lemma 2.13). Selecting a random anti-neighbor outside W_{i-1} grows W_{i-1} by a new node, meaning $\mathbf{Z}_i = |\mathbf{W}_i| - |W_{i-1}| = 1$ with probability at least $\tau/3 \cdot 2/3 \cdot 1/4 = \tau/12$. ∎

12.5. Coloring Put-Aside Sets

Proposition 12.13. *Suppose φ is a coloring such that only put-aside sets are uncolored and at least 0.9Δ vertices in each cabal verify $a(v) \leqslant M_K$. Then there is a $O(\mathsf{cd})$-round algorithm that computes a total coloring of H, with high probability.*

This section focuses on coloring put-aside sets in cabals, which we recall are extremely dense almost-cliques where $\tilde{e}_K < \ell_{\min}$. In Section 12.5.1, we describe the full algorithm as well as the formal guarantees given by each intermediate step. Details for intermediate steps are given in subsequent subsections.

12.5.1. Proof of Proposition 12.13. Let φ be the partial coloring produced by the algorithm before we color put-aside sets. Crucially, it is not adversarial in cabals. The only uncolored vertices are the put-aside sets, i.e., $V \setminus \operatorname{dom}\varphi = \bigcup_K P_K$. Proposition 3.24 ensures that each $P_K \subseteq I_K$ has size $r = \Theta(\ell_{\min})$ and all inliers verify $e(v) \leqslant 25\ell_{\min}$. From Lemma 11.9 and Proposition 12.10, we know there exist 0.9Δ vertices in K such that $a(v) \leqslant M_K$ (although vertices do not know if they verify it). Henceforth, we refer to uncolored vertices of K as $u_{K,1}, \ldots, u_{K,r}$. Each uncolored vertex learns its index by computing prefix sums on a BFS tree spanning K (Lemma 11.2). Note that each vertex knows r.

Parameters. We split the color space into contiguous *blocks* of b colors each, defining:

$$\begin{aligned} &\ell_s \stackrel{\text{def}}{=} \Theta(\ell_{\min}^3)\ , \quad b \stackrel{\text{def}}{=} 256\ell_s^6\ , \quad \text{and} \\ &B_i \stackrel{\text{def}}{=} \{(i-1)b+1, \ldots, ib\} \quad \text{for all} \quad i \in \left[\frac{\Delta+1}{b}\right]\ . \end{aligned} \tag{12.7}$$

ALGORITHM 32. COLORPUTASIDE

Input: The coloring φ as described in Proposition 12.13

Output: A total coloring φ_{total}

(1) **if** $|L_\varphi(K)| \geqslant \ell_s$ **then** TRYFREECOLORS
(2) **else**
  (a) FINDCANDIDATEDONORS (Section 12.5.2)
  (b) FINDSAFEDONORS (Section 12.5.3)
  (c) DONATECOLORS

Use Free Colors If Any (Step 1). We say a color is

- *unique* in K if exactly one vertex in K uses it and
- *free* in K if it is not used by colored vertices in K (i.e., it is in the clique palette).

Although there should be almost no free colors at this point of the algorithm, we cannot ensure there are none. It simplifies subsequent steps to assume there are few free colors. Hence, before running the donation algorithm, vertices count the number of colors $|L_\varphi(K)|$ in the clique palette using the query algorithm. If there are fewer than ℓ_s free colors, they go to Step 2a of Algorithm 32. Otherwise, they can all get colored in $O(\mathsf{cd})$ rounds with high probability as we now explain.

Suppose K has at least ℓ_s free colors. Vertices of K find a hash function $h_K : [\Delta + 1] \to [\text{poly}(\log n)]$ that does not collide on the ℓ_s smallest colors of $L_\varphi(K)$. This can be done using, say, random groups. We defer details to Lemma A.7 to preserve the flow of the chapter. Such a hash function can be represented in $O(\log \log n)$ bits and each hash takes $O(\log \log n)$ bits. Each $u_{K,i}$ samples $k = \Theta\left(\frac{\log n}{\log \log n}\right)$ indices in $\{1, 2, \ldots, \ell_s\}$ with replacement. They learn the hashes of the corresponding colors in $L_\varphi(K)$ with a straightforward adaption of the query algorithm (Lemma 11.3). Overall, the message with the hashes of the sampled colors takes $k \cdot O(\log \log n) = O(\log n)$ bits. A hash is safe iff it does not collide with $h(\varphi(E(u_{K,i})))$ the hashes of external neighbors nor with those

of other put-aside vertices. Since h_K is collision free on the ℓ_s smallest colors of $L_\varphi(K)$, each hash sampled by $u_{K,i}$ is uniform in a set of $|h(L_\varphi(K))| = |L(K)| = \ell_s$ colors. As $P_K \subseteq I_K$, external neighbors block $\leqslant 25\ell_{\min}$ hashes (recall external neighbors are not put-aside vertices). Put-aside vertices $u_{K,\neg i}$ from the same cabal block $kr \leqslant \ell_{\min}^2$ hashes. By union bound, a hash is unsafe with probability $\Theta(\ell_{\min}^2)/\ell_s \leqslant 1/\log n$ (because $\ell_s = \Theta(\ell_{\min}^3)$). Thus, the probability that none of the hashes sampled by $u_{K,i}$ are safe is $(1/\log n)^k = 2^{-k \log\log n} \leqslant 1/\mathrm{poly}(n)$. By union bound, all uncolored vertices of K find a safe hash with high probability. Using the query algorithm, they then learn which color of (the smallest ℓ_s colors of) $L(K)$ hashes to that value in $O(\mathsf{cd})$ rounds and get colored.

Remark 12.17. Henceforth, we assume that the number of free or repeated colors in K is at most $3\ell_s$. Indeed, we may assume that $|K| < \Delta + 1 + \ell_s$ as otherwise the colorful matching has size $\Omega(a(K)/\varepsilon) \gg \ell_s$, and the clique palette $L_\varphi(K)$ has at least ℓ_s free colors (by Lemma 4.19 for v with $a(v) \leqslant a(K)$). If the number of repeated colors is at least $2\ell_s$, there are at least ℓ_s free colors since $|L_\varphi(K)| \geqslant \Delta + 1 - |K| + |P_K| + 2\ell_s \geqslant \ell_s$ from the assumption that $|\Delta| < \Delta + 1 - \ell_s$.

Finding Candidate Donors (Step 2a). Henceforth, we assume $|L_\varphi(K)| < \ell_s$, i.e., there are few free colors. Being unique is necessary for a color to be donated, but it is not sufficient. FindCandidateDonors (Algorithm 33) computes in each cabal a set Q_K of candidate donors. A candidate donor holds a unique color in its cabal and is not adjacent to candidate donors or put-aside vertices of other cabals. Naturally, we look for many candidates as it increases the chances of finding a donation later.

Lemma 12.18. *After* FindCandidateDonors *(Algorithm 33), w.h.p., for each cabal K where $|L_\varphi(K)| < \ell_s$, the algorithm computes sets $Q_K \subseteq I_K$ of inliers such that*

(1) for each $v \in Q_K$, the color $\varphi(v)$ is unique *in K and at least $\frac{\Delta+1}{b} \cdot 8\ell_s^3$ of them have $a(v) \leqslant M_K$;*
(2) no edge connects Q_K to $P_{\neg K} \cup Q_{\neg K}$.

The algorithm ends after $O(\mathsf{cd})$ rounds and each vertex $v \in K$ knows if $v \in Q_K$.

Finding Safe Donations (Step 2b). FindSafeDonors (Algorithm 34) provides each $u_{K,i}$ with a large set $S_{K,i}$ of *safe donors*, all of which can be recolored using the *same* replacement color $\chi^{\mathsf{recol}}_{K,i}$, which is different from all $\chi^{\mathsf{recol}}_{K,\neg i}$. We emphasize that the donation is safe for the donor (the vertex in $S_{K,i}$) but that $u_{K,i}$ might reject it because the donated color is used by one of its external neighbors. Donors are safe as the replacement color is in their palette (Part 2) and not used as a replacement by donors from another group (Part 1). All donations for $u_{K,i}$ come from the same block (Part 3) so that they can be described with few bits. Finding many safe donors (Part 4) ensures that a random donation is likely to work (see Step 2c).

Lemma 12.19. *After* FindSafeDonors *(Algorithm 34), w.h.p., for each cabal K where $|L_\varphi(K)| < \ell_s$, for each $i \in [r]$, there exists a triplet $(\chi^{\mathsf{recol}}_{K,i}, j_{K,i}, S_{K,i})$ where $\chi^{\mathsf{recol}}_{K,i} \in L_\varphi(K)$, $j_i \in \left[\frac{\Delta+1}{b}\right]$ and $S_{K,i} \subseteq Q_K$ such that*

(1) each $\chi^{\mathsf{recol}}_{K,i} \neq \chi^{\mathsf{recol}}_{K,i'}$ for any $i' \in [r] \setminus \{i\}$ and sets $S_{K,i}$ are pairwise disjoint,
(2) each $v \in S_{K,i}$ has $\chi^{\mathsf{recol}}_{K,i} \in L_\varphi(v)$,
(3) each $v \in S_{K,i}$ has $\varphi(v) \in B_{j_i}$, and
(4) $|S_{K,i}| = \ell_s$.

The algorithm ends in $O(\mathsf{cd})$ rounds. Each vertex knows if $v \in S_{K,i}$ for some $i \in [r]$ and, if so, knows $\chi^{\mathsf{recol}}_{K,i}$. Each uncolored vertex $u_{K,i}$ knows j_i.

Donating Colors (Step 2c). Each uncolored vertex $u_{K,i}$ samples $k = \Theta(\frac{\log n}{\log\log n})$ colors with replacement in its set $\varphi(S_{K,i})$ of safe donations. We claim it samples some $\chi^{\mathsf{don}}_{K,i} \notin \varphi(E(u_{K,i}))$, i.e., a color unused by external neighbors. Indeed, recall that $u_{K,i}$ has at most $e(v) \leqslant 25\ell_{\min}$ external neighbors (by definition of inliers). As each color in $\varphi(S_{K,i})$ is unique, it contains $|\varphi(S_{K,i})| = |S_{K,i}| \geqslant \ell_s$ colors. By union bound, a uniform color in $\varphi(S_{K,i})$ conflicts with an external neighbor of $u_{K,i}$ with probability at most $25\ell_{\min}/\ell_s \leqslant 1/\log n$ (because $\ell_s \gg \ell^3_{\min}$, Eq (12.7)). Thus, the probability that all k samples hit a bad color is at most $(1/\log n)^k = 2^{-k\log\log n} \leqslant 1/\mathrm{poly}(n)$. By union bound, w.h.p., each uncolored $u_{K,i}$ finds $\chi^{\mathsf{don}}_{K,i}$ used by

a donor in $S_{K,i}$. We argue that the following coloring is total and proper

$$\varphi_{total}(w) \stackrel{\text{def}}{=} \begin{cases} \chi^{\mathsf{don}}_{K,i} & \text{when } w = u_{K,i} \\ \chi^{\mathsf{recol}}_{K,i} & \text{when } w \in S_{K,i} \text{ and } \chi^{\mathsf{don}}_{K,i} = \varphi(w) \\ \varphi(w) & \text{otherwise} \end{cases} .$$

Each uncolored vertex $u_{K,i}$ gets colored with $\chi^{\mathsf{don}}_{K,i}$. They are properly colored because 1) no external neighbors gets colored nor recolored (Proposition 3.24, Part 2 and Lemma 12.18, Part 2), 2) no put-aside neighbors in K gets the same color $\chi^{\mathsf{don}}_{K,i} \in \varphi(S_{K,i})$ since sets $S_{K,*}$ are disjoint (Lemma 12.19, Part 1) and contain only unique colors ($S_{K,i} \subseteq Q_K$ and Lemma 12.18, Part 1), 3) color $\chi^{\mathsf{don}}_{K,i}$ is not used in any recoloring in K (because it is not free), and 4) at most one vertex in K was colored $\chi^{\mathsf{don}}_{K,i}$, which recolor itself using some other color (Lemma 12.18, Part 1).

Let $v \in S_{K,i}$ be the vertex donating its color to $u_{K,i}$; recall it is recolored with $\chi^{\mathsf{recol}}_{K,i}$. It is properly colored because 1) none of its external neighbors gets colored nor recolored (Lemma 12.18, Part 2), 2) as explained before, it does not conflict with any $u_{K,*}$ getting colored, 3) it does not conflict with any recoloring in K as $\chi^{\mathsf{recol}}_{K,*}$ are all different (Lemma 12.19, Part 1), and 4) it does not conflict with other colored neighbors (in K or outside) as $\chi^{\mathsf{recol}}_{K,i} \in L_\varphi(v)$ (Lemma 12.19, Part 2).

Implementing Step 2c. Implementations of FindCandidateDonors and FindSafeDonors guarantee that a vertex knows when it belongs to sets $Q_{K,i}$ and $S_{K,i}$ for some $i \in [r]$. Moreover each $u_{K,i}$ knows the block j_i.

To sample colors in $\varphi(S_{K,i})$, uncolored vertices sample k indices in $\{1, 2, \ldots, |\varphi(S_{K,i})|\}$. To learn $|\varphi(S_{K,i})|$, we partition K into r random groups. Compute BFS trees $T_1, \ldots, T_r$ where each T_i spans the i-th random group and $S_{K,i}$. Observe that a vertex belongs to at most two trees (one for its random group, and possibly an other if it belongs to $S_{K,*}$). Hence, congestion on trees T_i causes only $O(1)$ overhead. By aggregation on trees T_i, for all $i \in [r]$ in parallel, we count $|S_{K,i}| = |\varphi(S_{K,i})|$ exactly in $O(\mathsf{cd})$ rounds.

After sampling indices in $\{1, 2, \ldots, |\varphi(S_{K,i})|\}$, each $u_{K,i}$ needs to learn the colors they correspond to. We claim the list of sampled donations can

be represented by a $O(\log n)$-bit message. Indeed, since $\varphi(S_{K,i}) \subseteq B_{j_i}$ for each i, to encode k colors, it suffices to represent each sampled color by its offset in B_{j_i}. Overall, it uses $k \cdot O(\log b) = O(\log n)$ bits.

To check for collisions with external neighbors, $u_{K,i}$ uses the same representation but also includes the index of the block j_i (which is not known to external neighbors). Then it broadcasts this $O(\log n)$-bit message to external neighbors. They respond, say, with a bitmap describing which sampled colors conflict with $\varphi(E(u_{K,i}))$.

12.5.2. Finding Candidate Donors. The algorithm first filters out vertices with external neighbors in put-aside sets, the remaining vertices join the set Q_K^{pre}. It then *activates* vertices independently with probability $\Theta(\ell_s^3/b) = \Theta(1/\ell_s^3)$, in a set Q_K^{active}. It retains the active vertices that have unique colors and have no external active neighbors, resulting in Q_K.

As Part 2 of Lemma 12.18 is direct from Algorithm 33, the focus of this section is on proving Part 1. We first prove the algorithm is correct and conclude with its implementation on cluster graphs.

ALGORITHM 33. FINDCANDIDATEDONORS

Input: The coloring φ and the integer N

Output: Sets $Q_K \subseteq I_K$

(1) Let Q_K^{pre} be uncolored inliers of K with no external neighbor in a put-aside set.

$$Q_K^{\mathsf{pre}} = \left\{ u \in I_K : E(v) \cap \left(\bigcup_{K' \neq K} P_{K'} \right) = \emptyset \right\}.$$

(2) Each $v \in Q_K^{\mathsf{pre}}$ joins Q_K^{active} w.p. $p = \frac{60\ell_s^3}{b}$.

(3) Let Q_K be the vertices of Q_K^{active} with no active external neighbors.

$$Q_K = \left\{ v \in Q_K^{\mathsf{active}} : \varphi(u) \text{ is unique} \quad \text{and} \quad N(v) \cap \left(\bigcup_{K' \neq K} Q_{K'}^{\mathsf{active}} \right) = \emptyset \right\}$$

Lemma 12.20. *At the end of Algorithm 33, w.h.p, Part 1 of Lemma 12.18 holds for each cabal.*

PROOF. We call a color *good* if and only if *i*) it is unique, and *ii*) used by a vertex in I_K with $a(v) \leqslant M_K$ and no external neighbors in a put-aside set. By extension, a vertex v is good if and only if $\varphi(v)$ is good. Observe that a good vertex is in Q_K^{pre} and it joins Q_K iff it becomes active in Step 2 while none of its $O(\ell_{\min})$ external neighbors do.

First, we claim that each cabal contains at least 0.75Δ good vertices. There are at most $3\ell_s$ colors that are not unique in K (Remark 12.17). By assumption on I_K, we drop at most $(0.1+\varepsilon)\Delta$ colors because they are used by vertices in $K \setminus I_K$ (Lemma 12.11). By Lemma 12.12, Part 3, at most $0.01|K| \leqslant (0.01+\varepsilon)\Delta$ inliers have an external neighbor in some $P_{K'}$ for $K' \neq K$. Overall, the number of good colors in $[\Delta+1]$ is at least $(\Delta+1) - (0.1+\varepsilon)\Delta - (0.01+\varepsilon)|K| - 3\ell_s \geqslant 3/4 \cdot \Delta$.

Steps 2 and 3 are the same as in LOWDEGREESAMPLE (Algorithm 6 in Section 3.5.2) where S_i is the set of good vertices in the i-th cabal. External degrees are at most $25\ell_{\min} \leqslant 1/(10p) = \Theta(\ell_{\min}^9)$ and $|S_i| \geqslant 3\Delta_{low}/4 \gg p^{-2}\log n = \Theta(\ell_{\min}^{18.2})$ from our choice of parameters (Eq (12.7)). By Lemma 3.26, we have that each Q_K^{active} has size at least $|S_i|p/5 \geqslant \frac{\Delta+1}{b} \cdot 8\ell_s^3$. ∎

Lemma 12.18. *After* FINDCANDIDATEDONORS *(Algorithm 33), w.h.p., for each cabal K where $|L_\varphi(K)| < \ell_s$, the algorithm computes sets $Q_K \subseteq I_K$ of inliers such that*

(1) for each $v \in Q_K$, the color $\varphi(v)$ is unique *in K and at least $\frac{\Delta+1}{b} \cdot 8\ell_s^3$ of them have $a(v) \leqslant M_K$;*
(2) no edge connects Q_K to $P_{\neg K} \cup Q_{\neg K}$.

The algorithm ends after $O(\mathsf{cd})$ rounds and each vertex $v \in K$ knows if $v \in Q_K$.

PROOF. Part 2 holds by construction of Q_K (Steps 2 and 3). Part 1 holds with high probability by Lemma 12.20.

Implementation-wise, the only non-trivial step in Algorithm 33 is testing if a color is unique in Step 3. Crucially, we only need to test the uniqueness of colors for active vertices. Since each vertex becomes active independently, w.h.p., each cabal contains at most $2(1+\varepsilon)\Delta p \ll \Delta/\Theta(\log^2 n)$ active vertices (recall $b \gg \ell_s^3 \log n$, Eq (12.7)). Hence, we can partition K into $|Q_K^{\mathsf{active}}|$ random groups, where the i-th group tests the uniqueness of $\varphi(v)$ where v is the i-th vertex in Q_K^{active}. Observe we can count the size of Q_K^{active} and order active vertices using the prefix sum algorithm (Lemma 11.2). A group tests if its attributed color χ is unique in two aggregations: first, they compute the smallest identifier of a vertex colored χ in K; then, they compute the next smallest identifier for a vertex colored χ. If they can find two different identifiers, then the color is not unique. They never misclassify a color since each random group is adjacent to all vertices of K (Lemma 4.9). Each group broadcasts its findings and vertices learn if their color is unique. ∎

12.5.3. Finding Safe Donors. Focus on cabal K, let $u_1, \ldots, u_r$ be the uncolored vertices, and let Q be the set of inliers returned by FindCandidateDonors. We show that if each vertex in Q samples a uniform color in the clique palette, at least $|P_K| = r$ colors are sampled by at least ℓ_s vertices from the same block. Throughout this section, for block j, let

$$\mathscr{C}(j) \stackrel{\text{def}}{=} Q \cap \varphi^{-1}(B_j) = \{v \in Q : \varphi(v) \in B_j\}$$

be the vertices of Q colored with colors from the j-th block.

Lemma 12.21. *If each vertex in Q samples a uniform color $\boldsymbol{\chi}(v) \in L_\varphi(K)$, then w.h.p., there exist r colors $\chi_1, \chi_2, \ldots, \chi_r \in L_\varphi(K)$ and r block indices $j_1, j_2, \ldots, j_r$ such that the number of vertices in $\mathscr{C}(j_i)$ that sampled χ_i is*

$$|\{v \in \mathscr{C}(j_i) : \boldsymbol{\chi}(v) = \chi_i \in L_\varphi(v)\}| \geqslant 4\ell_s \ . \tag{12.8}$$

Proof. Call a vertex $v \in Q$ *good* iff $a(v) \leqslant M_K$ and denote by $\mathscr{G} = \{v \in Q : a(v) \leqslant M_K\}$ the set of good vertices in K. We define a color $\chi \in L_\varphi(K)$ in the clique palette as *happy* iff χ is available to at least a $1/\ell_s$-fraction of the *good* vertices

$$|\{v \in \mathscr{G} : \chi \in L_\varphi(v)\}| \geqslant \frac{|\mathscr{G}|}{\ell_s} \ .$$

As we shall see, happy colors are likely to be sampled by many vertices.

First, we claim there exist at least r happy colors in the clique palette. Recall, by Lemma 4.19, that good vertices always have at least $|L_\varphi(v) \cap L_\varphi(K)| \geqslant |K \setminus \operatorname{dom}\varphi| = |P_K| = r$ colors available in the clique palette. Hence, if h is the number of happy colors, then

$$\begin{aligned} r &\leqslant \frac{1}{|\mathscr{G}|}\sum_{v\in\mathscr{G}} |L_\varphi(v)\cap L_\varphi(K)| \\ &= \frac{1}{|\mathscr{G}|}\sum_{\chi\in L_\varphi(K)} |\{v\in\mathscr{G} : \chi\in L_\varphi(v)\}| \qquad (12.9) \\ &< h + \frac{|L_\varphi(K)|}{\ell_s}\ , \end{aligned}$$

where the last inequality comes from the fact that each happy color can contribute to all palettes, while the $\leqslant |L_\varphi(K)|$ unhappy colors contribute to fewer than $|\mathscr{G}|/\ell_s$ palettes, by definition. Since we assumed that $|L_\varphi(K)| < \ell_s$, unhappy colors contribute very little and Eq (12.9) simplifies to $r < h+1$. Because r and h are integral, the number of happy colors is $h \geqslant r$.

We now argue that for each happy color χ, we can find a block j such that Eq (12.8) holds. By Part 1 in Lemma 12.18, there are at least $|\mathscr{G}| \geqslant \frac{\Delta+1}{b}\cdot 8\ell_s^3$ good vertices in K. This implies that there exists a block containing $\geqslant 8\ell_s^2$ vertices with χ in their palettes, as otherwise the total number of good vertices with χ in their palette would be $< \frac{\Delta+1}{b}\cdot 8\ell_s^2 < |\mathscr{G}|/\ell_s$ contradicting χ being happy. Let $j \in \left[\frac{\Delta+1}{b}\right]$ be the index of a block such that $\geqslant 8\ell_s^2$ vertices in $\mathscr{C}(j)$ have χ in their palette.

Since each vertex $v \in \mathscr{C}(j)$ samples $\boldsymbol{\chi}(v) = \chi$ independently with probability $1/|L_\varphi(K)| \geqslant 1/\ell_s$, we expect at least $8\ell_s$ of them to sample $\boldsymbol{\chi}(v) = \chi$. By Chernoff, w.h.p., at least $4\ell_s$ vertices in $\mathscr{C}(j)$ sample χ, which shows Eq (12.8) for happy color χ. By union bound, it holds with high probability for all happy colors in all cabals. ∎

Lemma 12.21 shows that after vertices sample a random color in the clique palette, it suffices to *detect* which colors χ have a block j such that $\mathscr{C}(j)$ contains many vertices that sampled χ. We achieve this through random groups and fingerprinting. Algorithm 34 gives the main steps and implementation details are in the proof of Lemma 12.19.

ALGORITHM 34. FINDSAFEDONORS

Input: The coloring φ and sets $Q_K \subseteq K$ as given by Step 2a.

Output: Sets $S_{K,i} \subseteq Q_K$, blocks j_i and colors $\chi_i^{\mathsf{recol}} \in L_\varphi(K)$ as described in Lemma 12.19

(1) Each $v \in Q_K$ samples a uniform color $\boldsymbol{\chi}(v) \in L_\varphi(K)$ *and drop it if* $\boldsymbol{\chi}(v) \notin L_\varphi(v)$.
(2) Partition K into $|L_\varphi(K)| \cdot \frac{\Delta+1}{b}$ random groups indexed as $(\chi, j) \in L_\varphi(K) \times \left[\frac{\Delta+1}{b}\right]$.
Group (χ, j) estimates the number of vertices in the j-th block that sampled χ as replacement

$$\beta_{\chi,j} \in (1 \pm 0.5)|\{v \in \mathscr{C}(j) : \boldsymbol{\chi}(v) = \chi \in L_\varphi(v)\}| \ .$$

(3) For each $\chi \in L_\varphi(K)$, find $j_\chi \in \left[\frac{\Delta+1}{b}\right]$ for which $\beta_{\chi,j} > 2\ell_s$. If no such j_χ exist, set $j_\chi = \perp$.
(4) Select r colors $\chi_1, \dots, \chi_r$ for which $j_\chi \neq \perp$
Return $\chi_{K,i}^{\mathsf{recol}} = \chi_i$ with $j_i = j_{\chi_i}$ and $S_{K,i} = \{v \in \mathscr{C}(j) : \boldsymbol{\chi}(v) = \chi_i \in L_\varphi(v)\}$

Lemma 12.19. *After* FINDSAFEDONORS *(Algorithm 34), w.h.p., for each cabal K where $|L_\varphi(K)| < \ell_s$, for each $i \in [r]$, there exists a triplet $(\chi_{K,i}^{\mathsf{recol}}, j_{K,i}, S_{K,i})$ where $\chi_{K,i}^{\mathsf{recol}} \in L_\varphi(K)$, $j_i \in \left[\frac{\Delta+1}{b}\right]$ and $S_{K,i} \subseteq Q_K$ such that*

(1) each $\chi_{K,i}^{\mathsf{recol}} \neq \chi_{K,i'}^{\mathsf{recol}}$ for any $i' \in [r] \setminus \{i\}$ and sets $S_{K,i}$ are pairwise disjoint,
(2) each $v \in S_{K,i}$ has $\chi_{K,i}^{\mathsf{recol}} \in L_\varphi(v)$,
(3) each $v \in S_{K,i}$ has $\varphi(v) \in B_{j_i}$, and
(4) $|S_{K,i}| = \ell_s$.

The algorithm ends in $O(\mathsf{cd})$ rounds. Each vertex knows if $v \in S_{K,i}$ for some $i \in [r]$ and, if so, knows $\chi_{K,i}^{\mathsf{recol}}$. Each uncolored vertex $u_{K,i}$ knows j_i.

PROOF. We go over steps of Algorithm 34.

Sampling (Step 1). Sampling a uniform color in the clique palette takes $O(\mathsf{cd})$ rounds using the query algorithm (Lemma 11.3). Each vertex checks if $\boldsymbol{\chi}(v) \in L_\varphi(v)$ with a $O(\mathsf{cd})$ round broadcast and bit-wise OR.

Fingerprinting (Step 2). By Lemma 4.9, w.h.p., all vertices are adjacent to at least one vertex from each group. Using the fingerprinting algorithm (Lemma 11.17), vertices of group (χ, j) aggregate fingerprints from $\mathscr{C}(j)$ and compute $\beta_{\chi,i}$ in $O(\mathsf{cd})$ rounds.

Selecting Blocks (Step 3). For each color $\chi \in L_\varphi(K)$, build a BFS trees T_χ spanning random group for that colors $\{(\chi, j), j \in \left[\frac{\Delta+1}{b}\right]\}$. Each group sets a bit $x_{\chi,j}$ to one if $\beta_{\chi,j} > 2\ell_s$ and zero otherwise. Use the prefix sum algorithm over T_χ (Lemma 11.2) to count for each group (χ, j) the prefix sum $\sum_{j' \prec j} x_{\chi,j}$. Let j_χ be the index for which this prefix sum is exactly one, i.e., the block of smallest index with enough vertices to sample $\boldsymbol{\chi}(v) = \chi$ in Step 1. If none exists, then $j_\chi = \perp$. This information is disseminated to all vertices in groups $\{(\chi, j), j \in \left[\frac{\Delta+1}{b}\right]\}$ by converge/broadcast on T_χ. Note that since random groups are disjoint, the trees T_χ for different colors are disjoint as well. In particular, the algorithm runs independently for each color without creating congestion, thus computing blocks j_χ for all $\chi \in L_\varphi(K)$ in parallel.

Output (Step 4). Similarly, using the prefix sum algorithm on one tree spanning K, we find the first r colors $\chi_1, \ldots, \chi_r$ for which such a block exist. For each $i \in [r]$, the algorithm returns $\chi_{K,i}^{\mathsf{recol}} \stackrel{\text{def}}{=} \chi_i$ and $S_{K,i}$ the vertices in $\mathscr{C}(j_{\chi_i})$ that sampled χ_i. Each random group broadcasts a message if it was selected, thus each vertex knows if it belongs to $S_{K,i}$ for some $i \in [r]$.

Correctness. By Lemma 12.21, w.h.p., there exist at least r colors for which Eq (12.8) holds. As $\beta_{\chi,j}$ is a 2-approximation of the left-hand side in Eq (12.8), there exist at least r pairs (χ, j) such that $\beta_{\chi,j} > 2\ell_s$. Hence, the algorithm always finds r colors in Step 4.

Clearly, the colors $\chi_1, \ldots, \chi_r$ selected by the algorithm are different and the sets $S_{K,i}$ are disjoint (since each vertex samples exactly one color). Since a vertex retains the sampled color only if it belongs to its palette, Part 2 holds. Part 3 is clear from the definition of $S_{K,i}$ as the subset of one block. Finally, Part 4 holds because when $\beta_{i,\chi} > 2\ell_s$, then at least ℓ_s vertices from block i sampled color χ. ∎

12.6. Preparing SlackColor in Non-Cabals

Proposition 12.9. *Let φ be a coloring such that reserved colors are unused ($[r_K] \cap \varphi(K) = \emptyset$ in all $K \notin \mathscr{K}_{cabal}$), Eq (12.6) holds, and vertices have uncolored degree $O(e(K))$. There is an algorithm that extends φ to $V_{dense} \setminus V_{cabal}$ in $O(\mathsf{cd} \log^* n)$ rounds with high probability.*

If a vertex does not have slack in $[r_v]$, it must have many available colors in $L(K_v) \setminus [r_v]$. In particular, if it tries a random color in $L(K_v) \setminus [r_v]$, it gets colored with constant probability. Trying random colors can be used to reduce the number of nodes without slack in $[r_v]$ to a small fraction of r_v so as to color them using reserved colors, for only $e(K) \ll r_K$ external neighbors can block colors.

The key technical challenge when implementing this on cluster graphs is to determine *when* a vertex has slack in reserved colors. Vertices cannot count the number of available colors (i.e., in $|L(v) \cap L(K_v) \setminus [r_v]|$) exactly. We rather (approximately) count the number of *nodes* using colors in $[\Delta + 1] \setminus [r_v]$.

Define $\mu_v^K(\chi) = |\varphi^{-1}(\chi) \cap K_v|$ as the number of nodes in K_v with color χ, and similarly $\mu_v^e(\chi) = |\varphi^{-1}(\chi) \cap E(v)|$. For each vertex v, we estimate $|L(v) \cap L(K) \setminus [r_v]|$ by counting the difference between the number of colors (i.e., the $\Delta + 1 - r_v$ term) and the number of nodes with non-reserved colors (i.e., the two sums)

$$z_v \stackrel{\text{def}}{=} \left((\Delta + 1 - r_v) - \sum_{c=r_v+1}^{\Delta+1} \mu_v^K(c) - \sum_{\chi=r_v+1}^{\Delta+1} \mu_v^e(\chi) \right) + \gamma_{12.7} \cdot e(K) + 40a(K) + x_v \ . \tag{12.10}$$

The last three terms account for the repeated non-reserved colors we expect for v (Lemma 12.7). We emphasize that z_v depends implicitly on the current coloring. Lemma 12.22 shows that z_v lower bounds the number of non-reserved colors in the clique palette that are available for v.

Lemma 12.22. *The number of non-reserved colors available to a dense non-cabal inlier $v \in I_{K_v} \subseteq K_v \notin \mathscr{K}_{cabal}$ is at least*

$$|L_\varphi(v) \cap L_\varphi(K_v) \setminus [r_v]| \geqslant z_v \ .$$

PROOF. Fix a node $v \in I_{K_v}$ for $K_v \notin \mathscr{K}_{cabal}$. Since $\mu_v^K(\chi) + \mu_v^e(\chi)$ is the number of nodes in $K_v \cup E(v)$ using color χ, when that color is used by at least one vertex, $\mu_v^K(\chi) + \mu_v^e(\chi) - 1$ count the contribution of χ-colored nodes to the slack of v. Hence, Eq (12.6) can be rewritten in terms of μ_v^K and μ_v^e as

$$\sum_{\substack{\chi \in [\Delta+1] \setminus [r_v]: \\ \mu_v^K(\chi)+\mu_v^e(\chi) \geqslant 1}} (\mu_v^K(\chi) + \mu_v^e(\chi) - 1) \geqslant \gamma_{12.7} \cdot e(K) + 40a(K) + x_v \ , \tag{12.11}$$

where we emphasize that the sum is only over non-reserved colors. Plugging Eq (12.11) into Eq (12.10), we upper bound z_v by the following complicated sum

$$z_v \leqslant \left(\sum_{\substack{\chi \in [\Delta+1] \setminus [r_v]: \\ \mu_v^K(\chi)+\mu_v^e(\chi) \geqslant 1}} (\mu_v^K(\chi) + \mu_v^e(\chi) - 1) \right) + \left((\Delta + 1 - r_v) - \sum_{\chi=r_v+1}^{\Delta+1} \mu_v^K(\chi) - \sum_{\chi=r_v+1}^{\Delta+1} \mu_v^e(\chi) \right)$$

where the $\mu_v^K(\chi)$ and $\mu_v^e(\chi)$ cancel out leaving only the terms $\Delta + 1 - r_v$ and a sum of -1 over all colors used in $K \cup N(v)$, which is exactly the number of available non-reserved colors

$$\begin{aligned} z_v &\leqslant (\Delta + 1 - r_v) - |\{\chi \in (r_v, \Delta + 1] : \mu_v^K(\chi) + \mu_v^e(\chi) \geqslant 1\}| \\ &= |L(v) \cap L(K) \setminus [r_v]| \ . \end{aligned}$$ ∎

This second lemma shows that when z_v is too small, then v has slack in the reserved colors because many neighbors are colored using the same non-reserved colors.

Lemma 12.23. *Let φ be a coloring such that $\varphi(K) \cap [r_K] = \emptyset$. Then, the number of reserved colors available to each inlier $v \in I_{K_v}$ is*

$$|[r_v] \cap L_\varphi(v)| \geqslant \deg_\varphi(v) + 0.5\gamma_{12.7} \cdot e(K) - z_v \ .$$

PROOF. Using that $\Delta + 1 = (\Delta - \deg(v)) + |K| + e(v) - a(v) \geqslant |K| + e(v) - x_v - \delta e(v)$ (Eq (12.3)), we lower bound z_v as

$$z_v \geqslant (|K|+e(v)-\delta e(v))-r_v-\left(\sum_{\chi=r_v+1}^{\Delta+1}\mu_v^K(\chi)+\sum_{\chi=r_v+1}^{\Delta+1}\mu_v^e(\chi)\right)+\gamma_{12.7}\cdot e(K)$$

Recall that $e(v) \leqslant 25e(K)$ and $\delta \leqslant 1/100$, and hence $\gamma_{12.7}\cdot e(K)-25\delta e(K) \geqslant 0.5\gamma_{12.7} \cdot e(K)$,. Because nodes of K do not used reserved colors, we have $|K| = |K \setminus \operatorname{dom}\varphi| + \sum_{\chi=r_v+1}^{\Delta+1}\mu_v^K(\chi)$. Partition external neighbors of v as

$$e(v) = \underbrace{|E(v) \setminus \operatorname{dom}\varphi|}_{\text{uncolored}} + \underbrace{|E(v) \cap \varphi^{-1}([r_v])|}_{\text{colored with } \chi \in [r_v]} + \underbrace{\sum_{\chi=r_v+1}^{\Delta+1}\mu_v^e(\chi)}_{\text{colored with } \chi > r_v} .$$

Since uncolored neighbors $N(v) \setminus \operatorname{dom}\varphi \subseteq (K_v \cup E(v)) \setminus \operatorname{dom}\varphi$, the lower bound on z_v becomes

$$z_v \geqslant \deg_\varphi(v) + |E(v) \cap \varphi^{-1}([r_v])| - r_v + 0.5\gamma_{12.7} \cdot e(K) .$$

Finally, as $[r_v] \cap \varphi(K_v) = \emptyset$, the only colors lost in $|[r_v] \cap L_\varphi(v)|$ must be used by external neighbors. That is $|[r_v] \cap L_\varphi(v)| = r_v - |E(v) \cap \varphi^{-1}([r_v])|$. So the lemma follows from the lower bound on z_v. ∎

12.6.1. Algorithm Analysis. We have now all the ingredients of Algorithm 35. The first phase runs $O(\mathsf{cd})$ random color trials. To decide if they should take part in the color trial, vertices approximate z_v. Indeed, with aggregation on a BFS tree spanning K, vertices of K count $\sum_{\chi=r_v+1}^{\Delta+1}\mu_v^K(\chi)$ exactly. Using the fingerprinting technique, they approximate $\sum_{\chi=r_v+1}^{\Delta+1}\mu_v^e(\chi)$ up to an error $\Theta(\delta)\cdot e(v)$. Since they also know $\widetilde{e}(K) \in (1 \pm \Theta(\delta))e(K)$, they appropriate z_v up to a small error.

Claim 12.24. *Each v can compute $\tilde{z}_v \in z_v \pm \delta e(K)$ in $O(1/\delta^2)$ rounds.*

ALGORITHM 35. COMPLETE

Input: The coloring φ_{sct} produced by SYNCHRONIZEDCOLORTRIAL

Output: A coloring φ such that $V \setminus V_{cabal} = \operatorname{dom}\varphi$.

Phase I

Let $\varphi_0 = \varphi_{sct}$.

(1) **for** $i = 1, 2, \ldots, t = O(1)$
  (a) Each v computes $\tilde{z}_{v,i} \in z_{v,i} \pm \delta e(K)$ (w.r.t. φ_{i-1})
  (b) Let v join S_i if $\tilde{z}_{v,i} \geqslant 0.25\gamma_{12.7} \cdot \widetilde{e}(K)$
  (c) Each $v \in S_i$ runs TryColor with $\text{List}(v) = |L_{\varphi_i}(K) \setminus [r_v]|$. Call φ_i the resulting coloring.

(2) Each v computes $\tilde{z}_{v,t+1} \in z_{v,t+1} \pm \delta e(K)$ (w.r.t. φ_t)

(3) Let $S_{t+1} = S_t \setminus \operatorname{dom} \varphi_t$ be the nodes with $\tilde{z}_{v,t+1} \geqslant 0.25\gamma_{12.7} \cdot \widetilde{e}(K)$.

(4) Each $v \in S_{t+1}$ runs MultiColorTrial with $\text{List}(v) = [r_v]$. Call φ_{t+1} the resulting coloring.

Phase II

(1) Each $v \in H \setminus \operatorname{dom} \varphi_{t+1}$ runs $O(1)$ rounds of TryColor with $\text{List}(v) = [r_v]$.

(2) Run SlackColor with $\text{List}(v) = [r_v]$ for the remaining uncolored nodes.

Algorithm 35 begins by reducing the number of nodes with too few reserved colors available by trying random colors $O(1)$ times. Observe that before Step 4, we do not use any reserved color in S.

Lemma 12.25. *After Step 3 of Algorithm 35, w.h.p., for each almost-clique $|S_{t+1} \cap K| \leqslant e(K)$. Moreover, prior steps did not use colors of $[r_K]$ in K.*

Proof. We show that when v joins S_i, it verifies the conditions of Lemma 11.5 for small universal constants $\gamma = \Theta(\gamma_{12.7})$ independent of i. That is, vertex v has enough non-reserved colors available to get colored with constant probability by TryColor without using reserved colors, i.e., with $\text{List}(v) = |L_{\varphi_i}(K_v) \setminus [r_v]|$. This implies the lemma because, by Lemma 11.5, w.h.p., the size of sets S_i shrinks by a $(1 - \gamma^2/32)$-factor each iteration. Since there are only $O(e(K))$ uncolored nodes in each K initially, after $O(\gamma_1^{-2}) = O(1)$ iterations, the number of nodes left is $|S_{t+1}| \leqslant e(K)$.

Vertices can sample a uniform color in $L(K_v) \setminus [r_v]$ using the query algorithm (Lemma 11.3). So the first assumption of Lemma 11.5 holds.

If $v \in S_i$ then $\tilde{z}_{v,i} \geqslant 0.25\gamma_{12.7} \cdot \widetilde{e}(K)$, by definition. By Claim 12.24, the error of approximation is small, thus $z_{v,i} \geqslant \tilde{z}_{v,i} - \delta e(K) \geqslant 0.01\gamma_{12.7} \cdot e(K)$ by choosing $\delta = \Theta(\gamma_{12.7})$ small enough (Eq (12.1)). Lemma 12.22 implies there are $|L(v) \cap L(K_v) \setminus [r_v]| \geqslant z_v \geqslant 0.01\gamma_{12.7} \cdot e(K)$ available non-reserved colors. As $e(K) \geqslant \ell_{\min}/2$ in non-cabals, this implies the second condition of Lemma 11.5, $|\operatorname{List}(v)| \gg \gamma_{12.7}^{-2} \log n$. Since uncolored degrees are $O(e(K))$, this also implies the forth condition of Lemma 11.5, i.e., $|L(v) \cap L(K) \setminus [r_v]| \geqslant \Omega(\gamma_{12.7}) \cdot \deg_{\varphi_i}(v)$.

If $|L(K)| \geqslant 2(r_K + 25e(K))$, since $e(v) \leqslant 25e(K)$, half of the non-reserved colors in the clique palette are available because $|L(v) \cap L(K) \setminus [r_K]| \geqslant |L(K)| - 25e(K) - r_K \geqslant |L(K)|/2$. Otherwise, if $|L(K)| < 2(r_K + 25\widetilde{e}(K)) = O(e(K))$, a constant fraction of the non-reserved colors are available because $|L(v) \cap L(K) \setminus [r_K]| \geqslant 0.01\gamma_{12.7} \cdot e(K) \geqslant \Omega(\gamma_{12.7}) \cdot |L(K)|$. This implies the third condition of Lemma 11.5 and concludes the proof. ∎

After Step 3, we color all nodes with $z_v \geqslant \Omega(e(K))$ using MultiColorTrial with reserved colors in Step 4. Meanwhile, the only uncolored nodes left have $z_v \leqslant O(e(K))$, and thus have $\Omega(e(K))$ available reserved colors and can be colored later.

Lemma 12.26. *After Step 4 in Algorithm 35, w.h.p., each uncolored v has*

$$|L(v) \cap [r_v]| \geqslant \deg_{\varphi_{t+1}}(v) + 0.2\gamma_{12.7} \cdot e(K) \ .$$

Proof. Before Step 4, none of the colors in $[r_K]$ has been used by nodes of K, i.e., $[r_K] \cap \varphi_t(K) = \emptyset$. Uncolored vertices $v \notin S_{t+1}$ have $z_{v,t+1} \leqslant \tilde{z}_{v,t+1}/(1-\delta) \leqslant 0.3\gamma_{12.7} \cdot e(K)$. Lemma 12.23 implies

$$\begin{aligned} |[r_v] \cap L_{\varphi_t}(v)| &\geqslant \deg_{\varphi_t}(v) + 0.5\gamma_{12.7} \cdot e(K) - z_{v,t+1} \\ &\geqslant \deg_{\varphi_t}(v) + 0.2\gamma_{12.7} \cdot e(K) \ . \end{aligned}$$

Note that each time a neighbor of v is colored with some $\chi \in [r_v] \cap L_\varphi(v)$, then both sides of this inequality decrease by one. Hence, the inequality holds for any extension of φ_t.

We now argue that SlackColor with $\operatorname{List}(v) = [r_v]$ colors all $v \in S_{t+1}$ with high probability. Observe that v can describe $\operatorname{List}(v)$ to all its neighbors simply by broadcasting r_v. Since $\varphi_t(K) \cap [r_K] = \emptyset$, if a reserved color

$\chi \notin L(v)$ for some $v \in S_{t+1}$, it must be because of an external neighbor. Hence, $|[r_v] \cap L(v)| \geqslant r_v - e(v) \geqslant r_v - 21\widetilde{e}(K) \geqslant 3 \cdot 75e(K)$ by our choice of r_v (Eq (12.2)). On the other hand, each $v \in S_{t+1}$ has active uncolored degree $\deg_{\varphi_t}(v; S_{t+1}) \leqslant 30e(K)$ (at most $e(K)$ in $S_{t+1} \cap K_v$ and at most $25e(K)$ external neighbors). Since $e(K) \geqslant \ell_{\min}/2$ in non-cabals, Proposition 11.7 applies and SlackColor colors all nodes of S_{t+1} in $O(\log^* n)$ rounds with high probability. ∎

Lemma 12.27. *At the end of* Complete*, w.h.p., all nodes in $V \setminus V_{cabal}$ are colored.*

Proof. By Lemma 12.26, after Step 4, the only nodes remaining to color have slack $0.2\gamma_{12.3} \cdot e(K)$. Also recall $e(K) \geqslant \ell_{\min}/2$ because $K \notin \mathscr{K}_{cabal}$. So by Proposition 11.7 with $\mathrm{List}(v) = [r_v]$ and $s(v) = 0.2\gamma_{12.3} \cdot e(K)$, w.h.p., SlackColor colors all the remaining vertices in $O(\mathsf{cd} \log^* n)$ rounds. ∎

CHAPTER 13

$(\Delta+1)$-Coloring Low-Degree Virtual Graphs

In this chapter, we complete the result of Chapter 12 with an algorithm for coloring low-degree graphs in $\text{poly}(\log\log n)$ round. More formally, we prove

THEOREM 9.5. *Let H be a virtual graph on network G with n machines, bandwidth $\mathsf{b} = O(\log n)$, congestion $\mathsf{c} \leqslant n$ and dilation d. Suppose that Δ is the maximum degree of H and that it is known to all machines. There is a $O(\mathsf{cd} \cdot \log^7 \log n)$-round algorithm to $(\Delta+1)$-color H, with high probability.*

We implicitly assume in this chapter that $\Delta \leqslant \Delta_{low} = \text{poly}(\log n)$. Our approach is slightly different depending of whether $\Delta \gg \log n$ or not, we refer to the two cases as

- the sub-logarithmic regime when $\Delta \leqslant O(\log n)$, and
- the poly-logarithmic regime when $O(\log n) \leqslant \Delta \leqslant \Delta_{low}$.

In the poly-logarithmic regime, the high-level algorithm is the same as our algorithm when $\Delta \geqslant \Delta_{low}$, i.e, Algorithm 27. Importantly, we color vertices in the same order, i.e., sparse vertices first, then non-cabals, and finally cabals, but the way we color those vertices is different. In the sub-logarithmic regime, the algorithm is much simpler, and deals with all vertices at the same time. The polylogarithmic-regime algorithm can be viewed as successive applications of the sub-logarithmic-regime algorithm for $\Delta \leqslant O(\log n)$ over different subsets of vertices.

In both cases, the main ingredients of our coloring algorithm are the reduction of uncolored degrees down to $O(\log n)$, vertices learning at least $\deg+1$ colors in their palette, reducing uncolored parts of the graph to connected components of size $\text{poly}\log n$ (*shattering*), and an algorithm adapted from the deterministic CONGEST model to $(\deg+1)$-list color $\text{poly}\log n$-sized subgraphs of the virtual graph in $\text{poly}\log\log n$ rounds. The degree

reduction and shattering parts of the algorithm are minor adaptations of a classic result by Barenboim, Elkin, Pettie, and Schneider [BEPS16]. The last part of this section is technically most novel: an adaptation of an algorithm by Ghaffari and Kuhn [GK21] to color $\operatorname{poly}\log n$-sized subgraphs in $\operatorname{poly}\log\log n$ rounds, for which we once more rely on fingerprinting. We state here this result, with details Section 13.3.

Lemma 13.1 (Ghaffari-Kuhn Algorithm in Virtual Graphs)**.** *Let $\Delta \leqslant \operatorname{poly}(\log n)$. Let F be an N-vertex virtual graph with maximum degree $\Delta_F \leqslant O(\log n)$ on a network of bandwidth $\Theta(\log n)$. Suppose each vertex knows a list $L_v \subseteq [\Delta+1]$ of $\deg_F(v)+1$ colors. There exists a randomized algorithm that L_v-list-colors F with probability $1-1/\operatorname{poly}(n)$ in $O(\log N \cdot \log^6 \log n)$.*

The plan for this section is as follows. In Section 13.1, we explain how we perform the coloring when $\Delta \leqslant O(\log n)$. This will set the stage for the polylogarithmic regime that we describe next. Finally, we analyze the implementation of the Ghaffari-Kuhn algorithm behind Lemma 13.1.

13.1. The Logarithmic Regime

The logarithmic regime is greatly simplified by the fact that in $O(\mathsf{cd})$ rounds, each vertex can learn the colors used by its neighbors, and conversely, which colors it still has available. To do so, we simply aggregate a $O(\log n)$-bitmap in each cluster, where each bit encodes whether a given color is used. The same idea is used when Δ is larger, once we are dealing with vertices of uncolored degree $O(\log n)$.

Algorithm 36. High-Level Coloring Algorithm when $\Delta \leqslant O(\log n)$

Input: A virtual graph H on G such that $\Delta \leqslant O(\log n)$

Output: A $(\Delta+1)$-coloring

(1) Shattering
(2) SmallInstanceColoring (Section 13.3)

In all subsequent sections, we repeatedly use essentially the same algorithm as Algorithm 36 to color subsets of vertices, with the small difference

that we add two steps before shattering: a degree reduction step, and a step to learn a set of deg +1 colors at each vertex.

Shattering. In both this regime and for higher Δ, SHATTERING simply consists of each uncolored node trying a random color from its palette for $O(\log\log n)$ rounds. By an argument of Barenboim, Elkin, Pettie and Schneider [BEPS16, Lemma 5.3], this results in the uncolored parts of the graph being of size $O(\Delta^2 \log_\Delta n) \leqslant O(\log^3 n)$. Between two random color trials, nodes update their palettes in $O(1)$ rounds by learning which colors from their list is used by their neighbors. Since lists have size deg +1 $\leqslant \Delta + 1 = O(\log n)$, this can be done by aggregating a $O(\log n)$-bitmap on their cluster as mentioned before. This allows the node to always try colors from their palette. In total, this step takes $O(\log\log n)$ rounds.

Coloring small connected components. Finally, vertices complete the coloring using Lemma 13.1. Note that even in our first simple setting of $\Delta \leqslant O(\log n)$, this is not immediate. In particular, even though $O(\log \Delta) = O(\log\log n)$ rounds suffice for each vertex to broadcast a $O(\log \Delta)$-bit message and receive the set of messages sent by its $\Delta = O(\log n)$ neighbors, our setting is more difficult than the BCONGEST setting with $\log \Delta$-bit messages. This is because while each vertex can receive the set of messages sent by its neighbors, it cannot a priori know how many times each message was received. As a result, we spend some effort adapting the Ghaffari-Kuhn algorithm to the virtual graph setting. This is done in Section 13.3.

13.2. The Polylogarithmic Regime

In the polylogarithmic regime, the idea is essentially to color vertices in an order which allows us to color them exactly as in the logarithmic regime, i.e., by performing $O(\log\log n)$ random color trials and using our adaptation of the Ghaffari-Kuhn algorithm. To reduce the degree to $O(\log n)$, and ensure that each vertex can learn deg +1 colors from its palette, we color the vertices in the same order as in the $\Delta \geqslant \Delta_{low}$ regime, and use the same technique as in this regime to obtain slack. For dense vertices, we also make use of the clique palette.

ALGORITHM 37. High-Level Coloring Algorithm for polylogarithmic Δ

Input A virtual graph H on G such that $O(\log n) \leqslant \Delta \leqslant \Delta_{low}$

Output A $(\Delta+1)$-coloring

(1) COMPUTEACD
(2) SLACKGENERATION in $V \setminus V_{cabal}$ (Proposition 12.3)
(3) COLORINGSPARSE
    Let $\ell = \Theta(\log n)$ for the purpose of defining cabals (i.e., cabals have $\tilde{e}_K \leqslant O(\log n)$)
    COLORINGDENSE in $V \setminus V_{cabal}$
(4) COLORINGDENSE in V_{cabal}

A small difference between this algorithm, for the polylogarithmic regime, and our $O(\log^* n)$ algorithm for $\Delta \geqslant \Delta_{low}$, is that we define cabals as almost-cliques with $O(\log n)$ external degree, rather than $O(\log^{1.1} n)$. The reasons for this change are that $e(K) \geqslant \Omega(\log n)$ is sufficient for inliers to obtain slack from their external neighbors during SLACKGENERATION, and changing the threshold avoids having to give a special treatment to almost-cliques with external degree between $\Theta(\log n)$ and $\Theta(\log^{1.1} n)$. This change is possible as we do not make use of any procedure requiring $\omega(\log n)$ slack, like SLACKCOLOR (Proposition 11.7).

ALGORITHM 38. COLORINGDENSE, $O(\log n) \leqslant \Delta \leqslant \Delta_{low}$

(1) COLORFULMATCHING
(2) COLORINGOUTLIERS
(3) COLORINGINLIERS

The three procedures COLORINGSPARSE, COLORINGOUTLIERS, and COLORINGINLIERS are near identical and very similar to the algorithm for $\Delta \leqslant O(\log n)$ (Algorithm 36). They only differ in the way in which vertices are able to reduce their degree and learn colors prior to shattering, i.e., the implementation of the first two steps, and have the exact same last two

steps. The rest of this section explains how each set of vertices implements its own version of each procedure.

ALGORITHM 39. COLORING(SPARSE/OUTLIERS/INLIERS)

(1) Repeat TRYCOLOR $O(\log\log n)$ times
(2) LEARNCOLORS
(3) SHATTERING
(4) SMALLINSTANCECOLORING (Lemma 13.1)

13.2.1. Sparse vertices and outliers. Sparse vertices have permanent slack $\Omega(\Delta)$, w.h.p., by Proposition 12.3, while outliers have (temporary) slack $\Omega(\Delta)$ from the fact that inliers in their clique are colored later. This is both useful for reducing the degree and for finding $\deg+1$ free colors. Let us focus on the subgraph induced by sparse vertices, which is the only active part of the graph during COLORINGSPARSE, and which gets fully colored by this procedure. The subgraphs induced by outliers in non-cabals and in cabals are colored similarly by COLORINGOUTLIERS

Degree reduction. A color sampled uniformly at random in $[\Delta+1]$ by an uncolored sparse vertex always has a constant probability of success. This is also true for outliers. By Corollary 11.6, after $O(\log\log n)$ random color trials in $[\Delta+1]$, each sparse vertex has at most $O(\log n)$ uncolored neighbors in the subgraph of sparse vertices (similarly in the subgraph induced by the currently considered outliers, when performing COLORINGOUTLIERS).

Learning colors. Recall that sparse vertices have permanent slack $\Omega(\Delta)$, where $\Delta \geqslant \Omega(\log n)$, w.h.p., and have at most $O(\log n)$ uncolored sparse neighbors after the degree reduction step. To learn colors, each sparse vertex spends $\Theta(\log\log n)$ rounds sampling $\Theta(\log n/\log\log n)$ colors that it has not previously sampled before, and asking its neighbors which of them are available. This has each vertex v send $\Theta(\log n)$ colors to its neighbors, each of which has a constant probability to be free unless $(1+\Omega(1))\deg(v)$ colors have already been discovered. This allows us to argue by Chernoff bound (Lemma 2.3) that each uncolored sparse vertex discovers at least $\deg(v)+1$ free colors from its palette, with high probability. The same

argument applies to outliers. Note that for both sparse vertices and outliers, the vertices have to ask their whole colored neighborhood whether some given colors are free, not just vertices of the same type.

Completing the coloring. From there, sparse vertices and outliers are colored in the same way we color vertices in the logarithmic regime (when $\Delta \leqslant O(\log n)$). We first have them try $O(\log \log n)$ random colors from their palette, using the colors discovered in the previous step and updating their palette between two random color trials, in $O(\mathsf{cd})$ rounds every time. Updating the palette in $O(\mathsf{cd})$ rounds is possible due to all machines in each support knowing the $O(\log n)$ palette from which the vertex tries colors, and thus, updating the palette can be done by aggregation of a $O(\log n)$-sized bitmap indicating which colors from this palette were taken by neighboring vertices. After that, the subgraph induced by uncolored sparse vertices or the currently considered set of outliers has connected components of size at most $\operatorname{poly}\log n$, and we can finish the coloring using Lemma 13.1.

13.2.2. Inliers. The treatment of inliers differs from that of sparse vertices and outliers, in that we rely on the clique palette to sample colors instead of directly sampling in $[\Delta + 1]$. We do not reserve colors, contrary to the case where $\Delta \geqslant \Delta_{low}$.

Querying The Clique Palette & Colorful Matching. In the following, we will need that vertices can use the query algorithm (Lemma 11.3) and the assumption that there is a large enough colorful matching. The query algorithm and the colorful matching algorithm when $a(K) \geqslant \log n$ (Lemma 11.9) only requires that $\Delta \gg \log n$. However, the colorful matching algorithm for cabals where $a(K) \leqslant O(\log n)$ as presented in Section 12.4 assumes that $\Delta \gg \log^2 n$, which might not hold here.

The algorithm of Section 12.4 requires $\Delta \gg \log^2 n$ only in two places: when it colors the $O(\log n)$ anti-edges and when it selects anti-edges in Algorithm 31. Let us explain how to implement the former first. For each trial, the algoritm samples a min-wise hash function $[O(\Delta)] \to [O(\Delta)]$ that can be represented in $O(\log \Delta) = O(\log \log n)$ bits and computes the anti-neighbor with the smallest hash value. Note that we can label vertices with identifiers $\{1, 2, \ldots, |K|\}$ which take $O(\log \Delta)$ bits to describe instead

of their $\Theta(\log n)$ bits identifiers. So each trial aggregates one $O(\log \Delta) = O(\log \log n)$ bit message, which requires $O(\log \log n)$ rounds to perform for all $k = O(\log n)$ trials in parallel. In fact, in this much time, all vertices of K can learn all aggregates, thus computing the anti-edges does not require additional communication.

Let F be the $O(\log n)$ anti-edges selected by the matching algorithm. To color them we use TryColor and SlackColor, which requires the endpoints of each anti-edge to broadcast the same $O(\log n)$-bit message[1]. We select different relays for each anti-edge in F: a vertex w_i incident to both endpoints of the i-th anti-edge in F. To color F, each relay runs TryColor and SlackColor and anti-edges repeat the corresponding messages. We find relays using a CONGEST algorithm for maximal matching. To ensure it can be emulated efficiently on virtual graph, we run the CONGEST algorithm on a random subset of the vertices.

Lemma 13.2. *Assume let $k = \Theta(\varepsilon^{-1} \log n)$ as in Algorithm 31 and $\Delta \geqslant 3k$. Let K be some almost-clique and F a set of $O(\log n)$ vertex-disjoint anti-edges of $G[K]$. There is an $O(\mathsf{cd} \cdot \log^4 \log n)$-round algorithm that computes relays $v_1, \dots, v_{|F|}$ such that $v_i \neq v_j$ for all $i \neq j$ and v_i is adjacent to both endpoints of the i-th anti-edge in F.*

Proof. Sample each vertex in K independently with probability $3k/\Delta$. By the Chernoff Bound, w.h.p., both endpoints of the i-th anti-edge are incident to $> k$ sampled vertices. By union bound, this holds for all anti-edges of F. Consider the bipartite graph that has the anti-edges of F on the left and the sampled vertices on the right. Put an edge between and anti-edge and a sampled vertex when the vertex is incident to both endpoints of the anti-edge, i.e., when it can act as a relay. Since each anti-edge has degree $\geqslant k$, each anti-edge is matched in a maximal matching. So we compute a maximal matching in the bipartite graph and let v_i be the sampled vertex matched with the i-th anti-edge.

To compute a maximal matching, run the congest algorithm of [Fis20]. It runs in $O(\log^2 \Delta \cdot \log N)$ rounds with $O(\log N)$ bit messages on N-vertices

[1] It cannot be reduced to $O(\log \Delta)$ bits because of the use of representative sets in the implementation of MultiColorTrial in virtual graphs.

graphs. Here, vertices have $O(\log \Delta)$-bits unique identifiers so the algorithm runs in $O(\log^2 \Delta \log N) = O(\log^3 \log n)$ rounds. In each round of Fischer's algorithm, a vertex may have to send $O(k \log \Delta) = O(\log n \cdot \log \log n)$ bits, since w.h.p. each anti-edge is incident to at most $6k$ sampled vertices, so it takes $O(\mathsf{cd} \cdot \log \log n)$ rounds to simulate each round in virtual graphs by broadcasting all sent messages. ∎

Non-cabals. In non-cabals, and more generally, when $a(K) + e(K) \geqslant \Omega(\log n)$, inliers receive $\Omega(a(K) + e(K))$ slack from SlackGeneration and ColorfulMatching, even when using the clique palette. Most importantly, for every uncolored inlier v, the intersection of its palette with the clique palette always contains more colors than the uncolored degree of v (Lemma 12.8). Focusing on the subgraph induced by inliers of non-cabals, by Lemma 11.5, each use of TryColor where each active inlier uses the clique palette of its almost-clique to sample colors reduces the degree of each inlier by a constant fraction, granted it has degree $\Omega(\log n)$. As a result, after $O(\log \log n)$ of trying random colors, the graph induced by the uncolored inliers in non-cabals has maximum degree $O(\log n)$.

Finally, since inliers have slack $\Omega(a(K) + e(K))$ even when using the clique palette in that regime, they can find $\deg +1$ free colors by sampling $\Theta(\log n)$ colors in the clique palette. This is similar to the way that sparse vertices and outliers find free colors, the crucial difference being that inliers sample colors from the clique palette instead of $[\Delta+1]$. From there, we have reached the point where the algorithm is the same as for sparse vertices and outliers.

Cabals. Contrary to inliers in non-cabals, however, they may not have slack. However, in cabals, every inlier has $e(v) \leqslant c \log n$ for some large constant c (depending on ε). So, as long as the clique palette contains at least $2c \log n$ colors, by Lemma 11.5, trying a color from the clique palette decreases the number of vertices by a constant factor. When the clique palette contains fewer than $2c \log n$ colors, all vertices may learn the whole clique palette in $O(\log \log n)$ rounds. Since they have $O(\log n)$ external neighbors, these inliers learn exactly which colors are used by their external neighbors, in $O(\log \log n)$ rounds, similarly to the way vertices can

efficiently learn all the colors used by their neighbors when $\Delta \leqslant O(\log n)$. So they learn their exact palette in $O(\log \log n)$ rounds. From then on, they can perform random color trials using only colors from their palette, which colors vertices with constant probability. After $O(\log \log n)$ rounds of this, each cabal contains $O(\log n)$ uncolored vertices with high probability.

By the accounting lemma (Lemma 4.19), the clique palette contains enough colors for each inlier. So, after learning the clique palette (or the $O(\log n)$ smallest colors of the clique palette) and the color of external neighbors, in $O(\log \log n)$ rounds, the vertices learn a list of $\deg_\varphi + 1$ many colors. We thus reach the point where inliers from cabals can be treated like all other types of vertices, and be colored by a combination of trying $O(\log \log n)$ random colors so the subgraph of uncolored vertices has connected components of size $\operatorname{poly} \log n$, and we color them in $\operatorname{poly} \log \log n$ rounds.

13.3. Coloring the Shattered Instances

In this section, we prove Lemma 13.1, which gives a $\operatorname{poly} \log \log n$ algorithm for coloring a subgraph of a virtual graph whose connected components are of size $\operatorname{poly} \log n$, and whose vertices know a list of at least $\deg + 1$ free colors.

Contrary to previous approaches for post-shattering, we do not run a deterministic algorithm on shattered instances but a randomized one. Recall that the whole graph has size n and shattered instances have size $N = \operatorname{poly}(\log n)$. We implement the Ghaffari-Kuhn algorithm for $(\deg+1)$-list-coloring in CONGEST [GK21, Section 4.2, arxiv version] but use the fingerprinting technique (see Section 11.3) to implement its subroutines on virtual graphs. The runtime is $O(\log N) \cdot \operatorname{poly}(\log \Delta) = \operatorname{poly} \log \log n$ and the success probability $1 - 1/\operatorname{poly}(n)$. Let us introduce some notations and the general scheme of this algorithm.

Overview of the Ghaffari-Kuhn Algorithm. Let φ be the partial coloring of H and suppose each uncolored vertex knows a list $L_v \subseteq L_\varphi(v) \subseteq [\Delta+1]$ of $\deg_\varphi(v)+1$ colors (given as input). The Ghaffari-Kuhn algorithm uses local rounding to L_v-list-color a constant fraction of the uncolored vertices, so repeating that $O(\log N)$ times colors the whole graph.

Partition the color space $\mathscr{C} \stackrel{\text{def}}{=} [\Delta+1]$ into $K \stackrel{\text{def}}{=} O(\sqrt{\log|\mathscr{C}|})$ chunks $P_1, P_2, \ldots, P_K$ of equal size and use local rounding to let vertices decide on a subspace P_i they restrict themselves to. Then, we recursively run the algorithm in parallel in each P_i. After $Q \stackrel{\text{def}}{=} O(\log|\mathscr{C}|/\log\log|\mathscr{C}|)$ levels of recursion, each part contains exactly one color and we obtain a coloring.

Let us explain how each vertex chooses its part so that the number of monochromatic edges in the end is $O(N)$. Let $\mathscr{L} \stackrel{\text{def}}{=} [K]$ be the set of labels for the parts $P_1, \ldots, P_K$. The Ghaffari-Kuhn algorithm takes a fractional labelling assignment — a vector $x \in [0,1]^{V_H \times \mathscr{L}}$ — and labeling-cost[2] $y_{u,\ell}+y_{v,\ell}$ on each edge $\{u,v\} \in E_H$ where $y \in [0,1]^{V_H \times \mathscr{L}}$ are penalties defined by each vertex. Each vertex v knows only $x_{v,\ell}$ and $y_{v,\ell}$ for each label. The cost of the x assignment, with respect the y-vector, is

$$C(x,y) \stackrel{\text{def}}{=} \sum_{uv \in E_H} \sum_{\ell \in \mathscr{L}} x_{v,\ell} x_{u,\ell} (y_{v,\ell} + y_{u,\ell}) \ . \tag{13.1}$$

The heart of the Ghaffari-Kuhn algorithm is an approximate rounding algorithm which takes as input the vectors $x, y \in [0,1]^{V_H \times \mathscr{L}}$ and produces a vector $x' \in \{0,1\}^{V_H \times \mathscr{L}}$ such that $C(x',y) \leqslant (1+\varepsilon)C(x,y)$.

Implementing Ghaffari-Kuhn on Cluster Graphs. The main technical challenge in virtual graphs is for vertices to estimate how costly it would be to increase the value on some labels. Namely, in virtual graphs, vertices cannot distinguish between being linked to one neighbor with label ℓ through many paths or having many neighbors with label ℓ. To overcome this issue, we use the fingerprinting technique with weights to approximate cost (or weights) of labels over neighborhoods. Lemma 13.4 describes the types of sums we can approximate efficiently. We use Lemma 13.4 to implement the two core routines of the Ghaffari-Kuhn algorithm: the weighted defective coloring (Lemma 13.6) and the approximate rounding lemma (Lemma 13.7). The key aspect of weights we rely on when using Lemma 13.4 is the separate contributions of u and v to the cost of each edge.

[2]Notice Eq (13.1) does not capture the full generality of the approximate rounding lemma of [GK21, Section 3], but this will suffice for the coloring algorithm described in [GK21, Section 4.2].

Definition 13.3. A vector $x \in [0,1]^I$ is 2^{-b}-integral iff $x_i = k_i/2^b$ where $k_i \in \mathbb{N}_{\geqslant 0}$ for all $i \in I$.

Lemma 13.4. *Let $b \geqslant 0$ and suppose $x \in [0,1]^{V_H}$ is a 2^{-b}-integral vector such that v knows b, x_v and each edge $\{u,v\} \in E_H$ holds $\alpha_{u\to v}, \alpha_{v\to u} \in \{0,1\}$. There exists a randomized $O\Big(\mathsf{cd}(\varepsilon^{-2} + \frac{\log b + \log \Delta}{\log n})\Big)$-round algorithm at the end of which every vertex knows $\widetilde{W}_v$ such that with probability $1 - 1/\mathrm{poly}(n)$, for each $v \in V_H$ we have*

$$\widetilde{W}_v \in (1 \pm \varepsilon) W_v \quad \text{where} \quad W_v = \sum_{u \in N(v)} \alpha_{u\to v} \cdot x_u \ .$$

Proof. For each u, let k_u be the integer such that $x_u = k_u/2^b$. At a high level, we duplicate k_u times each u and use the fingerprinting algorithm with edges filtering out neighbors for which $\alpha_{u\to v} = 0$. More formally, let $t = O(\varepsilon^{-2} \log n)$ and let $\mathbf{X}_{v,j,i}$ be independent samples from a geometric distribution of parameter $1/2$ where $j \in [k_v]$ and $i \in [t]$. Then, for all $i \in [t]$, let

$$\mathbf{Y}_i^v = \max_{\substack{u \in N(v) \\ \alpha_{u\to v} = 1}} \max_{j \in [k_u]} \mathbf{X}_{u,j,i}$$

which is the maximum of $\sum_{u \in N(v)} \alpha_{u\to v} k_u = 2^b W_v$ independent geometric variables since $\alpha_{u\to v} \in \{0,1\}$. Since all vertices know b, by Lemma 11.12 (with $d = 2^b W_v$), vertex v computes $\widetilde{W}_v$ from the values $(\mathbf{Y}_i^v)_{i\in[t]}$ such that, with high probability, $\widetilde{W}_v \in (1 \pm \varepsilon) W_v$. This holds simultaneously at all vertices with high probability by union bound.

To implement this on a virtual graph, one designated machine in $V(u)$ samples all variables $\mathbf{X}_{u,i,j}$ and locally computes the vector $(\max_j \mathbf{X}_{u,j,i})_{i\in[t]}$, i.e., the second max in the definition of $\mathbf{Y}_i^v$. By Lemma 11.16, encoding the $(\max_j \mathbf{X}_{u,j,i})_{i\in[t]}$ uses $O(t + \log\log k_u) = O(\varepsilon^{-2} \log n + \log b)$ bits since $k_u \leqslant 2^b$. Hence, those aggregated maximums can be distributed to all machines of $V(u)$ in $O(\varepsilon^{-2} + \frac{\log b}{\log n})$ rounds. Since machines of $V(u) \cap V(u)$ know $\alpha_{u\to v}$, we can properly aggregate each $\mathbf{Y}_i^v$ over all neighbors $u \in N(v)$ such that $\alpha_{u\to v} = 1$. By Lemma 11.16, encoding variables $(\mathbf{Y}_i^v)_{i\in[t]}$ (as well as all partial aggregates) can be done in $O(t + \log\log(2^b W_v))$ bits. Observe that since $x_v \leqslant 1$ it must be that $W_v \leqslant |N(v)|$; hence, pipelining requires $O(\varepsilon^{-2} + \frac{\log b + \log |N(v)|}{\log n})$ rounds of pipelining, which concludes the proof. ∎

Let us now define what weighted defective colorings are and explain how we compute them on virtual graphs. We emphasize that Lemma 13.6 mostly follows from previous work (e.g., [FGGKR23] and references thererin) so we focus on how we use Lemma 13.4 to implement it on virtual graphs.

Definition 13.5. Let H be a graph and $w : E_H \to \mathbb{R}_{\geqslant 0}$ be non-negative edge weights. For $q \geqslant 1$ and $\delta > 0$, a weighted δ-relative q-coloring of H is a function $\psi : V_H \to [q]$ such that

$$\text{for all } v \in V_H, \quad \sum_{u \in N(v)} \mathbb{I}[\psi(v) = \psi(u)] \cdot w(uv) \leqslant \delta \cdot \sum_{u \in N(v)} w(uv) \ .$$

Lemma 13.6 (Weighted Defective Coloring)**.** *Suppose we have a $O(\log^2 n)$-proper coloring. Let $\mathscr{L}$ be some set of labels and suppose each edge $\{u, v\} \in E_H$ has weight*

$$w(uv) = \sum_{\ell \in \mathscr{L}} x_{u,\ell} x_{v,\ell} (y_{v,\ell} + y_{u,\ell}) \ . \tag{13.2}$$

where $x, y \in [0,1]^{V_H \times \mathscr{L}}$ are $2^{-\Theta(\log(n))}$-integral vectors. Then, for any $\delta > 0$, there is a randomized $O\big(\mathsf{cd} \cdot \frac{1}{\delta} |\mathscr{L}| \log\log n \cdot \log\log\log n\big)$-round algorithm that returns a weighted δ-relative-defective $O(1/\delta^2)$-coloring of H with probability $1 - 1/\mathrm{poly}(n)$.

Proof. Starting from ψ_0, the given q_0-coloring with $q_0 = O(\log^2 n)$, we compute a sequence of δ_i-weighted-defective q_i-colorings ψ_i for $i = 1, \ldots, t = O(\log^* q_0)$. In [Lin92] (and also later in [Kuh09; MT20; BEG22] etc), it is shown that there is a universal constant $c \geqslant 1$ such that for any $s, q \in \mathbb{N}_{\geqslant 1}$ there exists sets $S_1, \ldots, S_q \subseteq [s^2 \tau]$ of *candidate colors* such that

$$\begin{aligned} &\text{for all } i \in [q], \quad |S_i| = s\tau \quad \text{and for all } j \neq i \quad |S_i \cap S_j| < \tau \\ &\qquad\qquad \text{where } \tau = c \cdot \min\{\log q, \log_s^2 q\} \ . \end{aligned} \tag{13.3}$$

From ψ_i, we compute ψ_{i+1} as follow. Let $S_1, \ldots, S_{q_i}$ be the sets such as described by Eq (13.3) with $s_i = 2^{t-i+2}/\delta$ and $q = q_i$. For succinctness, let $S_v \overset{\text{def}}{=} S_{\psi_i(v)}$ and $N_{i,b}(v)$ is the set of $u \in N(v)$ with $\psi_i(u) \neq \psi_i(v)$. Vertex v computes for each candidate color $\chi \in S_v$ the weight of its ψ_i-bichromatic edges to neighbors sharing candidate color χ:

$$W_{v,\chi} = \sum_{u \in N_{i,b}(v)} w(uv) \mathbb{I}[\chi \in S_u] \ .$$

Using Eq (13.3), one can show that there is a color $\chi \in S_v$ for which $W_{v,\chi} \leqslant W_v/s_i$ where $W_v = \sum_{u \in N_{i,b}(v)} w(uv)$. By having v choose a color $\psi_{i+1}(v) = \chi \in S_v$ such that $W_{v,\chi}$ is within a factor 2 from the minimum $W_{v,\chi'}$ for $\chi' \in S_v$, we therefore increase the defect at most by an additive $2W_v/s_i = \delta W_v/2^{t-i+1}$ term. After t iterations, we obtain a $O(1/\delta^2)$-coloring ψ_t with defect of $(2/s_0 + 2/s_1 + \ldots + 2/s_{t-1})W_v \leqslant \delta W_v$. We refer the readers to [KS18, Appendix A] or [Kuh09, Theorem 4.9] for more details.

We now explain how each vertex computes $\widetilde{W}_{v,\chi} \in (1 \pm 0.5)W_{v,\chi}$ for all $\chi \in S_v$. Simple calculation shows that one can decompose each $W_{v,\chi}$ as

$$W_{v,\chi} = \sum_{\ell \in \mathscr{L}} x_{v,\ell} y_{v,\ell} W^{(1)}_{v,\chi,\ell} + x_{v,\ell} W^{(2)}_{v,\chi,\ell}$$

where for each $\ell \in \mathscr{L}$ we define

$$W^{(1)}_{v,\chi,\ell} = \sum_{u \in N_{i,b}(v)} \mathbb{I}[\chi \in S_u] x_{u,\ell} \quad \text{and} \quad W^{(2)}_{v,\chi,\ell} = \sum_{u \in N_{i,b}(v)} \mathbb{I}[\chi \in S_u] x_{u,\ell} y_{u,\ell} \ .$$

We can assume without generality that all machines of $V(v)$ know all values $x_{v,\ell}$ and $y_{v,\ell}$ as broadcasting them requires $O(|\mathscr{L}|)$ rounds (they each take $O(\log n)$ bits to describe from our integrality assumption). Also note that machine in $V(v) \cap V(u)$ knows all candidate colors of u and v, hence can locally compute $\mathbb{I}[\chi \in S_u]$. Hence, to approximate $W_{v,\chi}$, it suffices that v approximates each $W^{(1)}_{v,\chi,\ell}$ and $W^{(2)}_{v,\chi,\ell}$ for all $\ell \in \mathscr{L}$. Notice that any fixed sum $W^{(1)}_{v,\chi,\ell}$ and $W^{(2)}_{v,\chi,\ell}$ can be computed w.h.p., at all vertices with candidate color χ in $O(\mathsf{cd})$ rounds, by Lemma 13.4 (with $\varepsilon = 1/2$ and $\alpha_{u \to v} = \mathbb{I}[\psi_i(u) \neq \psi_i(v) \wedge \chi \in S_u]$ in the instance of $W_{v,\chi,\ell}$). Hence, if vertices run this algorithm in parallel for all $\chi \in S_v$ and $\ell \in \mathscr{L}$, through basic pipelining, one step of color reduction runs in $O(\mathsf{cd} \cdot |\mathscr{L}| s_i \tau_i)$ rounds as $|S_v| = s_i \tau_i$. One can show by induction that $\log_{s_i} q_i \leqslant 4 \log^{(i+1)} q_0$ (see, e.g., [KS18, Proof of Theorem 2]) so that for all $i \in [0, t-1]$,

$$s_{i+1}\tau_{i+1} \leqslant 2^{t+2}/\delta \cdot 16c \Big(\log^{(i+1)} q_0 \Big)^2 \ .$$

So computing ψ_{i+1} from ψ_i for $i = 1, 2, \ldots, t-1$ takes $O(\mathsf{cd} \cdot |\mathscr{L}|) \cdot 2^{O(\log^* n)}/\delta \cdot O(\log^2 \log \log n) = O(\mathsf{cd} \cdot \frac{1}{\delta} |\mathscr{L}| \cdot \log^3 \log \log n)$ rounds. When we compute ψ_1 from ψ_0, note that $s_0 \tau_0 \leqslant 2^{t+2}/\delta \cdot c \log q_0$ so it takes $O(\mathsf{cd} \cdot \frac{1}{\delta} |\mathscr{L}| \cdot \log \log n \cdot \log \log \log n)$ rounds. Over the $t = O(\log^* n)$ iterations of color reduction, the cost of the first step dominates and we obtain the claimed runtime. ∎

Lemma 13.7 (Approximate Rounding)**.** *Let* $2^{-\Theta(\sqrt{\log n})} < \varepsilon < 1$, $b \in [1, \Theta(\log n)]$ *be an integer and* $\mathscr{L}$ *a set of labels with* 2^{-b}*-integral vectors* $x, y \in [0,1]^{V_H \times \mathscr{L}}$ *and we are given a proper* $O(\log^2 n)$*-coloring. There is a randomized algorithm that computes a* 2^{-b+1}*-integral* $x' \in [0,1]^{V_H \times \mathscr{L}}$ *such that, w.h.p.,* $C(x', y) \leqslant (1+\varepsilon)C(x, y)$*. It ends after*

$$O\left(\mathsf{cd}|\mathscr{L}|\left(\frac{\log\log n \cdot \log\log\log n}{\varepsilon} + \frac{1}{\varepsilon^4}\right)\right) \quad \textit{rounds.}$$

Proof. Recall the main steps of approximate rounding in [GK21, Section 3]. First, compute a $\varepsilon/8$-defective $O(1/\varepsilon^2)$-coloring using Lemma 13.6. Then, go sequentially over color classes $i = 1, 2, \ldots, O(1/\varepsilon^2)$ of the defective coloring[3] and, each time, vertices v of the i-th color class decide on two disjoint sets $\mathscr{L}_v^-, \mathscr{L}_v^+ \subseteq \overline{\mathscr{L}_v}$ of equal size where labels $\ell \in \overline{\mathscr{L}_v}$ are those for which $x_{v,\ell}$ is not a multiple of 2^{-b+1}. Then, all $x_{v,\ell}$ for $\ell \in \mathscr{L}_v^-$ are updated to $x_{v,\ell} - 2^{-b}$ and all $x_{v,\ell}$ for $\ell \in \mathscr{L}_v^+$ are increased to $x_{v,\ell} + 2^{-b}$. It is easy to verify that the resulting x-vector is 2^{-b+1}-integral.

To decide on $\mathscr{L}_v^-, \mathscr{L}_v^+$, for each $\ell \in \overline{\mathscr{L}_v}$, vertices compute the sums

$$W_{v,\ell} = \sum_{u \in N(v)} x_{u,\ell}(y_{u,\ell} + y_{v,\ell})$$

As shown in [GK21, Lemmas 3.4 and 3.5],

$$C(x', y) - C(x, y) \leqslant \sum_v \left(\sum_{\ell \in \mathscr{L}_v^+} W_{v,\ell} - \sum_{\ell \in \mathscr{L}_v^-} W_{v,\ell} \right) + \varepsilon/2 \cdot C(x, y) \ .$$

We compute approximations $\widetilde{W}_{v,\ell} \in (1 \pm \varepsilon) W_{v,\ell}$ for each $\ell \in \overline{\mathscr{L}_v}$ as follow. Using the definition of weights, simple calculation shows that $W_{v,\ell}$ can be decomposed as

$$W_{v,\ell} = y_{v,\ell} W_{v,\ell}^{(1)} + W_{v,\ell}^{(2)}$$

where

$$W_{v,\ell}^{(1)} = \sum_{u \in N(v)} x_{u,\ell}$$

[3] In [GK21], the authors quadratically improve the runtime by using an *average* weighted ε-relative defective $O(1/\varepsilon)$-coloring. Here, in the interest of simplicity, we compute only a weighted defective coloring.

and

$$W^{(2)}_{v,\ell} = \sum_{u \in N(v)} x_{u,\ell} y_{u,\ell} \ .$$

By Lemma 13.4, these sums can be approximated, w.h.p., up to multiplicative error $1+\varepsilon/4$ in $O(\mathsf{cd}|\mathscr{L}|/\varepsilon^2)$-rounds. Then, put the $|\overline{\mathscr{L}_v}|/2$ labels with the largest approximate weights in $\mathscr{L}_v^-$ and the rest in $\mathscr{L}_v^+$. This implies that

$$\sum_{\ell \in \mathscr{L}^-} \widetilde{W}_{v,\ell} \geqslant \sum_{\ell \in \mathscr{L}^+} \widetilde{W}_{v,\ell}$$

and hence

$$\sum_{\ell \in \mathscr{L}^-} W_{v,\ell} \geqslant \frac{1-\varepsilon/4}{1+\varepsilon/4} \sum_{\ell \in \mathscr{L}^+} W_{v,\ell} \geqslant (1-\varepsilon/2) \sum_{\ell \in \mathscr{L}^+} W_{v,\ell} \ .$$

In particular, this implies that $C(x', y) \leqslant (1+\varepsilon)C(x, y)$.

To conclude the proof, we consider the round complexity. We compute the weighted-defective coloring once at the beginning in the runtime given by Lemma 13.6. Then, for $O(\mathsf{cd}/\varepsilon^2)$ iterations, we need $O(\mathsf{cd}|\mathscr{L}|/\varepsilon^2)$ rounds to approximate weights, which results in the claimed runtime. ∎

We are now ready to implement the Ghaffari-Kuhn algorithm on the post-shattered instances.

Lemma 13.1 (Ghaffari-Kuhn Algorithm in Virtual Graphs)**.** *Let $\Delta \leqslant \mathrm{poly}(\log n)$. Let F be an N-vertex virtual graph with maximum degree $\Delta_F \leqslant O(\log n)$ on a network of bandwidth $\Theta(\log n)$. Suppose each vertex knows a list $L_v \subseteq [\Delta+1]$ of $\deg_F(v)+1$ colors. There exists a randomized algorithm that L_v-list-colors F with probability $1 - 1/\mathrm{poly}(n)$ in $O(\log N \cdot \log^6 \log n)$.*

Proof. Since $\Delta_F = O(\log n)$, we can properly $O(\log^2 n)$-color F in $O(1)$ rounds with probability $1 - 1/\mathrm{poly}(n)$ [HN23, Theorem 6.1]. When the color space is partitioned into $P_1, \ldots, P_K$, the Ghaffari-Kuhn algorithm turns the fractional labelling $x_{v,\ell} = \frac{|L_v \cap P_\ell|}{|P_\ell|}$ into a 2^{-b}-integral x' where $b = O(\log(KQ))$ without any communication. Since $y_{v,\ell} = \frac{1}{|L_v \cap P_\ell|}$ it can be made 2^{-b}-integral without increasing the cost by more than constant factor. By applying b times Lemma 13.7 times with $\varepsilon = \Theta(1/Qb)$, we obtain an integral x'' for which the cost has increased by a $(1 + O(1/Q))$ factor. Hence, after the Q levels of recursion, the total cost has increased

by a constant factor. As explained in [GK21], it induces a coloring of the vertices with $O(n)$ monochromatic edges. Considering the set of vertices incident to $O(1)$ monochromatic edges and running Linial on this $O(1)$-degree graph, we color a constant fraction of the vertices in $O(\mathsf{cd} \log^* \Delta)$ rounds. The total round complexity is

$$Qb \cdot K \cdot \left(\frac{\log\log n \cdot \log\log\log n}{\varepsilon} + \frac{1}{\varepsilon^4} \right) = O(\log^{5.5} \log n)$$

rounds, using that $Q = O(\log \Delta / \log\log \Delta)$, $K = O(\sqrt{\log \Delta})$, so $Qb = O(\log \Delta)$ and $\Delta \leqslant \text{poly}(\log n)$. Since we repeat this algorithm $O(\log N)$ times to color all vertices, the total round complexity is $O(\log N \cdot \log^6 \log n)$. Note that since $\Delta_F \leqslant O(\log n)$, it takes $O(\mathsf{cd})$ rounds to update the lists L_v of available colors between each iteration. ∎

CHAPTER 14

Conclusion

The initial ambition of this work was to understand which features of the computing and communication model make fast $(\Delta + 1)$-coloring feasible. While at the end of this thesis, we cannot fully characterize the frontier between models that would allow for various round complexities, we believe that this work represents substantial progress in this direction. In that regard, the central tenet of this conclusion would be that the efficiency of randomized $(\Delta + 1)$-coloring ultimately lies at the model's equilibrium between random color trials and synchronization.

In the LOCAL model, it had long been known that trying random colors is an effective strategy [Joh99]. In fact, the first important steps towards the current $\text{poly}(\log \log n)$ round LOCAL algorithm made by [SW10; EPS15; BEPS16] were mostly focused on harnessing the power of random color trials techniques. A key barrier was breached when the authors of [HSS18] realized that vertices for which random color trials are slow had a precise structure that could be exploited through synchronization. Later honed by [CLP20; HKMT21; HKNT22; HNT22], this insight lead to the $O(\log^* n)$ round shattering algorithm both in LOCAL and CONGEST.

Besides LOCAL and CONGEST, the palette sparsification theorem of [ACK19] was obtained through the same dichotomy. Sparse vertices are colored by random color trials while the almost-cliques get colored as a whole, thereby requiring a sequential and global approach. In CONGESTED-CLIQUE or MPC with $O(n)$ memory, the ability for synchronization is so strong that it suffices to color the graph entirely after a constant number of rounds [CFGUZ19; CDP21b]. But in MPC with sublinear space, the $O(\log \log \log n)$ round algorithm by [CDP21a] resorts back to the sparse-dense dichotomy, where the global capabilities of the model are used to speed-up exponentially that of [CLP20].

The technical core of this thesis centered on means to color almost-cliques given the ability of the model at hand, both in terms of color trials and synchronization inside almost-cliques. With this in mind, let us review the results of this thesis.

Part I: A Basic Algorithm. In Chapter 4, we presented an algorithm for $(\Delta+1)$-coloring graphs with $\Delta \geqslant \text{poly}(\log n)$ in $O(\log^* n)$ rounds in which vertices only broadcast one $O(\log n)$-bit message each round. It follows the general approach of the LOCAL algorithm (presented in Chapter 3), but introduces key technical ideas: a bandwidth efficient algorithm for computing the almost-clique decomposition, an additional step to compute a colorful matching, a version of SlackColor adapted to the broadcast constraint, and the idea of reserving colors during the first part of the algorithm. Observe that the majority of those modifications pertain to the color trials for the dense vertices. The BCONGEST algorithm uses the same technique for the synchronized color trial as in LOCAL, only the implementation differs. Finally, while coloring put-aside sets is trivial in LOCAL, in BCONGEST, it requires successive applications of color trials before centralization becomes feasible.

A strength of this algorithm is that, as it was designed for a weak model, it more easily adapted to various settings, such as the BC-Stream model (Chapter 5) or virtual graphs (Chapter 12).

Part II: Coloring with Oblivious Trials. In Chapter 6, we state the distributed palette sparsification theorem. Namely that if vertices sample lists of $O(\log^2 n)$ colors from $[\Delta+1]$, a distributed algorithm can color them in $O(\log^2 \Delta)$ rounds using only colors from those lists and communicating only on the edges between vertices whose lists intersect. This leads to a number of new results such as the first $\text{poly}(\log n)$ round algorithm for $(\Delta+1)$-coloring in NCC, or for coloring cluster graphs.

At the heart of this algorithm is a $(\Delta+1)$-coloring algorithm based on color trials only in $[\Delta+1]$. In LOCAL and even in BCONGEST, the algorithm could try colors from any well-choosen set of colors. Even worse, the synchronized color trial corrolated the trials of all vertices of the same almost-clique. On the contrary here, each vertex must use one of $O(\log^2 n)$

colors sampled from $[\Delta+1]$ before it has any knowledge of the graph. As a result, the algorithm presented in Chapter 7 runs in polylogarithmic rounds — much slower than that of Chapter 4 — but is applicable in models where synchronization is costly, like the NCC model.

In Chapter 8, we show that this complexity is nearly the best possible for our approach. Indeed, any algorithm that colors with $\text{poly}(\log n)$-sized lists of random colors while communicating on the induced sparsified graph requires $\Omega(\log \Delta / \log \log n)$ rounds to end.

Part III: Coloring an Unknown Graph. We concluded this thesis by studying scenarios in which the graph to be colored differs from the communication graph. Besides generalizing classical distributed graph coloring, this captures other previously studied settings, including cluster graphs and power graphs. In Chapter 9, we formalize the notion of embedding for virtual graphs and show that two natural parameters of those embeddings, the dilation and the edge congestion, are bottlenecks for coloring algorithms in Chapter 10.

We show that, although machines of the network know practically nothing of the graph to be colored except for the conflicts that they can see for themselves, virtual graphs can be colored in $O(\log^* n)$ rounds when Δ is polylogarithmic and in $\text{poly}(\log \log n)$ rounds otherwise. The algorithm for high-degree virtual graphs, presented over Chapters 11 and 12, follows the same scheme as the BCONGEST algorithm from Chapter 4 with key differences. Over virtual graphs, even simple tasks like computing the degree of a vertex become computationally expensive. So the first key technical ingredient introduced in Part III is an approximate counting technique called fingerprinting. Second, and perhaps the most novel part, is the coloring of put-aside sets, which requires a careful recoloring argument. Finally, our recoloring scheme requires that even very dense almost-cliques contain some repeated color, which color trials cannot guarantee. Such a requirement prompted the use of the fingerprinting for other purposes than counting, which demonstrates the versatility of the technique.

Unresolved Questions. The most pressing question regards truly sublogarithmic $(\Delta+1)$-coloring algorithms in the NCC model. In light of the severe

constraints the model imposes on how a vertex can synchronize with all its neighbors, the existence of an algorithm that is more efficient than the poly-logarithmic one obtained in this thesis would be surprising. Nonetheless, no lower bounds are known in this model and the $\Omega(\log \Delta / \log\log n)$ lower bound of Chapter 8 only applies to algorithm based on palette sparsification like ours.

A second fundamental question left open in this work is to determine whether distance-3 coloring (a coloring of G^3) can be obtained in time $O(\Delta^{1-o(1)})$. The virtual graph algorithm finds a distance-3 coloring in $O(\Delta \text{poly}(\log\log n))$ rounds (Corollary 9.11) and it is known that the linear dependency on Δ is necessary to try a color [FHN20]. Note that the $\Omega(\mathsf{c} + \mathsf{d}\log^* n)$ lower bound from Chapter 10 only applies to a specific class of instances that does not cover power graphs.

A myriad of other problems remain open. For instance, the BCONGEST algorithm presented in Chapter 4 is also the best-known $(\Delta+1)$-coloring algorithm in the broadcast congested clique, namely the CONGESTED-CLIQUE model where nodes are only allowed to broadcast one $O(\log n)$ bit message to every other vertex each round. What is the complexity of $(\Delta+1)$-coloring in this model? Can we take advantage of the global communications to obtain a speed-up over LOCAL? Or can one prove a lower bound? This thesis focused on the $(\Delta+1)$-coloring problem, but a lot of the applications of coloring in the literature on distributed graph algorithms deals with $(\deg+1)$-list-coloring, for which no algorithm even in BCONGEST is known. Therefore, a natural avenue for future research is to explore whether the techniques presented in this thesis can be applied in that setting. In [FHN24], we took a first step in this direction by solving the $(\widehat{\deg}+1)$-coloring problem in $\text{poly}(\log\log n)$ rounds on virtual graphs, but this problem is significantly simpler than its list-coloring variant. Finally, it would be interesting to know if the round complexity of coloring virtual graphs can be improved to $O((\mathsf{c}+\mathsf{d})\log^* n)$ rounds, which would then be nearly tight.

Bibliography

[AA20] Noga Alon and Sepehr Assadi. "Palette Sparsification Beyond (Δ+1) Vertex Coloring". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*. Vol. 176. LIPIcs. 2020. DOI: 10.4230/LIPICS.APPROX/RANDOM.2020.6 (cit. on pp. 45, 123, 124).

[ACK19] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. "Sublinear Algorithms for (Δ + 1) Vertex Coloring". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*. 2019. DOI: 10.1137/1.9781611975482.48 (cit. on pp. 9, 13, 19, 33–35, 37, 66, 67, 70, 71, 88, 118–121, 123–125, 138, 139, 142, 144, 146, 155, 235, 291, 319).

[AGGHSKL19] John Augustine, Mohsen Ghaffari, Robert Gmyr, Kristian Hinnenthal, Christian Scheideler, Fabian Kuhn, and Jason Li. "Distributed Computation in Node-Capacitated Networks". In: *The 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019, Phoenix, AZ, USA, June 22-24, 2019*. 2019. DOI: 10.1145/3323165.3323195 (cit. on pp. 5, 26, 119, 123).

[AGM12] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. "Analyzing graph structure via linear measurements". In: *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*. 2012. DOI: 10.1137/1.9781611973099.40 (cit. on pp. 12, 121, 123).

[AKM23] Sepehr Assadi, Pankaj Kumar, and Parth Mittal. "Brooks' Theorem in Graph Streams: A Single-Pass Semi-Streaming Algorithm for Δ-Coloring". In: *TheoretiCS* 2 (2023). DOI: 10.46298/THEORETICS.23.9 (cit. on pp. 123, 124).

[AKO18] Mohamad Ahmadi, Fabian Kuhn, and Rotem Oshman. "Distributed Approximate Maximum Matching in the CONGEST Model". In: *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*. Vol. 121. LIPIcs. 2018. DOI: 10.4230/LIPICS.DISC.2018.6 (cit. on p. 128).

[AKO20] Sepehr Assadi, Gillat Kol, and Rotem Oshman. "Lower Bounds for Distributed Sketching of Maximal Matchings and Maximal Independent Sets". In: *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*. 2020. DOI: 10.1145/3382734.3405732 (cit. on p. 66).

[AKZ22] Sepehr Assadi, Gillat Kol, and Zhijun Zhang. "Rounds vs Communication Tradeoffs for Maximal Independent Sets". In: *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*. 2022. DOI: 10.1109/FOCS54457.2022.00115 (cit. on pp. 66, 123).

[AMS96] Noga Alon, Yossi Matias, and Mario Szegedy. "The Space Complexity of Approximating the Frequency Moments". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. 1996. DOI: 10.1145/237814.237823 (cit. on p. 121).

[AW22] Sepehr Assadi and Chen Wang. "Sublinear Time and Space Algorithms for Correlation Clustering via Sparse-Dense Decompositions". In: *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*. Vol. 215.

LIPIcs. 2022. DOI: 10.4230/LIPICS.ITCS.2022.10 (cit. on p. 33).

[Bar16] Albert-László Barabási. *Network Science*. 2016 (cit. on p. 1).

[BBEHMOS20] Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. "Classification of Distributed Binary Labeling Problems". In: *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*. Vol. 179. LIPIcs. 2020. DOI: 10.4230/LIPICS.DISC.2020.17 (cit. on p. 63).

[BBKO22] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. "Distributed Δ-coloring plays hide-and-seek". In: *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*. 2022. DOI: 10.1145/3519935.3520027 (cit. on p. 9).

[Bec91] József Beck. "An Algorithmic Approach to the Lovász Local Lemma. I". In: *Random Struct. Algorithms* 2.4 (1991). DOI: 10.1002/RSA.3240020402 (cit. on pp. 6, 61).

[BEG22] Leonid Barenboim, Michael Elkin, and Uri Goldenberg. "Locally-iterative Distributed $(\Delta + 1)$-coloring and Applications". In: *J. ACM* 69.1 (2022). DOI: 10.1145/3486625 (cit. on p. 286).

[BEPS16] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. "The Locality of Distributed Symmetry Breaking". In: *J. ACM* 63.3 (2016). DOI: 10.1145/2903137 (cit. on pp. 6, 8, 18, 19, 61, 65, 66, 112, 122, 124, 200, 276, 277, 291).

[Ber23] Anton Bernshteyn. "Distributed algorithms, the Lovász Local Lemma, and descriptive combinatorics". In: *Inventiones mathematicae* 233.2 (2023). DOI: https://doi.org/10.1007/s00222-023-01188-3 (cit. on p. 26).

[BFHKLRSU16] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. "A lower bound for the distributed Lovász local lemma". In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. 2016. DOI: 10.1145/2897518.2897570 (cit. on pp. 9, 63).

[BJKST02] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. "Counting Distinct Elements in a Data Stream". In: *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings*. Vol. 2483. Lecture Notes in Computer Science. 2002. DOI: 10.1007/3-540-45726-7_1 (cit. on p. 121).

[Bla20] Jaroslaw Blasiok. "Optimal Streaming and Tracking Distinct Elements with High Probability". In: *ACM Trans. Algorithms* 16.1 (2020). DOI: 10.1145/3309193 (cit. on p. 200).

[BO90] F. Thomas Bruss and Colm Art O'Cinneide. "On the maximum and its uniqueness for geometric random samples". In: *Journal of applied probability* 27.3 (1990) (cit. on p. 227).

[Bol02] Béla Bollobás. *Modern Graph Theory*. Vol. 184. Graduate Texts in Mathematics. 2002. DOI: 10.1007/978-1-4612-0619-4 (cit. on pp. 130, 173).

[Bol11] Béla Bollobás. *Random Graphs, Second Edition*. Vol. 73. Cambridge Studies in Advanced Mathematics. 2011. DOI: 10.1017/CBO9780511814068 (cit. on p. 125).

[Bra17] Mark Braverman. "Interactive Information Complexity". In: *SIAM Rev.* 59.4 (2017). DOI: 10.1137/17M1139254 (cit. on pp. 209, 211).

[CCF02] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. "Finding Frequent Items in Data Streams". In:

Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings. Vol. 2380. Lecture Notes in Computer Science. 2002. DOI: 10.1007/3-540-45465-9_59 (cit. on p. 121).

[CDP21a] Artur Czumaj, Peter Davies, and Merav Parter. "Improved Deterministic (Δ+1) Coloring in Low-Space MPC". In: *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*. 2021. DOI: 10.1145/3465084.3467937 (cit. on pp. 26, 291).

[CDP21b] Artur Czumaj, Peter Davies, and Merav Parter. "Simple, Deterministic, Constant-Round Coloring in Congested Clique and MPC". In: *SIAM J. Comput.* 50.5 (2021). DOI: 10.1137/20M1366502 (cit. on pp. 8, 9, 25, 26, 291).

[CDP24] Artur Czumaj, Peter Davies-Peck, and Merav Parter. "Component stability in low-space massively parallel computation". In: *Distributed Comput.* 37.1 (2024). DOI: 10.1007/S00446-024-00461-9 (cit. on p. 9).

[CEFH25] Keren Censor-Hillel, Tomer Even, Maxime Flin, and Magnús M. Halldórsson. "When MIS and Maximal Matching are Easy in the Congested Clique". In: *Structural Information and Communication Complexity - 32nd International Colloquium, SIROCCO 2025, Delphi, Greece, June 2-4, 2025, Proceedings*. Vol. 15671. Lecture Notes in Computer Science. 2025. DOI: 10.1007/978-3-031-91736-3_12 (cit. on p. 20).

[CEKLM15] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. "One Trillion Edges: Graph Processing at Facebook-Scale". In: *Proc. VLDB Endow.* 8.12 (2015). DOI: 10.14778/2824032.2824077 (cit. on p. 2).

[CFGUZ19] Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. "The Complexity of (Δ+1)

Coloring in Congested Clique, Massively Parallel Computation, and Centralized Local Computation". In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*. 2019. DOI: 10.1145/3293611.3331607 (cit. on pp. 8, 9, 291).

[CK10] Alejandro Cornejo and Fabian Kuhn. "Deploying Wireless Networks with Beeps". In: *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings*. Vol. 6343. Lecture Notes in Computer Science. 2010. DOI: 10.1007/978-3-642-15763-9_15 (cit. on p. 5).

[CKP19] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. "An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model". In: *SIAM J. Comput.* 48.1 (2019). DOI: 10.1137/17M1117537 (cit. on pp. 62, 63).

[CKPWZ19] Yi-Jun Chang, Tsvi Kopelowitz, Seth Pettie, Ruosong Wang, and Wei Zhan. "Exponential Separations in the Energy Complexity of Leader Election". In: *ACM Trans. Algorithms* 15.4 (2019). DOI: 10.1145/3341111 (cit. on p. 5).

[CLP20] Yi-Jun Chang, Wenzheng Li, and Seth Pettie. "Distributed (Δ+1)-Coloring via Ultrafast Graph Shattering". In: *SIAM J. Comput.* 49.3 (2020). DOI: 10.1137/19M1249527 (cit. on pp. 7–10, 18, 19, 33, 34, 38, 48, 62, 65, 68, 124, 291).

[CM04] Graham Cormode and S. Muthukrishnan. "An Improved Data Stream Summary: The Count-Min Sketch and Its Applications". In: *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings*. Vol. 2976. Lecture Notes in Computer Science. 2004. DOI: 10.1007/978-3-540-24698-5_7 (cit. on p. 121).

[CP19] Yi-Jun Chang and Seth Pettie. “A Time Hierarchy Theorem for the LOCAL Model”. In: *SIAM J. Comput.* 48.1 (2019). DOI: 10.1137/17M1157957 (cit. on pp. 10, 26).

[CV86] Richard Cole and Uzi Vishkin. “Deterministic coin tossing and accelerating cascades: micro and macro techniques for designing parallel algorithms”. In: *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*. 1986. DOI: 10.1145/12130.12151 (cit. on p. 6).

[DF03] Marianne Durand and Philippe Flajolet. “Loglog Counting of Large Cardinalities (Extended Abstract)”. In: *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Proceedings*. Vol. 2832. 2003. DOI: 10.1007/978-3-540-39658-1_55 (cit. on p. 200).

[DG08] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: simplified data processing on large clusters”. In: *Commun. ACM* 51.1 (2008). DOI: 10.1145/1327452.1327492 (cit. on p. 2).

[DKO14] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. “On the power of the congested clique model”. In: *ACM Symposium on Principles of Distributed Computing, PODC ’14, Paris, France, July 15-18, 2014*. 2014. DOI: 10.1145/2611462.2611493 (cit. on p. 66).

[Doe20] Benjamin Doerr. “Probabilistic Tools for the Analysis of Randomized Optimization Heuristics”. In: *Theory of Evolutionary Computation - Recent Developments in Discrete Optimization*. Natural Computing Series. 2020. DOI: 10.1007/978-3-030-29414-4_1 (cit. on pp. 26, 27, 164, 316).

[DP09] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms*. 2009. DOI: 10.1017/CBO9780511581274 (cit. on pp. 26–29).

[Eis08] Bennett Eisenberg. “On the expectation of the maximum of IID geometric random variables”. In: *Statistics & Probability Letters* 78.2 (2008) (cit. on p. 227).

[EPS15] Michael Elkin, Seth Pettie, and Hsin-Hao Su. “(2Δ - l)-Edge-Coloring is Much Easier than Maximal Matching in the Distributed Setting”. In: *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*. 2015. DOI: 10.1137/1.9781611973730.26 (cit. on pp. 7, 34, 124, 291).

[FG17] Manuela Fischer and Mohsen Ghaffari. “Sublogarithmic Distributed Algorithms for Lovász Local Lemma, and the Complexity Hierarchy”. In: *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*. Vol. 91. LIPIcs. 2017. DOI: 10.4230/LIPICS.DISC.2017.18 (cit. on p. 15).

[FGGKR23] Salwa Faour, Mohsen Ghaffari, Christoph Grunau, Fabian Kuhn, and Václav Rozhon. “Local Distributed Rounding: Generalized to MIS, Matching, Set Cover, and Beyond”. In: *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*. 2023. DOI: 10.1137/1.9781611977554.CH168 (cit. on pp. 15, 25, 185, 286).

[FGHKN23] Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. “Coloring Fast with Broadcasts”. In: *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2023, Orlando, FL, USA, June 17-19, 2023*. 2023. DOI: 10.1145/3558481.3591095 (cit. on pp. 19, 35, 68, 88).

[FGHKN24] Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. “A Distributed Palette Sparsification Theorem”. In: *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*. 2024.

DOI: 10.1137/1.9781611977912.142 (cit. on pp. 20, 88).

[FGLPSY21] Sebastian Forster, Gramoz Goranci, Yang P. Liu, Richard Peng, Xiaorui Sun, and Mingquan Ye. "Minor Sparsifiers and the Distributed Laplacian Paradigm". In: *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. 2021. DOI: 10.1109/FOCS52979.2021.00099 (cit. on pp. 15, 185, 187, 190).

[FH25] Maxime Flin and Magnús M. Halldórsson. "Faster Dynamic (Δ+1)-Coloring Against Adaptive Adversaries". In: *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, July 8-11, 2025, Aarhus, Denmark*. Vol. 334. LIPIcs. 2025. DOI: 10.4230/LIPICS.ICALP.2025.79 (cit. on p. 21).

[FHK16] Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. "Local Conflict Coloring". In: *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*. 2016. DOI: 10.1109/FOCS.2016.73 (cit. on p. 8).

[FHN20] Pierre Fraigniaud, Magnús M. Halldórsson, and Alexandre Nolin. "Distributed Testing of Distance-k Colorings". In: *Structural Information and Communication Complexity - 27th International Colloquium, SIROCCO 2020, Paderborn, Germany, June 29 - July 1, 2020, Proceedings*. Vol. 12156. Lecture Notes in Computer Science. 2020. DOI: 10.1007/978-3-030-54921-3_16 (cit. on pp. 197, 294).

[FHN23] Maxime Flin, Magnús M. Halldórsson, and Alexandre Nolin. "Fast Coloring Despite Congested Relays". In: *37th International Symposium on Distributed Computing, DISC 2023, October 10-12, 2023, L'Aquila, Italy*. Vol. 281. LIPIcs. 2023. DOI: 10.4230/LIPICS.DISC.2023.19 (cit. on pp. 20, 33, 202, 224).

[FHN24] Maxime Flin, Magnús M. Halldórsson, and Alexandre Nolin. "Decentralized Distributed Graph Coloring II: Degree+1-Coloring Virtual Graphs". In: *38th International Symposium on Distributed Computing, DISC 2024, October 28 to November 1, 2024, Madrid, Spain*. Vol. 319. LIPIcs. 2024. DOI: 10.4230/LIPICS.DISC.2024.24 (cit. on pp. 20, 193, 194, 294).

[FHN25] Maxime Flin, Magnús M. Halldórsson, and Alexandre Nolin. "Decentralized Distributed Graph Coloring: Cluster Graphs". In: *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2025, Hotel Las Brisas Huatulco, Huatulco, Mexico, June 16-20, 2025*. 2025. DOI: 10.1145/3732772.3733549 (cit. on p. 20).

[Fis20] Manuela Fischer. "Improved deterministic distributed matching via rounding". In: *Distributed Comput.* 33.3-4 (2020). DOI: 10.1007/S00446-018-0344-4 (cit. on p. 281).

[FKMSZ05] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. "On graph problems in a semi-streaming model". In: *Theoretical Computer Science* 348.2 (2005). Automata, Languages and Programming: Algorithms and Complexity (ICALP-A 2004). DOI: https://doi.org/10.1016/j.tcs.2005.09.013 (cit. on p. 12).

[FM25] Maxime Flin and Parth Mittal. "$(\Delta + 1)$ vertex coloring in O(n) communication". In: *Distributed Comput.* 38.1 (2025). DOI: 10.1007/S00446-024-00475-3 (cit. on p. 20).

[FM85] Philippe Flajolet and G. Nigel Martin. "Probabilistic Counting Algorithms for Data Base Applications". In: *J. Comput. Syst. Sci.* 31.2 (1985). DOI: 10.1016/0022-0000(85)90041-8 (cit. on p. 200).

[GG23] Mohsen Ghaffari and Christoph Grunau. "Faster Deterministic Distributed MIS and Approximate Matching".

In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*. 2023. DOI: 10.1145/3564246.3585243 (cit. on p. 25).

[GG24] Mohsen Ghaffari and Christoph Grunau. "Near-Optimal Deterministic Network Decomposition and Ruling Set, and Improved MIS". In: *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*. 2024. DOI: 10.1109/FOCS61266.2024.00007 (cit. on pp. 8, 9, 25, 33, 62).

[GGHIR23] Mohsen Ghaffari, Christoph Grunau, Bernhard Haeupler, Saeed Ilchi, and Václav Rozhon. "Improved Distributed Network Decomposition, Hitting Sets, and Spanners, via Derandomization". In: *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*. 2023. DOI: 10.1137/1.9781611977554.CH97 (cit. on pp. 8, 15, 185).

[GGR21] Mohsen Ghaffari, Christoph Grunau, and Václav Rozhon. "Improved Deterministic Network Decomposition". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*. 2021. DOI: 10.1137/1.9781611976465.173 (cit. on pp. 8, 15, 65, 185, 187).

[GH16] Mohsen Ghaffari and Bernhard Haeupler. "Distributed Algorithms for Planar Networks II: Low-Congestion Shortcuts, MST, and Min-Cut". In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. 2016. DOI: 10.1137/1.9781611974331.CH16 (cit. on pp. 15, 119, 122, 133, 185).

[GK13] Mohsen Ghaffari and Fabian Kuhn. "Distributed Minimum Cut Approximation". In: *Distributed Computing - 27th International Symposium, DISC 2013, Jerusalem,*

Israel, October 14-18, 2013. Proceedings. Vol. 8205. Lecture Notes in Computer Science. 2013. DOI: 10.1007/978-3-642-41527-2_1 (cit. on pp. 15, 119, 122, 133, 185).

[GK21] Mohsen Ghaffari and Fabian Kuhn. "Deterministic Distributed Vertex Coloring: Simpler, Faster, and without Network Decomposition". In: *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. 2021. DOI: 10.1109/FOCS52979.2021.00101 (cit. on pp. 8, 25, 65, 66, 112, 120, 276, 283, 284, 288, 290).

[GKKLP18] Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. "Near-Optimal Distributed Maximum Flow". In: *SIAM J. Comput.* 47.6 (2018). DOI: 10.1137/17M113277X (cit. on pp. 15, 119, 122, 133, 185, 187).

[GKU19] Mohsen Ghaffari, Fabian Kuhn, and Jara Uitto. "Conditional Hardness Results for Massively Parallel Computation from Distributed Lower Bounds". In: *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*. 2019. DOI: 10.1109/FOCS.2019.00097 (cit. on p. 9).

[GLSS15] Dmitry Gavinsky, Shachar Lovett, Michael E. Saks, and Srikanth Srinivasan. "A tail bound for read-k families of functions". In: *Random Struct. Algorithms* 47.1 (2015). DOI: 10.1002/RSA.20532 (cit. on p. 29).

[GMT15] Sudipto Guha, Andrew McGregor, and David Tench. "Vertex and Hyperedge Connectivity in Dynamic Graph Streams". In: *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*. 2015. DOI: 10.1145/2745754.2745763 (cit. on p. 121).

[GR23] Jan Grebík and Václav Rozhoň. "Local problems on grids from the perspective of distributed algorithms,

finitary factors, and descriptive combinatorics". In: *Advances in Mathematics* 431 (2023). DOI: https://doi.org/10.1016/j.aim.2023.109241 (cit. on p. 26).

[GS17] Mohsen Ghaffari and Hsin-Hao Su. "Distributed Degree Splitting, Edge Coloring, and Orientations". In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. 2017. DOI: 10.1137/1.9781611974782.166 (cit. on p. 63).

[GZ22] Mohsen Ghaffari and Goran Zuzic. "Universally-Optimal Distributed Exact Min-Cut". In: *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*. 2022. DOI: 10.1145/3519270.3538429 (cit. on pp. 15, 119, 122, 133, 185).

[Hea08] Alexander Healy. "Randomness-Efficient Sampling within NC^1". In: *Comput. Complex.* 17.1 (2008). DOI: 10.1007/S00037-007-0238-5 (cit. on p. 112).

[HK73] John E. Hopcroft and Richard M. Karp. "An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs". In: *SIAM J. Comput.* 2.4 (1973). DOI: 10.1137/0202019 (cit. on p. 127).

[HKMT21] Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Tigran Tonoyan. "Efficient randomized distributed coloring in CONGEST". In: *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*. 2021. DOI: 10.1145/3406325.3451089 (cit. on pp. 8–11, 33, 38, 45, 65–67, 291).

[HKNT22] Magnús M. Halldórsson, Fabian Kuhn, Alexandre Nolin, and Tigran Tonoyan. "Near-optimal distributed degree+1 coloring". In: *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*. 2022. DOI: 10.1145/3519935.3520023 (cit. on pp. 18, 19, 33–35, 43, 48, 54, 62, 65, 68, 100, 120, 123, 124, 291).

[HMP24] Magnús M. Halldórsson, Yannic Maus, and Saku Peltonen. "Distributed Lovász Local Lemma under Bandwidth Limitations". In: *CoRR* abs/2405.07353 (2024). DOI: 10.48550/ARXIV.2405.07353. arXiv: 2405.07353 (cit. on p. 45).

[HN23] Magnús M. Halldórsson and Alexandre Nolin. "Superfast coloring in CONGEST via efficient color sampling". In: *Theor. Comput. Sci.* 948 (2023). DOI: 10.1016/J.TCS.2023.113711 (cit. on pp. 66, 69, 74, 75, 112, 225, 289).

[HNT21] Magnús M. Halldórsson, Alexandre Nolin, and Tigran Tonoyan. "Ultrafast Distributed Coloring of High Degree Graphs". In: *CoRR* abs/2105.04700 (2021). arXiv: 2105.04700 (cit. on pp. 19, 38).

[HNT22] Magnús M. Halldórsson, Alexandre Nolin, and Tigran Tonoyan. "Overcoming Congestion in Distributed Coloring". In: *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*. 2022. DOI: 10.1145/3519270.3538438 (cit. on pp. 8–11, 65–67, 69, 73, 74, 120, 291, 319, 320).

[HS20] Juho Hirvonen and Jukka Suomela. *Distributed algorithms*. 2020 (cit. on p. 25).

[HSS18] David G. Harris, Johannes Schneider, and Hsin-Hao Su. "Distributed (Δ +1)-Coloring in Sublogarithmic Rounds". In: *J. ACM* 65.4 (2018). DOI: 10.1145/3178120 (cit. on pp. 7, 8, 10, 18, 19, 33–35, 37, 38, 48, 65, 68, 124, 291).

[Ind01] Piotr Indyk. "A Small Approximately Min-Wise Independent Family of Hash Functions". In: *J. Algorithms* 38.1 (2001). DOI: 10.1006/JAGM.2000.1131 (cit. on p. 30).

[Joh99] Öjvind Johansson. "Simple Distributed *Delta*+1-coloring of Graphs". In: *Inf. Process. Lett.* 70.5 (1999). DOI: 10.1016/S0020-0190(99)00064-2 (cit. on pp. 6, 8, 11, 65, 122, 291).

[KLLRX15] Iordanis Kerenidis, Sophie Laplante, Virginie Lerays, Jérémie Roland, and David Xiao. "Lower Bounds on Information Complexity via Zero-Communication Protocols and Applications". In: *SIAM J. Comput.* 44.5 (2015). DOI: 10.1137/130928273 (cit. on p. 206).

[KLMMS14] Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. "Single Pass Spectral Sparsification in Dynamic Streams". In: *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*. 2014. DOI: 10.1109/FOCS.2014.66 (cit. on p. 121).

[KNW10] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. "An optimal algorithm for the distinct elements problem". In: *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*. 2010. DOI: 10.1145/1807085.1807094 (cit. on p. 200).

[KQ21] William Kuszmaul and Qi Qi. "The Multiplicative Version of Azuma's Inequality, with an Application to Contention Analysis". In: *CoRR* abs/2102.05077 (2021). arXiv: 2102.05077 (cit. on p. 27).

[KRZ21] Christian Konrad, Peter Robinson, and Viktor Zamaraev. "Robust Lower Bounds for Graph Problems in the Blackboard Model of Communication". In: *CoRR* abs/2103.07027 (2021). arXiv: 2103.07027 (cit. on p. 206).

[KS18] Ken-ichi Kawarabayashi and Gregory Schwartzman. "Adapting Local Sequential Algorithms to the Distributed Setting". In: *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*. Vol. 121. LIPIcs. 2018. DOI: 10.4230/LIPICS.DISC.2018.35 (cit. on p. 287).

[Kuh09] Fabian Kuhn. "Weak graph colorings: distributed algorithms and applications". In: *SPAA 2009: Proceedings of the 21st Annual ACM Symposium on Parallelism in*

Algorithms and Architectures, Calgary, Alberta, Canada, August 11-13, 2009. 2009. DOI: 10.1145/1583991.1584032 (cit. on pp. 286, 287).

[Lin92] Nathan Linial. "Locality in Distributed Graph Algorithms". In: *SIAM J. Comput.* 21.1 (1992). DOI: 10.1137/0221015 (cit. on pp. 3, 6, 63, 122, 205, 213, 286).

[LPPP05] Zvi Lotker, Boaz Patt-Shamir, Elan Pavlov, and David Peleg. "Minimum-Weight Spanning Tree Construction in $O(\log \log n)$ Communication Rounds". In: *SIAM J. Comput.* 35.1 (2005). DOI: 10.1137/S0097539704441848 (cit. on p. 5).

[Lub86] Michael Luby. "A Simple Parallel Algorithm for the Maximal Independent Set Problem". In: *SIAM J. Comput.* 15.4 (1986). DOI: 10.1137/0215074 (cit. on p. 65).

[Mot94] Rajeev Motwani. "Average-Case Analysis of Algorithms for Matchings and Related Problems". In: *J. ACM* 41.6 (1994). DOI: 10.1145/195613.195663 (cit. on p. 127).

[MR02] Michael Molloy and Bruce Reed. *Graph colouring and the probabilistic method*. Vol. 23. 2002 (cit. on p. 35).

[MR14] Michael Molloy and Bruce A. Reed. "Colouring graphs when the number of colours is almost the maximum degree". In: *J. Comb. Theory B* 109 (2014). DOI: 10.1016/J.JCTB.2014.06.004 (cit. on pp. 7, 45).

[MR85] Gary L. Miller and John H. Reif. "Parallel Tree Contraction and Its Application". In: *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*. 1985. DOI: 10.1109/SFCS.1985.43 (cit. on p. 63).

[MT20] Yannic Maus and Tigran Tonoyan. "Local Conflict Coloring Revisited: Linial for Lists". In: *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*. Vol. 179. LIPIcs. 2020. DOI: 10.4230/LIPICS.DISC.2020.16 (cit. on p. 286).

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. 2005. DOI: 10.1017/CBO9780511813603 (cit. on p. 6).

[New91] Ilan Newman. "Private vs. Common Random Bits in Communication Complexity". In: *Inf. Process. Lett.* 39.2 (1991). DOI: 10.1016/0020-0190(91)90157-D (cit. on pp. 74, 225).

[NS95] Moni Naor and Larry J. Stockmeyer. "What Can be Computed Locally?" In: *SIAM J. Comput.* 24.6 (1995). DOI: 10.1137/S0097539793254571 (cit. on pp. 6, 63, 205, 213).

[Pat07] Boaz Patt-Shamir. "A note on efficient aggregate queries in sensor networks". In: *Theor. Comput. Sci.* 370.1-3 (2007). DOI: 10.1016/J.TCS.2006.10.032 (cit. on p. 200).

[Pel00] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. 2000 (cit. on p. 4).

[PS97] Alessandro Panconesi and Aravind Srinivasan. "Randomized Distributed Edge Coloring via an Extension of the Chernoff-Hoeffding Bounds". In: *SIAM J. Comput.* 26.2 (1997). DOI: 10.1137/S0097539793250767 (cit. on p. 28).

[Ree98] Bruce A. Reed. "ω, Δ, and χ". In: *J. Graph Theory* 27.4 (1998). DOI: 10.1002/(SICI)1097-0118(199804)27:4\<177::AID-JGT1\>3.0.CO;2-K (cit. on pp. 7, 35, 37).

[RG20] Václav Rozhon and Mohsen Ghaffari. "Polylogarithmic-time deterministic network decomposition and distributed derandomization". In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*. 2020. DOI: 10.1145/3357713.3384298 (cit. on pp. 8, 15, 185, 187).

[RGHZL22] 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: *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*. 2022. DOI: 10.1145/3519935.3520074 (cit. on pp. 15, 119, 122, 133, 185).

[Roz24] Václav Rozhon. "An Invitation to Local Algorithms". In: *SIGACT News* (Dec. 2024). DOI: https://doi.org/10.1145/3710795.3710802. arXiv: 2406.19430 (cit. on pp. 26, 63).

[RY20] Anup Rao and Amir Yehudayoff. *Communication Complexity: and Applications*. 2020. DOI: 10.1017/9781108671644 (cit. on p. 209).

[Sha48] Claude Shannon. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* (1948) (cit. on p. 209).

[SHKKNPPW12] 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: *SIAM J. Comput.* 41.5 (2012). DOI: 10.1137/11085178X (cit. on p. 5).

[SW10] Johannes Schneider and Roger Wattenhofer. "A new technique for distributed symmetry breaking". In: *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing, PODC 2010, Zurich, Switzerland, July 25-28, 2010*. 2010. DOI: 10.1145/1835698.1835760 (cit. on pp. 7, 19, 33, 34, 48, 67, 124, 291).

[Vad12] Salil P. Vadhan. "Pseudorandomness". In: *Foundations and Trends® in Theoretical Computer Science* 7.1–3 (2012). DOI: 10.1561/0400000010 (cit. on pp. 29, 30).

APPENDIX A

Deferred Proofs

A.1. Sparsity & Almost-Clique Decomposition

Fact 3.3. *A non-isolated vertex with degree at most $\Delta - x$ is $(x/2)$-sparse.*

PROOF. Given the bound on the degree, the induced neighborhood contains at most $\binom{\Delta-x}{2}$ edges and the vertex is ζ-sparse for

$$\zeta = \frac{1}{\Delta}\left(\binom{\Delta}{2} - \binom{\Delta - x}{2}\right) = \frac{1}{2\Delta}\left(2x\Delta - x^2 - x\right) \geqslant x/2$$

using that $0 \leqslant x \leqslant \Delta - 1$ because v is not isolated. ∎

Fact 3.16. *If v has degree at least $\Delta - x \geqslant 1$, then $G[N(v)]$ contains at least $(\zeta(v) - x/2)\Delta$ anti-edges.*

PROOF. The number of anti-edges in $G[N(v)]$ is

$$\begin{aligned}
&\binom{\deg(v)}{2} - \#\text{ edges in } G[N(v)] \\
&\qquad = \zeta(v)\Delta + \binom{\deg(v)}{2} - \binom{\Delta}{2} \\
&\qquad = \zeta(v)\Delta - \frac{x\Delta}{2}(2 - x/\Delta - 1/\Delta) \\
&\qquad \geqslant \zeta(v)\Delta - x\Delta/2
\end{aligned}$$

using that $0 \leqslant x \leqslant \Delta - 1$ because v is not isolated. ∎

Lemma 3.8. *Every $v \in V_{dense}$ is $(\varepsilon/2^{14})(e(v) + a(v))$-sparse.*

PROOF. Let V_{sparse}, V_{dense} is the ε-almost-clique decomposition described in Proposition 3.6. Fix $v \in K$. If $a(v)/2 \leqslant e(v)$ since, by Part 3 in Proposition 3.6, v is $(\varepsilon/2^{12})e(v)$-sparse, we have that v is $\varepsilon(e(v)+a(v))/2^{14}$-sparse. If $a(v)/2 \geqslant e(v)$ and $\deg(v) \leqslant \Delta - a(v)/4$, then v is $a(v)/8$-sparse by Fact 3.3, thus it is $(e(v) + a(v))/16$-sparse.

Assume henceforth that $a(v)/2 \geqslant e(v)$ and $\deg(v) \geqslant \Delta - a(v)/4$. Let us lower bound the number of anti-edges in $G[N(v) \cap K]$. By handshake lemma, the number of edges in $G[N(v) \cap K]$ is

$$\frac{1}{2} \sum_{w \in N(v) \cap K} |N(v) \cap N(w) \cap K| \ . \tag{A.1}$$

Using that $|N(v) \cap N(w) \cap K| = |N(w) \cap K| - 1 - |N(w) \cap A(v)|$, it suffices that we lower bound the number of edges between $N(v) \cap K$ and $A(v)$. By rearranging the sum of $|N(w) \cap A(v)|$ over $w \in N(v) \cap K$ as a sum over anti-neighbors of v as

$$\sum_{w \in N(v) \cap K} |N(w) \cap A(v)| = \sum_{u \in A(v)} |N(v) \cap N(u)| \geqslant a(v)(1-\varepsilon)\Delta \ ,$$

where the last inequality uses Definition 3.5-(2) as u and v are from the same almost-clique. Plugging this into Eq (A.1), we obtain that the number of edges in $G[N(v) \cap K]$ is at most

$$\frac{|N(v) \cap K|(\Delta - 1)}{2} - (1-\varepsilon)a(v)\Delta/2 \ .$$

To count the anti-edges in $G[N(v) \cap K]$, we compare the first term to the number of pairs of neighbors in K as

$$\frac{|N(v) \cap K|(\Delta - 1)}{2} - \binom{|N(v) \cap K|}{2} = \frac{|N(v) \cap K|}{2}(\Delta - |N(v) \cap K|) \ . \tag{A.2}$$

Since $\deg(v) = |N(v) \cap K| + e(v) \geqslant \Delta - a(v)/4$, we have that $\Delta - |N(v) \cap K| \leqslant e(v) + a(v)/4 \leqslant 3a(v)/4$ by assumption. Hence Eq (A.2) is at most $3a(v)\Delta/8$ and we can conclude that $G[N(v) \cap K]$ contains at most

$$\binom{|N(v) \cap K|}{2} - (1/2 - 3/8 - \varepsilon)a(v)\Delta$$

edges. It follows that $G[N(v) \cap K]$ contains at least $(1/8 - \varepsilon)a(v)\Delta \geqslant a(v)\Delta/8$ anti-edges, for $\varepsilon \leqslant 1/4$. Hence v is $(e(v) + a(v))/16$-sparse because $a(v) \geqslant e(v)$. ∎

Lemma A.1. *Suppose $V_{sparse}, C_1, C_2, \ldots, C_r$ is a vertex partition such that every vertex in V_{sparse} is $(\varepsilon^2 \cdot \delta\Delta)$-sparse and each C_i is an ε-almost-clique. Then every $v \in C = C_i$ is at least $(\varepsilon^2 \cdot \delta e(v) - 1)/2$-sparse.*

PROOF. We may assume that $G[N(v)]$ contains at least $\binom{\Delta}{2} - \delta\varepsilon^2/2 \cdot e(v)\Delta$ edges, for the claim otherwise trivially holds. By handshaking lemma, the number of edges in $G[N(v)]$ is

$$\begin{aligned}\frac{1}{2}\sum_{u\in N(v)} |N(u)\cap N(v)| &\leqslant \frac{1}{2}\sum_{u\in N(v)} \Delta - |N(v)\setminus N(u)| \\ &\leqslant \binom{\Delta}{2} + \frac{\Delta}{2} - \sum_{u\in N(u)} |N(v)\setminus N(u)|\,.\end{aligned} \tag{A.3}$$

So we lower bound $|N(v)\setminus N(u)|$ for each $u \in E(v)$. If $u \in C_j \neq C$, then it has at most $\varepsilon\Delta$ shared neighbors with v, so $|N(v)\setminus N(u)| \geqslant (1-\varepsilon)\Delta$. If $u \in V_{sparse}$, then $G[N(u)]$ contains at most $\binom{\Delta}{2} - \delta\varepsilon^2\Delta$ many edges by definition. On the other hand, of the edges in $G[N(v)]$, at most $|N(v)\setminus N(u)|\Delta$ are not in $G[N(u)]$. Using our assumption on the number of edges in $G[N(v)]$, this means that $\binom{\Delta}{2} - \delta\varepsilon^2/2 \cdot e(v)\Delta \leqslant \binom{\Delta}{2} - \delta\varepsilon^2\Delta^2 + |N(v)\setminus N(u)|\Delta$, so that $|N(v)\setminus N(u)| \geqslant \delta\varepsilon^2/2\cdot\Delta$. Since all external neighbors contribute at least $\delta\varepsilon^2/2\cdot\Delta$ to the sum in Eq (A.3), the number of edges in $G[N(v)]$ is at most $\binom{\Delta}{2} - (\delta\varepsilon^2/2\cdot e(v) - 1/2)\Delta$. ∎

A.2. Expansion of the Sparsified Almost-Clique

Lemma 7.5 (Expansion)**.** *Let K be an almost-clique, and assume $\Delta \geqslant \beta^4$. With high probability, for all subsets $S \subseteq K$ of size at most $|S| \leqslant 3\Delta/4$,*

$$|E_{\widetilde{K}}(S, K\setminus S)| \geqslant |S|\beta^4/3200 \qquad \textit{and} \qquad |N_{\widetilde{K}}(S)\cap(K\setminus S)| \geqslant \Omega(|S|).$$

PROOF OF LEMMA 7.5. We show that for any fixed set $S \subseteq K$ with $|S| \leqslant 3\Delta/4$, the edge expansion property holds w.e.h.p. in $|S|\beta$. Since there are at most $|K|^x = \exp(O(x\log\Delta))$ subsets $S \subseteq K$ of size x, by choosing $\beta = \Theta(\log n)$ with a large enough constant, it holds w.h.p. for all sets S of size x by union bound. And thus for all sets of size at most $3\Delta/4$. Since the maximum degree in $\widetilde{G}$ is $O(\beta^4)$, the number of nodes in $\overline{S}$ is at least $|E_{\widetilde{K}}(S,\overline{S})|/O(\beta^4) = \Omega(|S|)$.

Henceforth $S \subseteq K$ is a fixed subset of size at most $3\Delta/4$ and set $\overline{S} \stackrel{\text{def}}{=} K\setminus S$ of size at least $\Delta/4 - \varepsilon\Delta = \Delta/5$. In total, $E_G(S,\overline{S})$ contains between $|S|\Delta/5$ and $|S|\Delta$ edges because each node $v \in S$ has at least $|N_G(v)\cap\overline{S}| \geqslant |\overline{S}| - 2\varepsilon\Delta \geqslant |K| - |S| - 2\varepsilon\Delta \geqslant (1/4 - 3\varepsilon)\Delta \geqslant \Delta/5$ neighbors in $\overline{S}$ before

sparsification — and at most Δ. Our analysis has two cases, depending on the size of S.

Each vertex v samples each color into $\mathbf{L}(v)$ independently with probability β^2/Δ.

Suppose $|S| \leqslant \beta^2$. For each $u \in S$, the classic Chernoff Bound shows that $|\mathbf{L}(u)| \leqslant \beta^2/2$ with probability at most $q = \exp(-\Omega(\beta^2))$. Define $\mathbf{X}_u$ as the indicator random variable for that event and $\mathbf{X} = \sum_{u \in S} \mathbf{X}_u$. Applying the tighter version of the Chernoff Bound [Doe20, Theorem 1.10.1, Eq. 1.10.2] with $\delta = 1/(2q) - 1$, we obtain

$$\mathbb{P}[\mathbf{X} \geqslant |S|/2] = \mathbb{P}\Big[\mathbf{X} \geqslant (1+\delta)\,\mathbb{E}[\mathbf{X}]\Big] \leqslant \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{|S|q}$$

and since $\ln(1+\delta) = \ln(1/2q) = \Theta(\beta^2) \geqslant 2$, this is at most

$$\mathbb{P}[\mathbf{X} \geqslant |S|/2] \leqslant \exp\big(-|S|q \cdot \Theta(\delta\beta^2)\big) \leqslant \exp\big(-\Theta(|S|\beta^2)\big)\ .$$

In other words, w.e.h.p. in $|S|\beta^2$ over $\mathbf{L}(S)$, there are at least $|S|/2$ vertices in S with a list of size at least $\beta^2/2$. Let us condition on such lists $\{L(u) : u \in S\}$ and call $S' \subseteq S$ the set of at least $|S|/2$ vertices with $|L(u)| \geqslant \beta^2/2$. Now, let us define $\mathbf{Z}_v$ for $v \in \overline{S}$ as the number of $u \in N_G(v) \cap S'$ such that $L(u) \cap \mathbf{L}(v) \neq \emptyset$. The random variable $\mathbf{Z} = \sum_{v \in \overline{S}} \mathbf{Z}_v$ counts the number of edges in $E_{\widetilde{K}}(S', \overline{S})$. By linearity of the expectation, for all $v \in \overline{S}$, we have that

$$\mathbb{E}[\mathbf{Z}_v] \geqslant |S'|\left(1 - \left(1 - \frac{\beta^2}{\Delta}\right)^{\beta^2/2}\right) \geqslant |S'|\frac{\beta^4}{\beta^4 + 2\Delta} \geqslant |S'|\frac{\beta^4}{3\Delta}$$

(using $\beta^4 \leqslant \Delta$ and $(1-x)^y \leqslant 1/(1+xy)$ for $x \in [0,1]$ and $y > 0$)

Since $\overline{S}$ contains at least $\Delta/5$ vertices, the expected number of edges in the sparsified graph is at least $|S'|\beta^4/3 \geqslant |S|\beta^4/6 = \mu$. As each $\mathbf{Z}_v \leqslant |S| \leqslant \beta^2$ by assumption, the Chernoff Bound gives

$$\mathbb{P}[\mathbf{Z} \leqslant \mu/2] \leqslant \exp\left(-\frac{|S'|\beta^4/6}{8\beta^2}\right) \leqslant \exp\big(-\Omega\big(|S|\beta^2\big)\big)\ ,$$

which concludes the proof when $|S| \leqslant \beta^2$.

Suppose $|S| \geqslant \beta^2$. We reveal colors in $\lceil(\Delta+1)/b\rceil$ batches, where $b = \lfloor\Delta/|S|\rfloor$, which is always at least one because $|S| \leqslant 3\Delta/4$. The i-th batch is the set of colors $B_i = [(i-1)b + 1, ib]$ and denote by $\mathbf{B}_i = \mathbf{L}(K) \cap B_i$

the random variable containing all the randomness of the i-th batch for all vertices of K (it is a random vector indexed by $K \times B_i$ for which each bit is independently set to one with probability β^2/Δ). We say that the edge $\{u, v\} \in E_G$ where $u \in S$ and $v \in \overline{S}$ is introduced by the i-th batch if $\mathbf{L}(u) \cap \mathbf{L}(v) \neq \emptyset$ and $\min(\mathbf{L}(u) \cap \mathbf{L}(v)) \in B_i$. For all $i \in [\lceil (\Delta+1)/b \rceil]$, let $\mathcal{E}_i$ be the event that colors $\{1, 2, \ldots, ib\}$ induce at least $|S|\beta^4/20$ edges in the sparsified graph. Define $\mathbf{Y}_i$ as the random variable equal to 0 when $\mathcal{E}_{i-1}$ holds and, when $\mathcal{E}_{i-1}$ does not hold, $\mathbf{Y}_i = 1$ if the i-th batch introduces fewer than $t = \beta^4/1600$ edges and $\mathbf{Y}_i = 0$ otherwise. We shall prove that

$$\mathbb{E}[\mathbf{Y}_i \mid \mathbf{B}_1, \ldots, \mathbf{B}_{i-1}] \leqslant \exp(-\Omega(\beta^2)) \; . \tag{A.4}$$

Consider a fixed $i \in [\lceil (\Delta + 1)/b \rceil]$. If $\mathbf{B}_1, \ldots, \mathbf{B}_{i-1}$ is set so that $\mathcal{E}_{i-1}$ holds, then $\mathbf{Y}_i = 0$ with probability one. So fix $\mathbf{B}_1, \ldots, \mathbf{B}_{i-1}$ so that $\mathcal{E}_{i-1}$ does *not* hold. Call $E_i \subseteq E_G(S, \overline{S})$ the set of edges that are *not* introduced by one of the first $i-1$ batches, i.e., the edges $\{u, v\}$ for which $|\mathbf{L}(u) \cap \mathbf{L}(v) \cap [(i-1)b]| = \emptyset$. Observe that by assuming $\overline{\mathcal{E}_{i-1}}$ holds, we have that $|E_i| \geqslant |S|\Delta/10$. This means that at least $|S|/20$ vertices of S are incident to at least $\Delta/20$ edges in E_i. Call $S' \subseteq S$ the set of such vertices. For each $u \in S'$, call $\mathbf{X}_u$ the random variable indicating if u samples at least one color in B_i. We have that

$$\mathbb{E}[\mathbf{X}_u \mid \mathbf{B}_1, \ldots, \mathbf{B}_{i-1}, \overline{\mathcal{E}_{i-1}}] \geqslant 1 - \left(1 - \frac{\beta^2}{\Delta}\right)^b \geqslant \frac{\beta^2}{\beta^2 + |S|} \geqslant \frac{\beta^2}{2|S|} \; .$$

(using $\beta^2 \leqslant |S|$ and $(1-x)^y \leqslant 1/(1+xy)$ for $x \in [0,1]$ and $y > 0$)

Conversely, by a union bound, we have $\mathbb{E}[\mathbf{X}_u | \mathbf{B}_1, \ldots, \mathbf{B}_{i-1}, \overline{\mathcal{E}_{i-1}}] \leqslant \beta^2 b/\Delta = \beta^2/|S|$. The $\{\mathbf{X}_u : u \in S\}$ are independent (even given the conditioning), hence the Chernoff Bound shows that, w.e.h.p. in β^2 over the randomness of $\mathbf{L}(S) \cap B_i$ when $\overline{\mathcal{E}_{i-1}}$, between $\beta^2/40$ and $2\beta^2$ vertices in S' sample a color in B_i. We henceforth fix the sampling of colors $\{L(u) \cap B_i : u \in S\}$ such that this holds and call $S'' \subseteq S$ the set of at least $\beta^2/40$ and at most $2\beta^2$ vertices that sampled a color in B_i.

For $v \in \overline{S}$, define $\mathbf{Z}_v$ as the number of vertices in $u \in S''$ for which $\{u, v\} \in E_i$, i.e., the edge is added to the sparsified graph at batch i. Thus, we have that $\mathbf{Z}_i = \sum_{v \in \overline{S}} \mathbf{Z}_v$. To lower bound its expectation over the

randomness of $\mathbf{L}(\overline{S}) \cap B_i$ given our conditioning, define $\mathbf{Z}_{u,v}$ for all pairs $\{u, v\} \in E_i$ with $u \in S''$. Clearly, $\mathbb{E}[\mathbf{Z}_{u,v} \mid \mathbf{B}_1, \ldots, \mathbf{B}_{i-1}, \mathbf{L}(S) \cap B_i, \overline{\mathcal{E}_{i-1}}] \geqslant \beta^2/\Delta$ as it is the probability that v samples the smallest color of $L(u) \cap B_i$. By linearity of the expectation,

$$\begin{aligned}\mathbb{E}[\mathbf{Z}_i \mid \mathbf{B}_1, \ldots, \mathbf{B}_{i-1}, \mathbf{L}(S) \cap B_i, \overline{\mathcal{E}_{i-1}}] \\ &= \sum_{u \in S''} \sum_{v \in \overline{S}} \mathbb{E}[\mathbf{Z}_{u,v} \mid \mathbf{B}_1, \ldots, \mathbf{B}_{i-1}, \mathbf{L}(S) \cap B_i, \overline{\mathcal{E}_{i-1}}] \\ &\geqslant |S''|\beta^2/20 \geqslant \beta^4/800 \ ,\end{aligned}$$

using the lower bound on $|S''|$ and that each $u \in S''$ is incident to at least $\Delta/20$ vertices in $v \in \overline{S}$ in E_i. Seeing $\mathbf{Z}_i$ as the sum of independent random variables $\mathbf{Z}_v$ with values in $[0, 2\beta^2]$, we may apply the Chernoff Bound and obtain that $\mathbf{Z}_i \geqslant t \stackrel{\text{def}}{=} \beta^4/1600$ w.e.h.p. in β^2. Overall, we have shown that

$$\begin{aligned}\mathbb{P}\big[\mathbf{Y}_i = 1 \mid \mathbf{B}_1, \ldots, \mathbf{B}_{i-1}, \mathbf{L}(S) \cap B_i, \overline{\mathcal{E}_{i-1}}\big] \\ &= \mathbb{P}[\mathbf{Z}_i < t \mid \mathbf{B}_1, \ldots, \mathbf{B}_{i-1}, \mathbf{L}(S) \cap B_i, \overline{\mathcal{E}_{i-1}}] \\ &\leqslant 2\exp(-\Omega(\beta^2))\end{aligned}$$

for all $i \in [\lceil(\Delta+1)/b\rceil]$. This concludes the proof of Eq (A.4).

Now, let $q = \exp(-\Omega(\beta^2))$ as in Eq (A.4) and apply the Chernoff Bound with domination on the sum of the $\mathbf{Y}_i = f_i(\mathbf{B}_1, \ldots, \mathbf{B}_i)$, $\mu = q\Delta/b = q|S|$ and $\delta = (1/2q - 1) = \exp(\Theta(\beta^2))$, we obtain

$$\begin{aligned}\mathbb{P}\left[\sum_i \mathbf{Y} \geqslant |S|/2\right] = \mathbb{P}\left[\sum_i \mathbf{Y} \geqslant (1+\delta)\mu\right] &\leqslant \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu \\ &\leqslant \exp(-\Omega(|S|\beta^2)) \ ,\end{aligned}$$

where the second inequality holds because the term between the parenthesis is asymptotically $\exp(-\Theta((1/q)\ln(1/q))) = \exp(-\Theta(\beta^2/q))$.

Consider lists for which the sum of $\mathbf{Y}_i$ is at most $|S|/2$. If $\mathcal{E}_{\Delta/b}$ holds, by definition the sparsified graph has enough edges. If $\mathcal{E}_{\Delta/b}$ does not hold, then neither does any of the $\mathcal{E}_i$ for $i \in [\Delta/b]$. As such, the sum of the $\mathbf{Y}_i$ counts the number of batches contributing at most t edges. Since at least half of the batches contribute at least t edges, the total number of edges introduced in the sparsified almost-clique is at least $|S|/2 \cdot t = S\beta^4/3200$.

∎

A.3. Almost-Clique Decomposition on the Sparsified Graph

First, we explain how we detect friendly edges. We emphasize that friendliness always refers to G, i.e., an edge $\{u, v\}$ is δ-friendly if $|N_G(v) \cap N_G(u)| \geqslant (1-\delta)\Delta$. The technique of [ACK19] requires each vertex to broadcast the ID of $O(\log n)$ random neighbors; hence works with $O(\log^2 n)$ bandwidth. The following algorithm combines the idea from [ACK19] with an algorithm of [HNT22] to reduce the bandwidth to $O(\log n)$. We use the following result:

Lemma A.2 ([HNT22, Lemma 2])**.** *Suppose every vertex holds a set $S_v \subseteq \mathscr{U}$. There exists an algorithm called* EstimateSimilarity(δ) *at the end of which, w.h.p., both endpoints of all edges $\{u, v\}$ know an estimator*

$$\tilde{s}(u, v) \in |S_v \cap S_v| \pm \delta \max(|S_u|, |S_v|) \ .$$

It uses $O(1)$ messages of $O(\delta^{-4} \log n + \log\log|\mathscr{U}| + \log \max_v |S_v|)$ bits.

Algorithm 40. Algorithm detecting friendly edges in $\widetilde{G}$.

Input: a graph G with n vertices and maximum degree Δ
Output: a set $E_\delta \subseteq E_G$

(1) Each vertex samples colors of $[\Delta+1]$ in a set $\mathbf{S}(v)$ independently with probability $p = \Theta(\delta^{-2} \log n/\Delta)$.
(2) For each edge $\{u, w\}$ where $\mathbf{S}(u) \cap \mathbf{S}(w) \neq \emptyset$, pick an arbitrary $\chi \in \mathbf{S}(u) \cap \mathbf{S}(w)$ and call $U = \{v \in N(u) : \chi \in \mathbf{S}(v)\}$ and $W = \{v \in N(w) : \chi \in \mathbf{S}(v)\}$. They run the algorithm EstimateSimilarity($\delta/32$) to get an estimator
$$\tilde{s}(u, w) \in |U \cap W| \pm (\delta/32)|U \cup W| \ .$$
(3) Add in E_δ every edge $\{u, w\}$ with $\mathbf{S}(u) \cap \mathbf{S}(w) \neq \emptyset$ such that $|\tilde{s}(u, w)| \geqslant (1 - 0.75\delta)p\Delta$.

Lemma A.3. *Let $\delta \in (0, 1)$. Suppose every vertex of an n-node graph G with maximum degree $\Delta \gg \delta^{-2} \log n$ samples a list of $O(\delta^{-2} \log n)$ colors u.a.r. in $[\Delta + 1]$. Algorithm 40 computes in $O(\delta^{-4} + \delta^{-2} \log \Delta)$ rounds a set E_δ of edges in the sparsified graph such that all $(\delta/2)$-friendly edges (in*

G) sampled in $\widetilde{G}$ belong to E_δ and all edges of E_δ are at least δ-friendly (in G). Vertices communicate only on the edges of the sparsified graph.

PROOF. In Step (1), all vertices broadcast their sampled colors to neighbors $u \in N(v)$ with $\mathbf{S}(u) \cap \mathbf{S}(v) \neq \emptyset$. By Chernoff, w.h.p., all sets $\mathbf{S}(v)$ contain $O(\delta^{-2} \log n)$ colors. So $\mathbf{S}(v)$ takes one message of $O(\log \Delta \cdot \delta^{-2} \log n)$ bits to describe, which thus takes $O(\delta^{-2} \log \Delta)$ rounds to broadcast. In Step (2), by Lemma A.2, running EstimateSimilarity takes $O(\delta^{-4})$ rounds.

Consider some edge $\{u, w\}$ with $\mathbf{S}(u) \cap \mathbf{S}(w) \neq \emptyset$ and let χ be their selected color. Define $\mathbf{X}_v$ for $v \in N(u) \cap N(w)$ to be one iff $\chi \in \mathbf{S}(v)$, so that $|U \cap W| = \sum_v \mathbf{X}_v$. Since χ is sampled in $\mathbf{S}(v)$ w.p. p independently for each v, we expect $\mathbb{E}[\sum_v \mathbf{X}_v] = p|N(u) \cap N(w)|$ and by Chernoff, w.e.h.p. $\delta^2 \cdot p\Delta$, we have that $|U \cap W| \in (1 \pm \delta/8)p|N(u) \cap N(w)|$. So, by Lemma A.2, w.h.p., we have that

$$\tilde{s}(u, w) \in (1 \pm \delta/8)p|N(u) \cap N(w)| \pm (\delta/32)|U \cup W|$$

Since $|U \cup W| \leqslant |U| + |W| \leqslant 4p\Delta$, w.h.p., this becomes

$$\tilde{s}(u, w) \in (1 \pm \delta/8)p|N(u) \cap N(w)| \pm (\delta/8)p\Delta \ .$$

Thus, when $\{u, w\}$ is a $(\delta/2)$-friendly edge, we have that $\tilde{s}(u, w) \geqslant (1 - \delta/8)(1 - \delta/2)p\Delta - (\delta/8)p\Delta \geqslant (1 - 0.75\delta)p\Delta$. Conversely, when $\{u, w\}$ is not δ-friendly, we have that $\tilde{s}(u, w) < (1 + \delta/8)(1 - \delta)p\Delta + (\delta/8)p\Delta \leqslant (1 - 0.75\delta)p\Delta$. ∎

Remark A.4. The algorithm of Lemma A.2 is based on the existence of a universal family of hash function for which the authors provide no explicit construction (see [HNT22, Lemma 1]). To the best of our knowledge, it requires $\Omega(|S_u| + |S_v|)$ space — so $\Omega(\Delta)$ space in Algorithm 40 — to compute a hash given its $\Theta(\log n)$-bit representation. In particular, this makes Algorithm 40 unsuitable for the Dist-Stream model. Recall however that in Dist-Stream, Algorithm 19 works with $\text{poly}(\log n)$ memory, albeit communicating on few edges outside of the sparsified graph.

Lemma A.5. *After Algorithm 40, w.h.p., vertices can compute in $O(1)$ rounds a set V_δ that contains all $(\delta/2)$-popular vertices and only δ-popular vertices. Vertices communicate only on the edges of the sparsified graph.*

PROOF. With high probability $\mathbf{S}(v) \neq \emptyset$, so $\mathbf{S}(v)$ and let χ be its smallest color. Recall that $F_\delta(v)$ denotes the set of $u \in N(v)$ such that $\{u, v\}$ is δ-friendly. Suppose v is $(\delta/2)$-popular, i.e., that $|F_{(\delta/2)}(v)| \geqslant (1 - \delta/2)\Delta$. Each $u \in F_{(\delta/2)}(v)$ samples χ independently with probability p, so by the Chernoff Bound, w.e.h.p. in $\delta^2 p\Delta$, we have that at least $(1 - 3\delta/4)p\Delta$ vertices in $u \in F_{(\delta/2)}(v)$ sample χ. Conversely, if v is not δ-popular, i.e., $|F_\delta(v)| < (1 - \delta)\Delta$, at most $(1 - 3\delta/4)p\Delta$ vertices $u \in F_\delta(v)$ sampled χ.

By union bound, both this and the claim of Lemma A.3 hold with high probability. If v is $(\delta/2)$-popular, it has at least $(1 - 3\delta/4)p\Delta$ edges in E_δ to vertices that sampled χ. If v is not δ-popular, then it is incident to fewer than $(1 - 3\delta/4)p\Delta$ edges in E_δ such that the other endpoint sampled χ. ∎

A.4. MultiColor Trials on the Sparsified Graph

From Section 3.4, it suffices to provide an implementation of MULTICOLORTRIAL such that Lemma 3.20 holds. We use Algorithm 3 except that instead of sampling x colors, we sample $O(x/\delta)$ colors, to guarantee that at least x sampled colors belong to $L_\varphi(v)$ w.e.h.p. in x. Lemma 7.17 is then obtained by plugging Lemma A.6 in Lemma 3.22 for phases (2) and (3). The total number of colors sampled by SLACKCOLOR when it uses Algorithm 41 is $\sum_i O(x_i/\delta) = O(\delta^{-1} \log n)$ because the sum of the x_i is dominated by its last term which is $O(\log n)$. Similarly, since the i-th call to Algorithm 41 requires to broadcast $O(x_i/\delta)$ colors, the number of rounds required by SLACKCOLOR is $\sum_i O(1 + \frac{x_i/\delta \cdot \log \Delta}{\log n}) = O(\log^* n + \delta^{-1} \log \Delta)$.

ALGORITHM 41. MULTICOLORTRIAL for Lemma 7.17

Input: a graph G, a partial proper coloring φ, $\delta \in (0, 1)$ and $x \leqslant \delta\Delta/2$.
Output: a proper extension of φ.

Repeat 6 times: each uncolored vertex v does,

(1) Sample a set $\mathbf{S}(v)$ of $160x/\delta$ colors in $[\Delta + 1]$ with replacement.
(2) If $\mathbf{S}(v) \cap L_\varphi(v) < x$, then set $\mathbf{S}'(v) = \emptyset$. Otherwise, let $\mathbf{S}'(v)$ be the first x colors sampled in $\mathbf{S}(v)$ that also belong to $L_\varphi(v)$.

(3) If $\chi \in \mathbf{S}'(v) \setminus \mathbf{S}'(N(v))$, then extend φ with χ at v. Otherwise, vertex v remains uncolored.

Lemma A.6. *Let G, φ, δ and x be as described in Algorithm 41 and such that $|L_\varphi(v)| \geqslant \deg(v) + \delta\Delta$. Let $S \subseteq V_G \setminus \operatorname{dom}\varphi$ be a set of uncolored vertices for which $2x \deg_\varphi(v) \leqslant |L_\varphi(v)|$. Then Algorithm 41 fails to colors all vertices of S with probability at most $2^{-3|S|x}$.*

Proof. Consider some $v \in S$. Denote by $\boldsymbol{\chi}_i$ the i-th color sampled in $\mathbf{S}(v)$ in Step 1. Define $\mathbf{X}_i$ to be one if the set $\{\boldsymbol{\chi}_1, \dots, \boldsymbol{\chi}_{i-1}\}$ already contains x colors in $L_\varphi(v)$. Otherwise, $\mathbf{X}_i$ equals one if $\boldsymbol{\chi}_i \in L_\varphi(v) \setminus \{\boldsymbol{\chi}_1, \dots, \boldsymbol{\chi}_{i-1}\}$ and zero otherwise. In the former case, $\mathbb{E}[\mathbf{X}_i \,|\, \boldsymbol{\chi}_1, \dots, \boldsymbol{\chi}_{i-1}] = 1$, and in the latter

$$\mathbb{E}[\mathbf{X}_i \,|\, \boldsymbol{\chi}_1, \dots, \boldsymbol{\chi}_{i-1}] \geqslant \frac{\delta\Delta - (x-1)}{\Delta + 1} \geqslant \delta/2 \ ,$$

where the last inequality uses that $x \leqslant \delta\Delta/2$. Let $\mathbf{X} = \sum_i \mathbf{X}_i$ and note that $\mathbb{E}[\mathbf{X}] \geqslant \mu = 80x$. By Chernoff with domination (Lemma 2.3), with have that

$$\mathbb{P}[\mathbf{X} < x] \leqslant \mathbb{P}[\mathbf{X} < \mu/2] \leqslant \exp(-\mu/8) = \exp(-10x) \ .$$

Observe that, by definition, $\mathbf{X} \geqslant x$ iff $|\mathbf{S}(v) \cap L_\varphi(v)| \geqslant x$. So $\mathbf{S}'(v) = \emptyset$ with probability at most $e^{-10x} < 2^{-10x}$.

Fix the sets of colors tried by neighbors arbitrarily. That is $\mathbf{S}'(N(v)) = T \subseteq [\Delta + 1]$ where $|T| \leqslant x \deg_\varphi(v) < |L_\varphi(v)|/2$. When $\mathbf{S}'(v) \neq \emptyset$, the color trial of v in Step (3) can be coupled with the following sequential process: sample $\boldsymbol{\chi} \in [\Delta + 1]$, if $\boldsymbol{\chi} \in L_\varphi(v)$ then adopt it only if $\boldsymbol{\chi} \notin T$; when v fails $\boldsymbol{\chi} \in T$ for the x-th time, it stops and remain uncolored. Each $\boldsymbol{\chi}$ for which we test if $\boldsymbol{\chi} \in T$ is uniformly distributed in $L_\varphi(v)$ and independent of earlier trials. So the probability that v fails all tests $\boldsymbol{\chi} \in T$ is at most 2^{-x}.

Overall, the probability that v fails to gets colored in one iteration of Steps 1 to 3 is at most $2^{-10x} + 2^{-x} \leqslant 2^{-x/2}$. Since our argument only relies on the randomness of colors sampled by v and that each iterations of Steps 1 to 3 are independent, the probability that all nodes of S fail to get colored is at most $(2^{-x/2 \cdot |S|})^6 = 2^{-3|S|x}$ as intended. ∎

A.5. Compressing the Size of Colors in Virtual Graphs

Lemma A.7. *Let k be some integer such that $\Delta \gg k^2 \log^2 n$. Let φ be any arbitrary coloring and K be an almost-clique. Call $D \subseteq L_\varphi(K)$ the set containing the k smallest colors in $L_\varphi(K)$. There exist a $O(\mathsf{cd})$-round algorithm running in K at the end of which, w.h.p., all $v \in K$ know the same hash function $h : [\Delta+1] \to [4k^2]$ that can be described in $O(\log\log n + \log k)$ bits and has* no collision on D, *i.e., $|h(D)| = |D|$.*

Proof. Let $N = [\Delta + 1]$, $M = 4k^2$ and $\varepsilon' = 1$. By Theorem 2.11, there exists some ε'-almost pairwise independent family $\mathscr{H}$ such that for any pair $\chi_1 \neq \chi_2 \in D$, a random $h \in \mathscr{H}$ has a collision $h(\chi_1) = h(\chi_2)$ with probability at most $\frac{2}{M}$. By union bound over all such pairs, the probability that a random $h \in \mathscr{H}$ has a collision on D is

$$\mathbb{P}_{h\in\mathscr{H}}[\exists \chi_1 \neq \chi_2 \in D, h(\chi_1) = h(\chi_2)] \leqslant \frac{2k^2}{M} < 1/2 \ .$$

Each $h \in \mathscr{H}$ requires $O(\log\log N + \log M) = O(\log\log n + \log k)$ bits to represent.

Partition the almost-clique into $k^2 \cdot \Theta(\log n)$ random groups. Each group correspond to a triplet $(i, j, t) \in [k] \times [k] \times [\Theta(\log n)]$. For each $t \in [\Theta(\log n)]$, one vertex samples a uniform $h_t \in \mathscr{H}$ and broadcast it to all groups (i, j, t). Vertices of the group for (i, j, t) query for the i-th and j-th color in $L_\varphi(K)$ the clique palette (Lemma 11.3). They check if h_t collides for the corresponding pair, i.e., if $h_t(\chi_i) = h_t(\chi_j)$ where χ_i (resp. χ_j) is the i-th color (resp. j-th color).

By aggregating a bit-wise OR in all groups $\{(i, j, t), i, j \in [k] \times [k]\}$, those groups learn if h_t is collision-free on D. Since each h_t is collision-free with probability $1/2$, w.h.p., at least one must be. We find an arbitrary one by simple converge cast in K. It can then be broadcasted to all vertices of K. ∎