A Distributed Palette Sparsification Theorem

Maxime Flin Reykjavik University maximef@ru.is Mohsen Ghaffari MIT ghaffari@mit.edu Magnús M. Halldórsson Reykjavik University mmh@ru.is

Fabian Kuhn University of Freiburg kuhn@cs.uni-freiburg.de Alexandre Nolin CISPA alexandre.nolin@cispa.de

Abstract

The celebrated palette sparsification result of [Assadi, Chen, and Khanna SODA'19] shows that to compute a $\Delta + 1$ coloring of the graph, where Δ denotes the maximum degree, it suffices if each node limits its color choice to $O(\log n)$ independently sampled colors in $\{1, 2, \ldots, \Delta + 1\}$. They showed that it is possible to color the resulting sparsified graph—the spanning subgraph with edges between neighbors that sampled a common color, which are only $\tilde{O}(n)$ edges—and obtain a $\Delta + 1$ coloring for the original graph. However, to compute the actual coloring, that information must be gathered at a single location for centralized processing. We seek instead a local algorithm to compute such a coloring in the sparsified graph. The question is if this can be achieved in poly(log n) distributed rounds with small messages.

Our main result is an algorithm that computes a $\Delta + 1$ -coloring after palette sparsification with $O(\log^2 n)$ random colors per node and runs in $O(\log^2 \Delta + \log^3 \log n)$ rounds on the sparsified graph, using $O(\log n)$ -bit messages. We show that this is close to the best possible: any distributed $\Delta + 1$ -coloring algorithm that runs in the LOCAL model on the sparsified graph, given by palette sparsification, for any poly $(\log n)$ colors per node, requires $\Omega(\log \Delta / \log \log n)$ rounds. This distributed palette sparsification result leads to the first poly $(\log n)$ -round algorithms for $\Delta + 1$ -coloring in two previously studied distributed models: the Node Capacitated Clique, and the cluster graph model.

Contents

1	Introduction	1
	1.1 Background and State of the Art	1
	1.2 Our Results	2
	1.3 Corollaries for Other Models	3
	1.4 Related Work and Problems	4
2	Technical Introduction	5
	2.1 Comparison with Palette Sparsification	5
	2.2 Our Approach and New Ideas	6
	2.2.1 Step 1: Preconditioning of Almost-Cliques	6
	2.2.2 Step 2: Distributed Colorful Matching	7
	2.2.3 Step 3: Augmenting Trees	7
	2.3 Lower Bound	9
3	Preliminaries	10
	3.1 Distributed Coloring	10
	3.2 Sparse-Dense Decomposition	11
4	Palette Sampling and The Sparsified Graph	11
	4.1 Palette Sampling	11
	4.2 Decomposition and Properties	12
5	The Distributed Palette Sparsification Theorem	15
5 6	The Distributed Palette Sparsification Theorem	15 18
5 6	The Distributed Palette Sparsification Theorem Augmenting Paths 6.1 Detailed Description of the Algorithm	15 18 19
5 6	The Distributed Palette Sparsification Theorem Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees	15 18 19 21
5 6	Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees 6.3 Harvesting the Trees	15 18 19 21 23
5 6 7	Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees 6.3 Harvesting the Trees Colorful Matching	15 18 19 21 23 26
5 6 7 8	Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees 6.3 Harvesting the Trees Colorful Matching Lower Bound	15 18 19 21 23 26 30
5 6 7 8	Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees 6.3 Harvesting the Trees Colorful Matching Lower Bound	15 18 19 21 23 26 30
5 6 7 8 A	Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees 6.3 Harvesting the Trees Colorful Matching Lower Bound Concentration Bounds	15 18 19 21 23 26 30 41
5 6 7 8 A B	Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees 6.3 Harvesting the Trees Colorful Matching Lower Bound Concentration Bounds Omitted Proofs	15 18 19 21 23 26 30 41 42
5 6 7 8 A B	The Distributed Palette Sparsification Theorem Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees 6.3 Harvesting the Trees Colorful Matching Lower Bound Concentration Bounds Omitted Proofs B.1 Computing the Almost-Clique Decomposition	15 18 19 21 23 26 30 41 42 42
5 6 7 8 A B	The Distributed Palette Sparsification Theorem Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees 6.3 Harvesting the Trees 6.3 Harvesting the Trees Colorful Matching Lower Bound Concentration Bounds Omitted Proofs B.1 Computing the Almost-Clique Decomposition B.2 Preconditioning Almost-Clique	15 18 19 21 23 26 30 41 42 42 44
5 6 7 8 A B	The Distributed Palette Sparsification Theorem Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees 6.3 Harvesting the Trees 6.3 Harvesting the Trees Colorful Matching Lower Bound Concentration Bounds Omitted Proofs B.1 Computing the Almost-Clique Decomposition B.2 Preconditioning Almost-Clique B.3 Analysis of RandomPush	15 18 19 21 23 26 30 41 42 42 44 46
5 6 7 8 A B	The Distributed Palette Sparsification Theorem Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees 6.3 Harvesting the Trees 6.3 Harvesting the Trees Colorful Matching Lower Bound Concentration Bounds Omitted Proofs B.1 Computing the Almost-Clique Decomposition B.2 Preconditioning Almost-Clique B.3 Analysis of RandomPush Corollaries for Other Models	15 18 19 21 23 26 30 41 42 42 44 46 47
5 6 7 8 A B C	The Distributed Palette Sparsification Theorem Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees 6.3 Harvesting the Trees 6.3 Harvesting the Trees Colorful Matching Lower Bound Concentration Bounds Omitted Proofs B.1 Computing the Almost-Clique Decomposition B.2 Preconditioning Almost-Clique B.3 Analysis of RandomPush Corollaries for Other Models C.1 Coloring in Distributed Streaming	15 18 19 21 23 26 30 41 42 42 44 46 47 47
5 6 7 8 A B C	The Distributed Palette Sparsification Theorem Augmenting Paths 6.1 Detailed Description of the Algorithm 6.2 Growing the Trees 6.3 Harvesting the Trees 6.3 Harvesting the Trees Colorful Matching Lower Bound Concentration Bounds Omitted Proofs B.1 Computing the Almost-Clique Decomposition B.2 Preconditioning Almost-Clique B.3 Analysis of RandomPush Corollaries for Other Models C.1 Coloring in Distributed Streaming C.2 Coloring in the Cluster Graph Model	15 18 19 21 23 26 30 41 42 42 44 46 47 47 47 47

1 Introduction

The Palette Sparsification Theorem of Assadi, Chen, and Khanna (ACK, henceforth) [ACK19] is a beautiful and powerful sparsification result for the Δ + 1-coloring problem: the problem of assigning a color $c(v) \in \{1, \ldots, \Delta + 1\}$ to each node $v \in V$ of an *n*-node graph G = (V, E)such that adjacent nodes $u, v \in V$, for which $uv \in E$, receive different colors. Here, Δ is the maximum degree of the graph. ACK show that we can Δ + 1-color any graph G, by list-coloring a sparse sub-graph \tilde{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 L(v) of $O(\log n)$ colors for each vertex $v \in V$, with high probability, the graph G is L-listcolorable. That is, there exists a coloring of G where each v is assigned a color $c(v) \in L(v)$. To compute a $\Delta + 1$ -coloring of G, one then computes an L-list-coloring of the sub-graph \tilde{G} retaining only edges $uv \in E$ where $L(u) \cap L(v) \neq \emptyset$. A simple argument shows that \tilde{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, where each graph node needs to interact and coordinate with only poly(log n) of its neighbors. However, all known applications of the palette sparsification theorem require gathering the sparsified subgraph \tilde{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 poly log n bits per round.

In this paper, 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 polylogarithmic randomized algorithms in constrained settings studied in the distributed literature [RGH⁺22, GKK⁺15, GH16, GK13, GZ22, AGG⁺19].

1.1 Background and State of the Art

Distributed Coloring. The Δ + 1-coloring problem has been one of the central problems in the study of distributed graph algorithms [PS97, Joh99, SW10, FHK16, BEPS16, HSS18, CLP20, GGR21, HKMT21, GK21, HKNT22]. In fact, this was the main problem studied by Linial in his celebrated paper introducing the LOCAL model [Lin92]. In this model, we have a communication network between processors, abstracted as an undirected graph, and this is also the graph for which we want to compute a vertex coloring. Each vertex is equipped with a $O(\log n)$ -bit unique identifier (where n = |V|) and communicates in synchronous rounds with its neighbors. The variant of this model with $O(\log n)$ -bit messages is known as the CONGEST model [Pel00].

In recent years, there has been exciting progress on sublogarithmic time randomized algorithms [BEPS16, HSS18, CLP20, GK21, HKNT22, HNT22, GG23] culminating in state-of-the-art complexities of $O(\log^3 \log n)$ in CONGEST and $\tilde{O}(\log^2 \log n)$ in LOCAL. In fact, when $\Delta \ge \Omega(\log^4 n)$ — which is the interesting range for [ACK19] — the best round complexity known is $O(\log^* n)$ [HKNT22, HNT22].

In constrained distributed models such as *cluster graphs* and *the node congested clique* (see Section 1.3), where nodes can effectively send/receive only poly $\log n$ bits per rounds (or more generally, poly $\log n$ bit aggregate summaries of the messages), no 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 \tilde{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 poly(log n) bits. However, that hinges on whether \tilde{G} can be colored fast *distributively*. Unfortunately, the proof of [ACK19] is intrinsically centralized (for reasons explained in Section 2.1). All applications of [ACK19] use centralization to compute the $\Delta + 1$ -coloring. The research question at the core of our paper 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?

1.2 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 $\Delta \ge \Omega(\log^4 n)$), 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 $\Delta \ge \Omega(\log^4 n)$. When $\Delta \le O(\log^4 n)$, 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 1. [Distributed Palette Sparsification Theorem] Suppose that each node in a graph G samples $\Theta(\log^2 n)$ colors u.a.r. from $[\Delta+1]$. There is a distributed messagepassing algorithm operating on the sparsified graph, that computes a valid list-coloring in $O(\log^2 \Delta + \log^3 \log n)$ rounds, using $O(\log n)$ -bit messages. In particular, each node needs to communicate with only $O(\log^4 n)$ different neighbors.

Our Techniques in a Nutshell. We shall give an overview of our algorithm in Section 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 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 poly log n number of colors in the sparsification.

1.3 Corollaries for Other Models

Distributed Streaming. The semi-streaming model – where we skim through the (very large) set of edges of the graph and store only poly log n bit of memory per node before solving the problem – has been studied extensively [AMS96, BJK⁺02, CCF02, CM04, AGM12, ACK19]. A frequent technique in this setting is *(distributed) sketching* [AGM12, KLM⁺14, GMT15, ACK19]: nodes locally compress their neighborhoods to 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 locak at their edges in one streaming pass, using only 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 $\tilde{O}(n)$ edges: each neighborhood must contain at most poly log n edges. See Appendix C.1 for a more precise definition.

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., [RGH⁺22, GKK⁺15, 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 and convergecast in the cluster. One round of communication on the cluster graph involves: (1) broadcasting a 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 poly(log n)-bit aggregate function from the cluster nodes to the center.

Prior to our work, it remained open whether one can compute a $\Delta + 1$ -coloring in polylog 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 polylog 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 Appendix C.2 for definitions and the actual result.

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 $[AGG^+19]$ 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 Appendix C.3, this can be replaced by an existential construction¹.

1.4 Related Work and Problems

The groundbreaking palette sparsification theorem of Assadi, Chen, and Khanna [ACK19] showed that Δ + 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 [AKM22].

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.

Coloring plays a central role in distributed algorithms as a natural approach to breaking symmetry and scheduling access to exclusive resources. In particular, the original work of Linial [Lin92] introducing local algorithms and the LOCAL model was specifically about the Δ + 1-coloring problem. Since then, there has been a lot of work on local coloring algorithms, both randomized [Joh99, BEPS16, SW10, HSS18, CLP20, HKMT21, HKNT22] and deterministic (e.g., [Bar15, BEG18, MT20, GK21, GG23]).

The Node-Capacitated Clique model was introduced in [AGG⁺19] to model distributed systems built on top of virtual overlay networks. They gave algorithms for the maximal independent set problem and O(a)-coloring, with time linear in a, where a is the arboricity of the graph. The question of efficient $\Delta + 1$ -coloring has remained open.

Many models of distributed computing, both in theory and in nature, are much more restrictive than LOCAL or CONGEST, both in terms of communication abilities and processing power: e.g., beeping model [CK10], wireless (ad-hoc, unstructured...), programmable matter [DDG⁺14], networked finite-state machines [EW13]. These often capture distributed features of the natural and physical world. A logical direction is therefore to identify models that strongly limit the power of the nodes, yet allow for fast distributed computation. Few features are as fundamental as limiting the amount of space available.

A recent work by [FGH⁺23] uses some (of the earlier) subroutines from our work to Δ + 1-color graphs in the broadcast congest model of distributed computing. In that model, per round, each node must send the same $O(\log n)$ -bit message to all of its neighbors. They adapt Algorithms 9 and 10 to compute an almost-clique decomposition and a colorful matching in O(1) rounds. How-

¹We believe that our algorithm can be implemented using poly $\log n$ -wise independent random bits to sample each of the poly $\log n$ colors in the lists. It would be at the cost of a higher poly $\log n$ round complexity and possibly larger poly $\log n$ number of colors in lists. As this is not a major contribution and requires a significant amount of extra technicalities, we reserve this for future work.

ever, we emphasize that the core technical challenges and contributions in the two works are different.

2 Technical Introduction

In this section, we outline the techniques we use to prove Theorems 1 and 2. Our algorithm builds on existing literature, both streaming [ACK19, AA20, AKM22] 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 2.1, we review the palette sparsification theorem of [ACK19] and explain why it does not extend to our setting. Then, in Section 2.2, we outline the technical novelties in our algorithm. Finally, in Section 2.3, we describe an overview of our lower bound result.

2.1 Comparison with Palette Sparsification

The proof of [ACK19] relies on a variant of the sparse-dense decomposition introduced by [Ree98]. Variants of this decomposition were used in earlier distributed coloring algorithms [HSS16, CLP18]. This decomposition partitions the graph into a set of "locally sparse" vertices and a collection of "almost-cliques". For intuition, consider a (somewhat degenerate) example of a sparse node: a vertex of degree $\Delta/2$. If we try to color this vertex with a color from $[\Delta + 1]$ uniformly at random, it has probability 1/2 to succeed (no matter what colors its neighbors choose). Repeating this process iteratively $O(\log n)$ times is enough to color all such vertices with high probability. It was observed by [EPS15] that this reasoning extends to sparse vertices (with a more involved analysis). It is worth noting that such randomized color trials are easily implemented in LOCAL (and CONGEST) and form a core component of fast randomized distributed coloring algorithms [Joh99, BEPS16, CLP20]. Indeed, the part of the algorithm of [ACK19] for coloring sparse vertices is also already distributed. Theorem 1 improves the $O(\log n)$ -round algorithm of [ACK19] to $O(\log \Delta)$ by using the faster algorithm of [SW10] to color sparse nodes, but we do not have 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 1, for reasons expounded in Section 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 C, [ACK19] looks for a perfect matching in the bipartite graph with vertices of C on the one side, colors in $[\Delta + 1]$ on the other and an edge between $v \in C$ and $c \in [\Delta + 1]$ if c is in L(v) and not used by an outside neighbor. If $|C| \leq \Delta + 1$, classic results from random graph theory (e.g., [Bol98, 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 1 provides a *distributed algorithm* to compute it. Note that learning the full topology of C 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 6), while also managing outside conflicts. In large almost-cliques $|C| \ge \Delta + 1$, such a perfect matching cannot exist. To deal with those, [ACK19] shows the existence of a *colorful matching*. Namely, they color pairs of anti-neighbors in C (pair of nodes that are not connected by an edge) using the same color so that the number of uncolored nodes decreases twice as fast as the number of free colors. We give a fast distributed version of the sequential algorithm of [ACK19] (see Section 2.2.2). While this is not the main technical contribution of our work, we think this procedure itself might find applications in future distributed coloring algorithms.

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.

2.2 Our Approach and New Ideas

2.2.1 Step 1: Preconditioning of Almost-Cliques

Like [ACK19], our algorithm heavily relies on an ε -almost-clique decomposition. More formally, it decomposes the graph into a set V_{sparse} of locally sparse nodes and a collection C_1, \ldots, C_t of almost-cliques. An ε -almost-clique is a cluster $|C| \leq (1 + \varepsilon)\Delta$ such that each node $v \in C$ has $|N(v) \cap C| \geq (1 - \varepsilon)\Delta$ neighbors in C (see Definition 4). 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 3 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 Lemmas 6.1 and 7.1). We now give further details on that aspect. Readers that are not familiar with palette sparsification results may skip the remainder of this subsubsection on the first reading.

The key property of cliques that our algorithm uses is that when k nodes are uncolored, there are k "available colors" that are used by no one in the clique. The colorful matching (Section 2.2.2) 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 C, i.e., k = 1. In our algorithm, a constant fraction of C samples $K \log n$ colors for some large enough K > 0. This way, the probability that at least one node in this almost-clique finds that one available color is at least $1 - (1 - 1/\Delta)^{K\Delta \log n} \ge 1 - 1/\operatorname{poly}(n)$. Having nodes sample more colors has a drawback: it increases the competition for colors. If a node $v \in C$ 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 $1 - (1 - 1/\Delta)^{\varepsilon K\Delta \log n} \ge 1 - 1/\operatorname{poly}(n)$ (as $\varepsilon K \in \Theta(1)$). During preconditioning, we ensure that nodes in active almost-cliques have at most $\Delta/(K \log n)$ external neighbors in other active almost-cliques. The probability that an external neighbor blocks the one color that v is looking for becomes $1 - (1 - 1/\Delta)^{\frac{\Delta \log n}{K \log n}} \approx 1 - 1/e$. Therefore, only a small fraction of C is affected by the randomness outside of C. This argument is made formal in Lemma 6.1.

To precondition almost-cliques, we use an idea from distributed coloring algorithms [SW10, EPS15, HKMT21]. Namely, nodes that have $\Omega(\Delta/\log n)$ connections to nodes in other almostcliques are $\Omega(\Delta/\log n)$ sparse. By coloring nodes in a carefully chosen order, we get Theorem 3. As it only uses well-known techniques from distributed coloring, we defer the analysis to Appendix B.2. 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.

2.2.2 Step 2: Distributed Colorful Matching

A colorful matching is defined as a matching in the complement graph of the almost-clique such that endpoints of a matched edge are colored the same (Definition 6). Intuitively, this reduces the size of the clique: if one merges matched nodes, one reduces the number of nodes in the almost-clique while maintaining a proper coloring of the original graph. In an almost-clique of $(1 + \varepsilon)\Delta$ nodes, finding $\varepsilon\Delta$ such pairs essentially reduces the coloring problem to that of coloring a clique. This technique was introduced by [ACK19] in the first palette sparsification theorem to deal with that exact issue and we claim no novelty in its use. In [ACK19], however, only the existence of a large enough matching is proven, whereas we also need to compute it efficiently in a distributed setting.

We define \bar{d}_C as the average anti-degree in C: such that $\bar{d}_C|C|/2$ is the number of anti-edges. When nodes try a random color in $[\Delta+1]$, an anti-edge is monochromatic with probability $1/(\Delta+1)$. Therefore, the expected number of monochromatic edges is $\delta \bar{d}_C$ for some small constant $\delta > 0$, because a node can retain its color with constant probability. Using tools similar to [EPS15], one can show this random variable is concentrated near its mean with probability $1 - e^{-\Omega(\bar{d}_C)}$. It implies that in cliques with $\bar{d}_C \ge \Omega(\log n)$, for any constant $K = O(1/\varepsilon)$, we can accumulate $K\bar{d}_C$ anti-edges in the colorful matching in $O(K/\delta)$ rounds (Lemma 7.3).

We use a different approach when $\bar{d}_C \leq O(\log n)$. Note that when \bar{d}_C is smaller than some constant, nodes have hardly any anti-edges. Hence we can also assume $\bar{d}_C \geq \Omega(1)$; 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 clique (Lemma 7.3). Since the probability of a non-edge having both endpoints sample a common color is $\Theta(\log^2 n/\Delta)$, only $O(\bar{d}_C \log^2 n) = O(\log^3 n)$ potential monochromatic non-edges are sampled. By taking advantage of the high expansion property of the sparsified clique, we can disseminate the list of $O(\log^3 n)$ sampled monochromatic edges in $O(\log \Delta)$ rounds to all nodes in the clique, which can then compute the colorful matching locally. Because $\Omega(\Delta)$ colors are available in the clique, the concurrent coloring of external neighbors can block at most a small fraction of the colors (Lemma 7.1).

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 almostcliques. 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 C at the current iteration. We say that color c is available to a node v if c is neither used in C nor by external neighbors of v that were colored during the preconditioning step (these nodes will never change colors). In each iteration, we grow a forest of augmenting paths (Definition 8). An augmenting path is a path $u_0, u_1, \ldots u_i$ 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 c. Provided with such a path, we can recolor u_i with c 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, \ldots, u_i verifying 1) and 2) but not 3), then u_i samples a uniform color $c \in [\Delta + 1]$ 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 C will have k colors available (Lemma 5.1). On the other hand, the argument sketched in Section 2.2.1 shows that because of the preconditioning step, with high probability over the randomness outside of C, at least $\Omega(\Delta)$ nodes in C 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 10). Since they represent a small fraction of C and the path explores the almost-clique randomly, we are unlikely to fail due to spoiled nodes (Lemma 6.4).

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 Fig. 1 for a high-level description of one iteration.

<u>Augmenting Path Algorithm</u>

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 $c \in S_u$ find $v_{c,u} \in C$ colored c and connect them to v in the forest.

Harvesting the trees.

- 1. If $k \ge \Omega(\log n)$, 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 $k \leq O(\log n)$, 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 C. 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 4), where we grow the trees, and the harvesting phase (Algorithms 5 and 6), 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 $k \leq O(\log n)$, we cannot show progress with high probability with a simple concentration on k. See Section 6.1 for a more detailed description.

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 $V = \{1, \ldots, n\}$ that we want to color with $n = \Delta + 1$ colors. Every node $v \in V$ samples a random subset S_v of colors $C = \{1, \ldots, n\}$ as follows. For each node $v \in V$ and each color $x \in C$, x is included in S_v independently with probability p = f(n)/n, where $f(n) \ge c \ln n$ for a sufficiently large constant c and $f(n) \le polylog n$. Recall that the sparsified graph is the graph induced by all edges of K_n between nodes $u, v \in V$ with $S_u \cap S_v \ne \emptyset$. 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 vis 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 $C = \{1, \ldots, n\}$ represents the set of colors. There is an edge between nodes $v \in V$ and $x \in C$ 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 $p \ge c \ln n/n$ 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., [Bol98, 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 $x \in C$ 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 2*n*-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 $d \ge 0$, 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 Tis 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 Section 8.

3 Preliminaries

Notation. For any integer $k \ge 1$, we write [k] for the set $\{1, 2, \ldots, k\}$. For a tuple $X = (X_1, X_2, \ldots, X_k)$ and some $i \in [k]$, we define $X_{\le i} = (X_1, \ldots, X_i)$. For a graph G = (V, E) and any set $S \subset V$, we denote by $N_G(S) = \{v \in V \setminus S : \exists u \in S \text{ and } uv \in E\}$ the neighborhood of S. Moreover, for any sets $S_1, S_2 \subseteq V$, let $E_G(S_1, S_2) = E \cap (S_1 \times S_2)$ be the set of edges between nodes of S_1 and S_2 .

A partial $(\Delta + 1)$ -coloring is a function from the nodes V to $[\Delta + 1] \cup \{\bot\}$ that assigns colors in $[\Delta + 1]$ or no colors (in the form of the null color \bot) to vertices, such that adjacent vertices have different colors in $[\Delta + 1]$ (but they may both have color \bot). For some partial $(\Delta + 1)$ -coloring of the graph and any set $S \subseteq V$, we denote by \hat{S} the uncolored nodes of S and by \check{S} the colored ones.

When we say that an event happens "with high probability", we mean it occurs with probability $1 - 1/\operatorname{poly}(n)$ for a suitably large polynomial in n to union bound over polynomially many such events.

3.1 Distributed Coloring

A standard technique in distributed coloring used by randomized algorithms is to have each node repeatedly try a color picked uniformly at random in its palette: the set of colors not already used by its neighbors. It was introduced by [Joh99] and is used in all efficient distributed algorithms [BEPS16, CLP20, HKNT22]. We introduce this notion here for further use.

Definition 1 (Palette). The *palette* Ψ_v of a node v with respect to some coloring of the nodes is the set of colors that are not used by its neighbors.

Definition 2 (Slack). The slack of v is the difference between the size of its palette and its uncolored degree.

If nodes have slack proportional to their degree, they can be colored in $O(\log^* n)$ rounds of LOCAL by trying random colors. The following result has origins in [SW10] and was generalized by [CLP20]. It is straightforward to see that the proof of [HKNT22] extends to our setting.

Lemma 3.1 (Lemma 1 in [HKNT22]). Consider the (deg +1)-list coloring problem where each node v has slack $s(v) = \Omega(d(v))$. Let $1 < s_{\min} \leq \min_v s(v)$ be globally known. For every $\kappa \in (1/s_{\min}, 1]$, there is a randomized LOCAL algorithm SLACKCOLOR(s_{\min}) that in $O(\log^* s_{\min} + 1/\kappa)$ rounds properly colors each node v w.p. $1 - \exp(-\Omega(s_{\min}^{1/(1+\kappa)})) - \Delta e^{-\Omega(s_{\min})}$, even conditioned on arbitrary random choices of nodes at distance ≥ 2 from v. Using $O(\log n)$ -bit messages, it requires $O(\log \Delta)$ rounds of communication, and requires nodes to sample up to $\Theta(\frac{s_v \log n}{\Delta})$ colors in $[\Delta + 1]$.

3.2 Sparse-Dense Decomposition

We use a decomposition of the graph into *locally sparse* nodes, which have many non-edges from in their neighborhood, and dense clusters called *almost-cliques* (also informally called *cliques*). Almost-clique decomposition was first introduced in graph theory by [Ree98] and has been used extensively in streaming [ACK19, AW22] and distributed [HSS18, CLP20, HKNT22] algorithms.

Definition 3 (Sparsity). The *sparsity* of a node v is the value $\zeta_v = \frac{1}{\Delta} \left(\binom{\Delta}{2} - |E(N(v))| \right)$. We say a node is ζ -sparse if $\zeta_v \ge \zeta$, otherwise it is ζ -dense.

Definition 4 (Almost-Clique Decomposition). For $\varepsilon \in (0, 1/3)$, a ε -almost-clique decomposition is a partitioning of the vertices into sets $V_{\text{sparse}}, C_1, \ldots, C_k$ for some k such that:

- 1. All $v \in V_{\text{sparse}}$ are $\Omega(\varepsilon^2 \Delta)$ -sparse.
- 2. For any $i \in [k]$, almost-clique C_i has the following properties:
 - (a) $|C_i| \leq (1+\varepsilon)\Delta;$
 - (b) $|N(v) \cap C| \ge (1 \varepsilon)\Delta$ for all nodes $v \in C_i$.

For a dense node v in some almost-clique C, we call its *external degree* e_v the number of neighbors v has outside of its almost-clique, i.e. $e_v = |N(v) \setminus C|$, and its *anti-degree* a_v the number of non-neighbors in C, i.e. $a_v = |C \setminus N(v)|$. We denote by $\overline{d}_C = \sum_{v \in C} a_v / |C|$ the average anti-degree of C.

4 Palette Sampling and The Sparsified Graph

Parameters. We assume that $\Delta \ge \Omega(\log^4 n)$ as otherwise nodes can store all their adjacent edges and simply run the CONGEST algorithm of [HKMT21]. We define the following parameters² for our algorithm:

 $\begin{array}{ll} \alpha & \stackrel{\text{def}}{=} 500: & \text{quantify the number of leaves that an augmenting tree must have,} \\ \beta & \stackrel{\text{def}}{=} [C \log n]: & \text{used to bound the number of sampled colors,} & (1) \\ \varepsilon & \stackrel{\text{def}}{=} 10^{-8}: & \text{a small enough constant.} \end{array}$

The constant C in β is sufficiently large for high probability events 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:

$$\varepsilon \leq 1/\alpha^2$$
 and $2\alpha < 1/(18\varepsilon)$. (2)

4.1 Palette Sampling

Similar to [ACK19], we see the lists of random colors 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 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 $O(\log n)$.

²which we have not attempted to optimize.

Algorithm 1. Palette Sampling for a node v.

- $L_1(v)$: sample $O(\log^2 n)$ colors independently and uniformly at random in $[\Delta + 1]$.
- $L_2(v) = \{L_{2,i}(v), \text{ for } i \in [O(1/\varepsilon)]\} \cup \{L_2^*(v)\}$: for each $i \in [O(1/\varepsilon)]$, sample each color in $L_{2,i}(v)$ independently with probability $\frac{1}{4\Delta}$. Sample each color independently in $L_2^*(v)$ with probability $\gamma \cdot \frac{\beta}{\Delta}$ for some constant $\gamma = \Theta(1/\varepsilon^2)$ defined in Lemma 7.4.
- $L_3(v) = \{L_{3,i}(v) = L_{3,i}^{\mathsf{G}}(v) \cup L_{3,i}^{\mathsf{H}}(v), \text{ for } i \in [\beta]\}$ where we sample each color $c \in [\Delta + 1]$ in $L_{3,i}^{\mathsf{G}}$ and $L_{3,i}^{\mathsf{H}}$ independently with probability $\frac{20\beta^2}{\Delta}$.

By union bound, a fixed color $c \in [\Delta + 1]$ is included in $L_2(v)$ with probability $O(\frac{\beta}{\varepsilon^2 \Delta})$ and in $L_3(v)$ with probability $O(\frac{\beta^2}{\Delta})$. Since each color is included in L_2 independently, a simple Chernoff bound proves the following claim:

Claim 4.1. With high probability, $|L_2(v)| \leq O(\log n/\varepsilon^2)$ and $|L_3(v)| \leq O(\log^2 n)$ for all $v \in V$. Furthermore, each $L_{3,i}^{\mathsf{G}}(v)$ and $L_{3,i}^{\mathsf{H}}(v)$ contains at least 6β different colors.

Our algorithm will color each $v \in V$ with a color from its list 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 5 (The Sparsified Graph). For a graph G = (V, E) and lists L(v) such as described in Algorithm 1, let $\tilde{G} = (V, \tilde{E})$ be the subgraph of G with edges $uv \in E$ such that $L(u) \cap L(v) \neq \emptyset$. We call \tilde{G} the sparsified graph. For any set $S \subseteq V$, we denote by \tilde{S} the induced subgraph $\tilde{G}[S]$.

Claim 4.2 ([ACK19, Lemma 4.1]). For any graph G, w.h.p., the sparsified graph \widetilde{G} has maximum degree $O(\log^4 n)$.

4.2 Decomposition and Properties

Similarly to other distributed algorithms [HSS18, CLP20, HKNT22] our algorithm needs to know the almost-clique decomposition in order to compute the coloring. However, existing CONGEST algorithms require communication with \gg poly log *n* nodes [HKMT21, HNT22]. Building on [HNT22], we give a CONGEST algorithm where nodes need only to communicate on a sparse subgraph of *G*. Algorithm 2 gives an overview of the communication needed.

We emphasize that Algorithm 2 uses a sparse subgraph of G that is independent of the sparsified subgraph induced by the random lists (Definition 5). We found it simpler to state our algorithm this way. Observe, however, that Algorithm 2 could be implemented using edges sampled from random lists. We briefly explain why. Two nodes u and v adjacent in the sparsified graph share at least one color c. To know if they share a large fraction of their neighborhood — i.e., if they are friends (Definition 11) — notice that the number of nodes in $N(u) \cap N(v)$ that sample c is concentrated, and therefore provides an unbiased estimator for the size of this set. Using a bandwidth compression technique introduced by [HNT22], 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 12) — 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. Algorithm 2. High-level algorithm computing ε -almost-clique decomposition. Input: the sparsified graph $\tilde{G} = (V, \tilde{E})$. Output: an ε -almost-clique decomposition of G.

- 1. Each node v samples a value $r(v) \in [\Theta(\Delta/\varepsilon)]$ using public randomness.
- 2. Each node v with degree at least $\Delta/2$ computes
 - the set $F(v) = \{r(u) : u \in N(v), r(u) \leq \sigma\}$ where $\sigma = \Theta(\log n/\varepsilon^4)$.
 - a set $E_s(v)$ of $O(\log n/\varepsilon^2)$ random edges.
- 3. (Communication Phase) After $O(1/\varepsilon^4)$ rounds of communication using edges $E_s(v)$ and $O(\log \Delta)$ rounds of communication on \tilde{G} , each v knows if it is sparse or dense as well as the unique identifier of its cluster if it is dense.

Remark 4.3. Some important remarks about Algorithm 2.

- Note that F(v) depends on the randomness of the entire neighborhood of v. This is not an issue for any of our applications as it can easily be computed on a stream with O(log n/ε⁴) local memory, with public randomness in the Node Congested Clique and with aggregation of a single O(log n/ε⁴)-bitmap in cluster graphs.
- 2. Nodes sample edges that might not belong to \tilde{G} . Nonetheless, we assume they can communicate along these edges. To remove this assumption, one could encode the sampling of $E_s(v)$ within the palette sampling process. Observe that adding $E_s(v)$ does not affect the sparsity of \tilde{G}^3 . We phrase it this way for simplicity.

Lemma 4.4. There is a $O(\log \Delta)$ -round algorithm computing an ε -almost-clique decomposition. It only broadcasts $O(\log n/\varepsilon^4)$ -bit messages and samples $O(\log n/\varepsilon^2)$ edges per node.

Since Lemma 4.4 is a rather straightforward extension of [HNT22], we defer its proof to the appendix (see Appendix B.1).

Useful properties of the sparsified clique. Let C be an ε -almost-clique. The sparsified clique \widetilde{C} is a random graph on (almost) Δ nodes where edges are sampled with probability $O(\log^4 n/\Delta)$. As such, the graph \widetilde{C} has typical properties of random graphs.

Lemma 4.5 (Expansion). Let C be an almost-clique, and assume $\Delta \ge \beta^4$. With high probability, for all subsets $S \subseteq C$ of size at most $|S| \le 3\Delta/4$,

$$|E_{\widetilde{C}}(S, C \setminus S)| \ge |S|\beta^4/40 \quad and \quad |N_{\widetilde{C}}(S) \cap (C \setminus S)| \ge \Omega(|S|).$$

Proof. We show that for any fixed set $S \subseteq C$, the edge expansion property $(|E_{\widetilde{C}}(S, C \setminus S)| \ge |S|\beta^4/40)$ holds w.p. $1 - \exp(-\Omega(|S|\beta))$. Since there are at most $|C|^x = \exp(O(x \log \Delta))$ subsets $S \subseteq C$ of size x, this allows us to claim by union bound that, w.h.p., the edge expansion property holds for all subsets $S \subseteq C$. The vertex expansion property then follows easily.

Let us consider a fixed subset $S \subseteq C$, and set $\overline{S} \stackrel{\text{def}}{=} C \setminus S$ and $L \stackrel{\text{def}}{=} \max\left(1, \frac{\Delta}{|S|\beta}\right)$. We partition the colors into $(\Delta + 1)/L$ groups of size L. Let $B_i \subseteq [\Delta + 1]$ be the colors of bucket number i. We introduce several random variables.

³This is the reason we sample $E_s(v)$ only for high-degree nodes; nodes of degree less than $\Delta/2$ will be sparse anyway.

- For each color $c \in [\Delta + 1]$ and $e = uv \in E_G[S \times \overline{S}]$, let $X_{c,e}$ be the indicator random variable for whether e's S-endpoint node u sampled color c. Let $X_{c,u} = \sum_{v \in \overline{S}: uv \in E[S \times \overline{S}]} X_{c,uv}$ and $X_c = \sum_{u \in S} X_{c,u}$.
- For each color $c \in [\Delta+1]$ and $e = uv \in E_G[S \times \overline{S}]$, let $Y_{c,e}$ be the indicator random variable for whether both of e's endpoint nodes u and v sampled color c. Note that $X_{c,e} = 0 \Rightarrow Y_{c,e} = 0$. Let $Y_{c,v} = \sum_{u \in S: uv \in E[S \times \overline{S}]} Y_{c,uv}$ and $Y_c = \sum_{v \in \overline{S}} Y_{c,v} = \sum_{e \in E[S \times \overline{S}]} Y_{c,e}$.
- For each $i \in [\Delta/L]$, let $Z_i = \sum_{c \in B_i} \sum_{e \in E[S \times \overline{S}]} (X_{c,e} \cdot Y_{c,e})$ be the contribution of bucket number i to the number of edges between S and \overline{S} .

Note that the random variables Z_i are defined with multiplicity, i.e., possibly counting each edge multiple times. We will fix that soon.

A node $v \in S$ has at least $|N_G(v) \cap \overline{S}| \ge |\overline{S}| - \varepsilon \Delta \ge (1/4 - \varepsilon)\Delta \ge \Delta/5$ neighbors in \overline{S} before sparsification – and at most Δ . In total, $E_G[S \times \overline{S}]$ contains between $|S|\Delta/5$ and $|S|\Delta$ edges. We have

$$\mathbb{E}[X_{c,e}] = \beta^2/(\Delta+1), \quad \mathbb{E}[X_c] \in [\beta^2|S|/5, \beta^2|S|], \quad \text{and} \quad \mathbb{E}[Z_i] \in [L\beta^2|S|/5, L\beta^2|S|].$$

Let us now define auxiliary random variables $X'_{c,e}$, $Y'_{c,e}$, Z'_i , and related quantities in a manner that avoids the issue of overcounting. We reveal the colors in increasing order, and for each color c, let \mathcal{G}_c be the event that smaller colors already sampled $|S|\beta^4/20$ distinct edges into the sparsified graph. Note that \mathcal{G}_c is fully determined by the randomness of earlier colors. Let $X'_{c,e} = X_{c,e}$ and $Y'_{c,e} = Y_{c,e}$ when \mathcal{G}_c holds. When \mathcal{G}_c does not hold, let $X'_{c,e}$ and $Y'_{c,e}$ always equal 0 if both endpoints of e sampled an earlier color, and otherwise let $X'_{c,e} = X_{c,e}$ and $Y'_{c,e} = Y_{c,e}$ as before. $Z'_i = \sum_{c \in B_i} \sum_{e \in E[S \times \overline{S}]} (X'_{c,e} \cdot Y'_{c,e})$. Since $|S|\beta^4/10 \leq |S|\Delta/10$, there are always at least $|S|\Delta/10$ edges for which $\mathbb{E}[X'_{c,e}] > 0$ and $\mathbb{E}[Y'_{c,e}] > 0$. We have

$$\mathbb{E}[X_c'] \in [\beta^2 |S|/10, \beta^2 |S|], \text{ and } \mathbb{E}[Z_i] \in [L\beta^2 |S|/10, L\beta^2 |S|].$$

Consider a bucket B_i . We now make all random decisions regarding whether each node in S samples each color $c \in B_i$. Consider the summation $\sum_{c \in B_i} \frac{1}{\Delta} X'_c$. It has an expected value of at least $L \cdot \beta^2 |S|/(10\Delta)$, and is a sum of random variables $\frac{1}{\Delta} X'_c$ whose sampling spaces are contained in the range [0, 1]. Hence, $\sum_{c \in B_i} X'_c \ge L \cdot \beta^2 |S|/20$ w.p. at least $1 - \exp(-\Omega(L \cdot \beta^2 |S|/\Delta)) \ge 1 - \exp(-\Omega(\beta))$ by Lemma A.2 (Chernoff bound), i.e., w.h.p. Furthermore, for each $v \in \overline{S}$ and color $c \in [\Delta + 1]$, at most $O(\beta^2)$ of its neighbors in S sample c, w.h.p.

Let us now make all random decisions regarding whether nodes in \overline{S} sample each color in B_i . Let us analyze the summation

$$Z'_{i} = \sum_{c \in B_{i}} \sum_{e \in E[S \times \overline{S}]} (X'_{c,e} \cdot Y'_{c,e}) = \sum_{c \in B_{i}} \sum_{v \in \overline{S}} \left(\sum_{u \in S: uv \in E[S \times \overline{S}]} (X'_{c,uv} \cdot Y'_{c,uv}) \right).$$

 Z'_i has an expected value of at least $L \cdot \beta^4 |S|/(10\Delta)$. As random choices where fixed in S s.t. each vertex $v \in \overline{S}$ has at most $O(\beta^2)$ of its neighbors in S sample each color c, the inner term $\left(\sum_{u \in S: uv \in E[S \times \overline{S}]} (X'_{c,uv} \cdot Y'_{c,uv})\right)$ is distributed in $[0, O(\beta^2)]$, i.e., the decision of any given $v \in \overline{S}$ for each color impacts the sum by at most $O(\beta^2)$. We can thus divide the sum by $O(\beta^2)$ in order to get random variables distributed in [0, 1] and apply Lemma A.2. This gives that $Z'_i \ge L \cdot \beta^4 |S|/(20\Delta)$ w.p. at least $1 - \exp(-\Omega((L \cdot \beta^4 |S|/\Delta)/\beta^2)) \ge 1 - \exp(-\Omega(\beta))$.

Hence, each bucket contributes at least $L \cdot \beta^4 |S|/(20\Delta)$ to the overall sum $Z' = \sum i \in [\Delta/L]Z'_i$, w.h.p., regardless of the choices of previous buckets. Let us analyze the probability that $\Delta/(2L)$ or more buckets fail to have this good contribution. Consider a specific set of $\Delta/(2L)$ buckets. The probability that they all fail to have a good contribution is $\exp(-\Omega(\beta\Delta/L))$. There are less than $2^{\Delta/L}$ ways to choose $\Delta/(2L)$ buckets out of Δ/L , so by union bound the probability that less than $\Delta/(2L)$ buckets have a good contribution is at most $2^{\Delta/L} \cdot \exp(-\Omega(\beta\Delta/L)) = \exp(-\Omega(\beta\Delta/(2L)))$. Finally, $\exp(-\Omega(\beta\Delta/L)) = \exp(-\Omega(\min(\beta\Delta, |S|\beta^2))) < \exp(-\Omega(|S|\beta))$, so the sparsified graph contains at least $\beta^4 |S|/40$ edges w.p. at least $1 - \exp(-\Omega(|S|\beta))$.

This small failure probability $\exp(-\Omega(|S|\beta))$ allows us to union bound over all choices of S, using β a large enough $\Omega(\log n)$. Hence, w.h.p., the edge expansion property holds. The vertex expansion follows from Claim 4.2. Since the maximum degree in \widetilde{G} is $O(\beta^4)$, the number of nodes in \overline{S} is at least $\frac{|E_{\widetilde{C}}(S,\overline{S})|}{O(\beta^4)} = \Omega(|S|)$.

Lemma 4.5 implies the following result as any two nodes can reach more than half of the clique in $O(\log \Delta)$ hops.

Corollary 4.6. The sparsified almost-clique \widetilde{C} 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 4.7. For a clique C, let u and v be two nodes of C and $c \in [\Delta + 1]$ be an arbitrary color. Then, with high probability, there exist at least $2\beta^2/5$ nodes $w \in N_G(u) \cap N_G(v) \cap C$ that sample $c \in L(w)$. In particular, if $c \in L(u) \cap L(v)$ then u and v are at two hops from each other in the sparsified graph \tilde{G} .

Proof. Let $q \stackrel{\text{def}}{=} \frac{\beta^2}{\Delta}$. Nodes u and v share at least $(1 - 2\varepsilon)\Delta$ neighbors in C and each $w \in N_G(u) \cap N_G(v) \cap C$ sampled the color c with probability (at least) q. In expectation, at least $(1 - 2\varepsilon)\Delta \cdot q \ge (4/5)\beta^2$ 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 $1 - \exp(-\Omega(\beta^2)) \gg 1 - 1/\operatorname{poly}(n)$.

5 The Distributed Palette Sparsification Theorem

In this section, we give the CONGEST algorithm for our main theorem. Again, we assume $\Delta \ge \Omega(\log^4 n)$ and show a runtime of $O(\log^2 \Delta)$.

Theorem 1. [Distributed Palette Sparsification Theorem] Suppose that each node in a graph G 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 $O(\log^2 \Delta + \log^3 \log n)$ rounds, using $O(\log n)$ -bit messages. In particular, each node needs to communicate with only $O(\log^4 n)$ different neighbors.

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 (Definition 4).

Theorem 3. 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 $\Delta/\eta \ge K \log n$ for a large enough constant K > 0. There exists an algorithm computing a partial coloring of G where all uncolored nodes are partitioned in almost-cliques C_1, \ldots, C_t for some t such that for any $i \in [t]$, almost-clique C_i is such that:

- 1. $|C_i| \leq (1+\varepsilon)\Delta;$
- 2. $|N(v) \cap C_i| \ge (1 \varepsilon)\Delta$ for all nodes $v \in C_i$;
- 3. $|N(v) \cap \bigcup_{j \neq i} C_j| \leq e_{\max} = \Delta/\eta$

Furthermore, the algorithm runs in $O(\log \Delta + \log \eta)$ rounds, uses $O(\eta \log n)$ colors from lists L_1 , and samples $O(\eta \log n)$ edges per node.

Note that the bound on the external degree given by Property 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 3 with maximum external degree

$$e_{\max} \stackrel{\text{def}}{=} \Delta/\eta \quad \text{where} \quad \eta \stackrel{\text{def}}{=} \max(160\alpha\beta, L_{\max}/\varepsilon) ,$$
 (3)

where $L_{\text{max}} = O(\log n)$ is the upper bound on the size of lists L_2 of Claim 4.1. As this uses only standard techniques from distributed coloring, we sketch the algorithm here and defer the complete proof to Appendix B.2.

Proof Sketch. Compute an $(\varepsilon/3)$ -almost-clique decomposition in $O(\log \Delta)$ rounds (by Lemma 4.4). We use the fact that nodes with external degree Δ/η are $\Omega(\Delta/\eta)$ -sparse; hence receive permanent slack from randomized color trials (Lemma B.4).

To ensure Property 3, we divide cliques of the almost-clique decomposition into two categories: introvert cliques, with at most $(2\varepsilon/3)\Delta$ nodes of high external degree; and extrovert cliques, where more than $(2\varepsilon/3)\Delta$ nodes have high external degree. We begin by generating slack in V_{sparse} and extroverted cliques. We next color nodes of low-external-degree in extroverted cliques using the $(\varepsilon/3)\Delta$ temporary slack provided by their inactive neighbors of high external degrees. We color sparse nodes and high-external-degree nodes in introverted cliques using their permanent slack. By Lemma 3.1, this takes $O(\log \Delta)$ rounds. We finish by coloring the high-external-degree nodes in extroverted cliques in two steps: first $O(\log \log n)$ rounds of randomized color trials to reduce their degree to $O(\Delta/\log n)$, then $O(\log \Delta)$ rounds using their permanent slack. What remains uncolored are then only the low-external-degree nodes in introverted cliques.

To detect nodes of high external degree, we use random edge samples. As we sample $O(\eta \log n) = O(\log^2 n)$ edges per node, it is not an issue for our applications: nodes communicate to only $O(\log^2 n)$ nodes during this step.

Step 2: Colorful Matching

The remaining uncolored nodes are very dense (more than Δ/η -dense). We find a large matching of anti-neighbors in each clique and color (the endpoints of) each such node-pair with a different color. Such matchings are called *colorful* and were introduced by [ACK19] in the original palette sparsification theorem.

Definition 6 (Colorful Matching). For any partial coloring of the nodes, a matching M in \overline{C} (anti-edges in C) is *colorful* if and only if the endpoints of each edge in M are colored the same.

If a clique has a colorful matching, the set of colors that are not used in the clique approximates well the palette of nodes with small anti-degree. We call this set of colors the clique palette.

Definition 7 (Clique Palette). For an almost-clique C, the *clique palette* Ψ_C (w.r.t. a valid coloring of the vertices) is the set of colors not used by nodes of C.

The following lemma formalizes the idea that Ψ_C and Ψ_v are similar (for most nodes v). Recall that \hat{C} and \check{C} denote respectively the set of uncolored and colored nodes of C.

Lemma 5.1. Let C be an almost-clique with a colorful matching M and fix any partial coloring of the nodes. We say $v \in C$ is promising (with respect to M) if it satisfies $a_v \leq |M|$, and otherwise it is unpromising. For each promising node v, we have

$$|\Psi_C \cap \Psi_v| \ge |\widehat{C}|.$$

Proof. Let C, M and v be as described above. We have $|\Psi_C| \ge \Delta - |\check{C}| + |M|$ because Ψ_C loses at most one color per colored node but saves one for each color used by the colorful matching. Since nodes are either colored or uncolored, i.e. $|C| = |\hat{C}| + |\check{C}|$, we can lower bound the number of colors in the clique palette by $|\Psi_C| \ge |\hat{C}| + \Delta - |C| + |M|$. On the other hand, observe that $\Delta \ge |N(v) \cap C| + e_v$ and $|C| = |N(v) \cap C| + a_v$. Hence, we have $|\Psi_C| \ge |\hat{C}| + |M| + e_v - a_v \ge |\hat{C}| + e_v$, using that v is promising. The lemma follows as $|\Psi_C \cap \Psi_v| \ge |\Psi_C| - e_v \ge |\hat{C}|$.

In our setting, the existence of such a matching is not enough; we must compute it in few rounds. In Section 7, we describe how to compute a colorful matching of $\Theta(K \cdot \bar{d}_C)$ anti-edges in O(K) rounds for any $K = O(1/\varepsilon)$.

Theorem 4. Let K > 0 be a constant such that $K < 1/(18\varepsilon)$. There is a $O(\log \Delta)$ -round algorithm computing a colorful matching of size at least $K \cdot \overline{d}_C$ with high probability in all cliques C of average anti-degree $\overline{d}_C \ge 1/(2\alpha)$.

Corollary 5.2. After Step 2, there are at most Δ/α unpromising nodes in C.

Proof. In a clique C with $\bar{d}_C \leq 1/(2\alpha)$, at most $|C|/(2\alpha)$ nodes have anti-degree at least 1, by Markov inequality. In a clique C with $\bar{d}_C \geq 1/(2\alpha)$, we compute a colorful matching M with $2\alpha \cdot \bar{d}_C$ edges. By Markov inequality, at most $|C|/(2\alpha)$ nodes have anti-degree more than |M|. In both cases, at most $(1 + \varepsilon)\Delta/2\alpha \leq 2\Delta/(2\alpha) = \Delta/\alpha$ nodes are unpromising.

Step 3: Coloring Dense Nodes

Reducing the number of uncolored nodes. 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 1. 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 5.3. 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.

Proof. Consider a clique C with $k \ge \Delta/(\alpha\beta)$ uncolored nodes. By Lemma 5.1, every node has $|\Psi_C \cap \Psi_v| \ge k$ colors in its palette. A node is set as active (independently) with probability 1/4. For a node v, denote its uncolored degree by \hat{d}_v . If $\hat{d}_v < |\Psi_v|/2$, a random color in Ψ_v colors v with probability 1/2. Otherwise, by the classic Chernoff bound, with probability $1 - e^{-\Omega(\hat{d}_v)} \ge 1 - e^{-\Omega(k)} \ge 1 - 1/\operatorname{poly}(n)$, node v has at most $\hat{d}_v/2$ active neighbors. Therefore, for any conditioning on the colors tried by active neighbors, v retains a uniform random color $c \in \Psi_v$ with probability 1/2.

If v samples $t = \alpha\beta$ colors in $[\Delta + 1]$ independently, it fails to find at least one color from its palette with probability $(1 - |\Psi_v \cap \Psi_C|/\Delta)^t \leq (1 - 1/(\alpha\beta))^t \leq \exp(-t/(\alpha\beta)) \leq 1/2$. Overall, a fixed node vretains a color with probability $1/4 \times 1/2 \times 1/2 = 1/16$. So the expected number of uncolored nodes $\mathbb{E}[|\hat{C}|]$ in C decreases by constant factor each round. By Chernoff with domination Lemma A.2, it holds with probability $1 - e^{-\Omega(|\hat{C}|)} = 1 - 1/\operatorname{poly}(n)$ at each round (because $\Delta \geq \Omega(\log^4 n)$). After $O(\log(\alpha\beta)) = O(\log \log n)$ rounds, with high probability, $|\hat{C}| < \Delta/(\alpha\beta)$.

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 2.2.3.

Theorem 5. Assume all cliques have at most $\Delta/(\alpha\beta)$ uncolored nodes and at most Δ/α unpromising nodes. There is a $O(\log \Delta)$ -round algorithm AUGPATH that colors a constant fraction of the nodes in each clique with high probability.

Before giving a detailed description and proof of the AUGPATH algorithm in Section 6, we conclude this section with the proof of our main theorem.

Proof of Theorem 1

If $\Delta = O(\log^4 n)$, nodes can store all their adjacent edges and color the graph in $O(\log^3 \log n)$ rounds of CONGEST [BEPS16, GK21].

Assume now $\Delta \ge \Omega(\log^4 n)$. We precondition almost-cliques (using Theorem 3) with $\eta = O(\log n)$ (Eq. (3)) in $O(\log \Delta)$ rounds and using $O(\log^2 n)$ random colors. The colorful matching requires $O(\log \Delta)$ rounds (Theorem 4) and almost-cliques have at most Δ/α unpromising nodes (Corollary 5.2). For $O(\log \log n) = O(\log \Delta)$ rounds, nodes try random colors from their palettes. All cliques are left with $\Delta/(\alpha\beta)$ uncolored nodes (by Lemma 5.3). We run AUGPATH for $O(\log \Delta)$ times. Each time, the number of uncolored nodes decreases by a constant factor with high probability (Theorem 5). Overall, we use $O(\log^2 \Delta)$ rounds to complete the coloring.

6 Augmenting Paths

This section is dedicated to the central argument of Theorem 1.

Theorem 5. Assume all cliques have at most $\Delta/(\alpha\beta)$ uncolored nodes and at most Δ/α unpromising nodes. There is a $O(\log \Delta)$ -round algorithm AUGPATH that colors a constant fraction of the nodes in each clique with high probability.

We first give a high-level description of the complete algorithm in Section 6.1. Section 6.2 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 6.3.

6.1 Detailed Description of the Algorithm

Algorithm 3. AUGPATH.

Input: A partial coloring such that each almost-clique has $|\hat{C}| < \Delta/(\alpha\beta)$ uncolored nodes, at most Δ/α unpromising nodes, and each $v \in C$ is connected to at most e_{\max} nodes in other almost-cliques.

For each almost-clique C, do in parallel:

1. Count the number of uncolored nodes $k = |\hat{C}|$.

2. F = GROWTREE (Algorithm 4).

- 3. if $k \ge \beta$, run HARVEST(k, F) for high k (Algorithm 5).
- 4. if $k < \beta$, run HARVEST(k, F) for small k (Algorithm 6).

Definition 8 (Augmenting Path). Let $P = u_1 u_2, \ldots, u_t$ be a path in C where u_1 is uncolored and u_i has color c_i for each $2 \leq i \leq t$. We say it is an *augmenting path* if u_t , the colored endpoint of P, knows a color $c \neq c_t$ such that if we recolor each node u_i using color c_{i+1} for $i \in [t-1]$ and u_t using c, the coloring of the graph remains proper.

From an uncolored node $u = u_1$ in a 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 $c \in [\Delta + 1]$, if c is not used in the clique P is an augmenting path; if c 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 clique before finding the one available color. To speed-up this process, we grow a tree of many augmenting paths.

Definition 9 (Augmenting Tree/Forest). An *augmenting tree* is a tree such that each root-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 Ψ_C 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 C 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 clique (Lemma 5.1). 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 Challenge 1, when growing the trees, we ensure they all grow at the same speed. More precisely, the GROWTREE algorithm (Algorithm 4) takes as input a forest F and finds exactly β children for each leaf in Ffor $\lfloor \log_{\beta}(\Delta/(\alpha k)) \rfloor$ 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 6.4).

Bounding the number of spoiled nodes. When a leaf v attempts to recolor its path to some uncolored node, it must sample colors in $\Psi_C \cap \Psi_v$. This might be a problem for two reasons. First, if v is unpromising (Lemma 5.1). The second possibility is it that its k colors in $\Psi_C \cap \Psi_v$ 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 Ψ_C for all nodes.

Definition 10 (Spoiled Node). We say a node $v \in C$ is spoiled if $|\Psi_C \cap \Psi_v| \leq k/2$ after conditioning on the outside.

We deal with Technical Challenge 2 in two ways. We previously computed a colorful matching (Definition 6) of size $\Theta(\bar{d}_C)$ for a large enough constant to reduce the number of unpromising nodes to a sufficiently small fraction of C (Corollary 5.2). Second, we show that it is very unlikely that nodes outside of C block more than k/2 colors for a large fraction of C. It stems from the two following observations

- external neighbors in other cliques try $\Theta(\log n)$ uniform colors in $[\Delta + 1]$, and
- the preconditioning of almost-cliques reduced the external degree to $e_{\max} = O(\Delta/\log n)$. (Theorem 3)

So, with high probability, $\Omega(\Delta)$ nodes in C have $\Omega(k)$ available colors in Ψ_C . More precisely, in Lemma 6.1, we show that with high probability over the randomness outside of C, at most $3\Delta/\alpha$ nodes are spoiled in C. Then, Lemma 6.4 show that with high probability over the randomness inside C, all trees have $\Theta(\Delta/k)$ unspoiled leaves.

Harvesting trees. While Technical Challenge 2 was about the conflict with external neighbors, Technical Challenge 3 is about the conflicts inside the 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 $\Psi_C \cap \Psi_v$ 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 6.6).

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 clique the list of $\Theta(\log n)$ available colors to each of the k uncolored nodes. Conflicts are then resolved locally (by every node).

6.2 Growing the Trees

Algorithm 4. GROWTREE (for the ℓ -th iteration of AUGPATH). **Input:** An almost-clique C with $|\hat{C}| = k < \Delta/(\alpha\beta)$ uncolored nodes, a colorful matching M of size at most $2\alpha \cdot d_C$ and at most Δ/α unpromising nodes. Define $d = \left| \log_{\beta} \frac{\Delta}{\alpha k} \right|$ and $U_0 = \hat{C}$. For i = 0 to d, (G1) If i < d, let $x = 5\beta$. For 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 $L_{3,\ell}(u)$. (G2) For each active node $u \in U_i$, let $S'_u \subseteq S_u$ be the colors c that are i) unused by colored external neighbors of u and $c \notin L_{3,\ell}(\bigcup_{C' \neq C} C' \cap N(u));$ ii) unused by nodes of $B_i = U_{\leq i} \cup M$; iii) uniquely sampled: $c \notin \bigcup_{v \in U_i \setminus \{u\}} S_v$; and iv) used by a colored node $v_{u,c} \in N(u) \cap C$ (G3) For each $u \in U_i$, choose y_u arbitrary colors $c_1, \ldots, c_{y_u} \in S'_u$ where $y_u = \begin{cases} \beta & \text{if } i < d \\ |S'_u| & \text{if } i = d \end{cases}.$ Define $v_{u,j}$ as the node of $(N(u) \cap C) \setminus B_i$ with color c_j for all $j \in [y_u]$.

Let $U_{i+1} = \bigcup_{u \in U} \{v_{u,1}, \dots, v_{u,y_u}\}$ 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 C (Definition 10). Note that the probability is taken only over the randomness of nodes *outside of* C.

Lemma 6.1. Consider an almost-clique C. With high probability over $L_{3,\ell}(V \setminus C)$, almost-clique C contains at most $3\Delta/\alpha$ spoiled nodes.

Proof. By Corollary 5.2, the clique contains at least $|C| - \Delta/\alpha \ge (1 - \varepsilon - 1/\alpha)\Delta \ge (1 - 2/\alpha)\Delta$ promising nodes, and each such node v has at least k colors in $\Psi_C \cap \Psi_v$. 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 $\Psi'_v \subseteq \Psi_C \cap \Psi_v$. We consider the k|S| pairs (c, v) where $v \in S$ is selected promising node and c is a color $c \in \Psi'_v$ from its selected colors.

Let us denote by $N_{\text{ext}}(v) \stackrel{\text{def}}{=} N(v) \cap \bigcup_{C' \neq C} C'$, the external neighbors of v that might get (re)colored. Let $B = \bigcup_{v \in S} N_{\text{ext}}(v)$ the set of vertices that are external neighbors of at least one node in S. Recall that, by Theorem 3, for each $v \in C$, $|N_{\text{ext}}(v)| \leq e_{\text{max}}$. For each $u \in B$ and color $c \in [\Delta + 1]$, let $X_{u,c}$ be defined as⁴:

$$X_{u,c} = \frac{|\{v \in S \cap N(u) \text{ such that } c \in \Psi'_v\}|}{e_{\max}} \cdot \mathbb{I}[c \in L^*_{3,\ell}(u)] ,$$

⁴where $\mathbb{I}[\mathcal{E}]$ is the indicator r.v. of some event \mathcal{E}

counting how many times a $u \in B$ is in conflicting with the selected nodes S over a selected color c, re-scaled by $1/e_{\max}$ so $X_{u,c}$ is distributed in [0, 1].

Let $X = \sum_{u \in B} \sum_{c \in [\Delta+1]} X_c$. Each edge between $v \in S$ and $u \in B$ contributes $1/e_{\max}$ to X for each color $c \in \Psi'_v$ such that $c \in L_{3,\ell}(u)$. There are at most $|S|e_{\max}$ such edges, and each color $c \in \Psi'_v$ is sampled in $L_{3,\ell}(u)$ with probability at most $20\beta/\Delta$. By linearity of expectation, $\mathbb{E}[X] \leq |S| \cdot k \cdot \frac{40\beta}{\Delta} = \Theta(k\beta)$ because $|S| = \Theta(\Delta)$. Note e_{\max} cancelling itself. Random variables $X_{u,c}$ are independent, since each color is sampled independently. By Chernoff

Random variables $X_{u,c}$ are independent, since each color is sampled independently. By Chernoff Bound (Lemma A.2), $\Pr[X \ge 2\mathbb{E}[X]] \le \exp(-\mathbb{E}[X]/3) \le 1/\operatorname{poly}(n)$. Therefore, there are at most $e_{\max} \cdot \frac{40\beta k|S|}{\Delta} \le k|S|/(4\alpha)$ conflicts between the selected colors of nodes in S and the colors sampled by their external neighbors (by Eq. (3)). Therefore, at most $\frac{k|S|/(4\alpha)}{k/2} \le \Delta/\alpha$ node are left with fewer than k/2 colors in $\Psi_v \cap \Psi_C$.

Growing the Forest. Henceforth, we fix the random lists $L_{3,\ell}(V \setminus C)$ such that C contains at most $3\Delta/\alpha$ spoiled nodes, which holds w.h.p. by Lemma 6.1. Let $U = \bigcup_i U_i$ and π be the values produced by Algorithm 4, 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 $|S_u| \ge y_u$. Then F is a forest of β -ary trees of depth d + 1 and at most $6\Delta/(\alpha k)$ leaves. We reveal the randomness inside C 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 6.2. Let $0 \leq i < d$. Then, for any lists in $L_{3,\ell}^{\mathsf{G}}(U_{\leq i})$, if a node u samples a fresh color c, we have

 $\Pr_{c \in [\Delta+1]} [c \text{ violates a condition in Step } (\mathbf{G2}) | U_{\leqslant i}] \leqslant 12/\alpha \ .$

Proof. For a fixed i < d, we have $|U_i| \leq k\beta^i$. We bound the number of colors c might be conflicting with for each item in Step (G2):

- i) Node u has at most $\varepsilon \Delta$ colored external neighbors (Theorem 3, Property 2). The number of colors in $L_{3,\ell}(\bigcup_{C'\neq C} C' \cap N(u))$ is at most $40\beta e_{\max} < \Delta/\alpha$ by Eq. (3).
- ii) For $i \leq d$, the number of nodes in $U_{\leq i}$ is $\sum_{j=0}^{d} |U_j| \leq 2k\beta^d \leq 2\Delta/\alpha$. Adding the colorful matching, at most $\Delta/\alpha + 2\alpha \bar{d}_C \leq (1/\alpha + 2\varepsilon\alpha)\Delta \leq 3\Delta/\alpha$ nodes are colored in B_i (by Eq. (2)).
- iii) The number of colors sampled (and thus blocked) by active nodes is $x|U_i| = xk\beta^i \leq 6k\beta^d \leq 6\Delta/\alpha$ where the first inequality comes from i < d.
- iv) The number of uncolored nodes in C is at most $\Delta/\alpha\beta$; hence, the number of colors used in $N(u) \cap C$ is be at least⁵ $(1 \varepsilon)\Delta \Delta/\alpha\beta \ge (1 2/\alpha)\Delta$ by Eq. (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 6.2 implies the following corollary.

Corollary 6.3. With high probability over $L_{3,\ell}^{\mathsf{G}}(U_{\leq d})$, for each i < d and $u \in U_i$, $|S'_u| \ge \beta \stackrel{\text{def}}{=} y_u$.

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.

⁵Note that, apart for the colorful matching, each node in C uses a different color

Lemma 6.4. Let U_d be the set given by Algorithm 4. With high probability over $L_{3,\ell}^{\mathsf{G}}(U_d)$, the number of unspoiled nodes in U_{d+1} is at least Δ/α .

Proof. For each node $u \in U_{d-1}$, define a random variable $X_{u,i}$ for each of its sampled color $i \in [x]$. Let $X_{u,i}$ be one if and only if the *i*-th color it samples 1) has no conflict in Step (G2) and 2) the corresponding $v_{u,c}$ is unspoiled. The analysis is similar to Lemma 6.2 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 6.1. Indeed, there are at most $3\Delta/\alpha$ spoiled nodes in C. 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 $x|U_d| \leq 6\Delta/\alpha$.

If we union bound over all possible failures for a random color $c \in [\Delta + 1]$, we get $\Pr[X_{u,c} = 1|U_d \setminus \{u\}] \leq 15/\alpha \leq 1/25$. 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 A.2) shows that, w.p. $1 - e^{-\Omega(|U_d|)}$, at most $|U_d|/4$ nodes sample more than x/5 bad colors. Note that, by definition of d,

$$|U_d| = k\beta^d \ge k\beta^{\log_\beta(\Delta/\alpha k) - 1} \ge \frac{\Delta}{\alpha\beta}$$
.

Therefore, $1 - e^{-\Omega(|U_d|)} \ge 1 - e^{-\Omega(\Delta/\beta)} \ge 1 - \text{poly}(n)$ (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

$$\frac{|U_d|}{4} \cdot \frac{4x}{5} \ge \frac{|U_d|}{5} \left(\frac{6\Delta}{\alpha |U_d|} - 1\right) \qquad (\text{because } x \stackrel{\text{def}}{=} \left\lfloor \frac{6\Delta}{\alpha |U_d|} \right\rfloor)$$
$$= \frac{6\Delta/\alpha - |U_d|}{5} \ge \frac{\Delta}{\alpha} , \qquad (\text{because } |U_d| \le \Delta/\alpha)$$

which concludes the proof of the Lemma.

6.3 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 6.4, 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 6.5. 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 (G2) of Algorithm 4, 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 $k \ge \Omega(\log n)$.

Algorithm 5. HARVEST (for k greater than $\Omega(\log n)$). **Input:** the forest F of augmenting trees computed by Algorithm 4.

- (H1) Leaves $v \in F$ try one fresh color $c_v \in L_3^{\mathsf{H}}(v)$. We call v successful if it can adopt c_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.
- (H2) Each leaf can learn in O(1) rounds if a leaf from another tree sampled the same color (by Claim 4.7). We call a tree *successful* if it has at least one successful leaf that is not conflicting with leaves from other trees.
- (H3) If a tree is successful, it can recolor the path from its root to its successful leaf in O(d) rounds.

Lemma 6.6. Let C be a clique with $\beta \leq k \leq \Delta/(\alpha\beta)$ uncolored nodes. In Step (H3), with high probability over $L_{3,\ell}^{\mathsf{H}}(C)$, we color at least $\Omega(k/\alpha)$ uncolored nodes.

Proof. Let k' = 0.9k and $T_1, \ldots, T_{k'}$ be k' trees with at least $\Delta/(2\alpha k)$ successful 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 $\Psi_C \cap \Psi_v$, which we call its selected colors.

For each $i \in [k']$, let 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 X_i may be expressed as the difference between two random variables Y_i and Z_i , where Y_i corresponds to the event that at least one selected leaf of T_i tries one of its selected colors, and 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:

$$\Pr[X_i = 1] \ge \frac{\Delta}{2\alpha k} \cdot \frac{k}{2\Delta} \cdot \left(1 - \frac{k/2 + 1}{\Delta}\right)^{\Delta/(2\alpha k) - 1} \cdot \left(1 - \frac{1}{\Delta}\right)^{6\Delta/\alpha}$$

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 c. The third term corresponds to all other selected leaves of T_i sampling neither one of their selected colors nor c. The last term corresponds to all leaves not trying c. Using $1-x/2 \ge e^{-x}$ for $x \in [0, 1]$ (Lemma A.1), $\Pr[X_i = 1] \ge e^{-12/\alpha}/(4\alpha)$. Let $X = \sum_{i=1}^{k'100} X_i$, by linearity $\mathbb{E}[X] \ge k' \cdot e^{-12/\alpha}/(4\alpha) = \Omega(k/\alpha)$.

Consider similarly the random variables Y_i and their sum Y. We have:

$$\Pr[Y_i = 1] = 1 - \left(1 - \frac{k}{2\Delta}\right)^{\Delta/(4\alpha k)} \ge \frac{1}{8\alpha + 1}$$

Therefore, $\mathbb{E}[X]$ and $\mathbb{E}[Y]$ are both of order $\Theta(k/\alpha)$. Additionally, Y is 1-Lipschitz and 1-certifiable, and Z = Y - X is 2-Lipschitz and 2-certifiable. Hence, by Lemma A.4, $\Pr[X < k' \cdot e^{-12/\alpha}/(8\alpha)] \leq \exp(-\Omega(k/\alpha))$. Since X is a lower bound on the number of trees that successfully recolor their root, at least $\Omega(k/\alpha)$ uncolored nodes get colored, w.h.p.

Coloring the last nodes. Assume now that only $k = O(\log n)$ uncolored nodes remain.

Algorithm 6. HARVEST (for k smaller than $O(\log n)$). **Input:** the forest F of augmenting trees computed by Algorithm 4.

- (L1) 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 $\{S_v, v \in \widehat{C}\}$. All nodes then locally compute the same matching of size $\Omega(k)$ in the graph with vertices $\widehat{C} \cup [\Delta + 1]$ and edges (v, c) iff $c \in S_v$.

Sending the sets $\{S_v, v \in \hat{C}\}$ in Step (L3) is done by RANDOMPUSH.

Algorithm 7. RANDOMPUSH.

Input: An almost-clique C 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 6.7. Let $x \leq \beta^3$ messages of $O(\log n)$ bits each known by exactly one node in the sparsified almost-clique \tilde{C} . After $O(\log \Delta)$ iterations of RANDOMPUSH, each node in the sparsified almostclique learns all x messages, with high probability.

The proof of Lemma 6.7 follows easily from the expansion of the sparsified almost-clique (see Lemma 4.5), and we defer it to Appendix B.3.

Lemma 6.8. Let C be a clique with $k \leq \beta$ uncolored nodes. At the end of Step (L3), with high probability over $L_{3,\ell}^{\mathsf{H}}(C)$, we color at least $\Omega(k/\alpha)$ uncolored nodes.

Proof. There exist k' = 0.9k uncolored nodes with at least $\Delta/(2\alpha k)$ successful paths to unspoiled leaves.

Let us now argue that an uncolored node v with this many successful 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 a random variable X_i such that:

- If previous leaves discovered k/4 distinct colors already, then $X_i = 1$ w.p. 1,
- If previous leaves discovered fewer than k/4 distinct colors, then $X_i = 1$ iff leaf number i discovers a new color.

When 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:

$$\mathbb{E}[X_i] = 1 - \left(1 - \frac{k}{4\Delta}\right)^{\beta} \ge 1 - \frac{1}{1 + k\beta/(4\Delta)} = \frac{k\beta}{4\Delta + k\beta} \ge \frac{k\beta}{5\Delta}$$

where we used Lemma A.1 for the first step and the last comes from $\Delta \ge \beta^2 \ge k\beta$. Therefore, $\mathbb{E}[\sum_i X_i] \ge \frac{\Delta}{2\alpha k} \cdot \frac{k\beta}{5\Delta} = \beta/(10\alpha)$. The series of X_i satisfy the conditions of Lemma A.2, hence it has value at least $\beta/(20\alpha)$, w.h.p. By definition of $\{X_i\}$, if $\beta/(20\alpha) \ge k/4$, this gives

that at least k/4 colors are found in the tree, while if $\beta/(20\alpha) \leq k/4$, we only get that at least $\beta/(20\alpha) \geq k/(20\alpha)$ colors are found in the tree (as $\beta \geq k$).

We now argue that our algorithm has the claimed runtime. Nodes can share their sampled colors with nodes in other cliques in $O(\log \Delta)$ rounds as the total amount of bits to communicate is $O(\log n \cdot \log \Delta)$.

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 (ID_v, c) for each of the colors $c \in S_v$, 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 $\hat{C} \cup [\Delta + 1]$ and edges (v, c) such that $c \in S_v$ is now known to all nodes in C. 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.

7 Colorful Matching

In this section, we show the following theorem:

Theorem 4. Let K > 0 be a constant such that $K < 1/(18\varepsilon)$. There is a $O(\log \Delta)$ -round algorithm computing a colorful matching of size at least $K \cdot \bar{d}_C$ with high probability in all cliques C of average anti-degree $\bar{d}_C \ge 1/(2\alpha)$.

Throughout this section, we fix a clique C and fix the colors used and sampled outside of C adversarially. At the beginning of this step, nodes of C are uncolored. For a set D of colors, define $\operatorname{avail}_D(e)$ to be the number of colors that an anti-edge e can adopt in D without conflicting with external neighbors. This includes all possible colors in L_2 that external neighbors might use to color themselves. If D contains a single color c, we abuse notation and denote $\operatorname{avail}_D(e)$ by $\operatorname{avail}_c(e) \in \{0, 1\}$. By extension, for a set F of anti-edges $\operatorname{avail}_D(F) = \sum_{e \in F} \operatorname{avail}_D(e)$. A similar quantity was introduced by [ACK19]. A major difference with [ACK19] is that colors become unavailable if an uncolored external neighbor merely samples it in L_2 . In particular, one uncolored external neighbor blocks $\Theta(\log n)$ colors. We can afford to lose so many colors because of preconditioning and (in contrast to Section 6), 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 C.

Lemma 7.1. Let $D = [\Delta + 1]$ and F be the set of all anti-edges in C. For any (possibly adversarial) conditioning outside of C, we have $\operatorname{avail}_D(F) \ge \overline{d}_C \Delta^2/3$.

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 $L_{2,i}$ for $i \in [O(1/\varepsilon)]$ and L_2^* . By Claim 4.1, an uncolored neighbor in another clique blocks at most $L_{\max} = O(\log n)$ colors. Since a node in C has at most $\varepsilon \Delta$ colored neighbors (necessarily outside C) and at most $e_{\max} = \Delta/\eta$ neighbors in other cliques (by Property 3

of Theorem 3 and Eq. (3)), we have $\operatorname{avail}_D(e) \ge \Delta - \varepsilon \Delta - L_{\max} \cdot \Delta/\eta \ge (1 - 2\varepsilon)\Delta$. Summing over all edges, we get

$$\operatorname{avail}_D(F) = \sum_e \operatorname{avail}_D(e) \ge \frac{\bar{d}_C |C|}{2} (1 - 2\varepsilon) \Delta \ge \bar{d}_C \Delta^2 / 3$$
.

To compute a colorful matching, we greedily add same-colored anti-edges to M. Each time we do so, we remove the color of that edge from D, which then becomes unavailable to other edges, and we remove the matched edge as well as all its adjacent edges from F. We argue that as long as the total number of available colors is large, there must be colors available to many edges. We call a color c heavy if $\operatorname{avail}_c(F) \ge \overline{d}_C \Delta/20$. The following claim is immediate from the limited contributions to $\operatorname{avail}_D(F)$ from both all non-heavy colors and from each heavy color individually.

Claim 7.2. As long as avail_D(F) $\geq \bar{d}_C \Delta^2/6$, there are at least $\Delta/10$ heavy colors in D.

When \bar{d}_C is large. We first run the following algorithm. It produces a large enough matching in cliques with $\bar{d}_C \ge \beta$.

Algorithm 8. MATCHING. Input: a constant $0 < K < 1/(18\varepsilon)$. Output: a colorful matching M_C of size $K \cdot \overline{d}_C$ in each almost-clique such that $\overline{d}_C \ge \beta$.

Initially, $M_C = \emptyset$ for each almost-clique C. For i = 1 to $5 \cdot 10^3 \cdot K$, in each almost-clique C in parallel:

- 1. Each uncolored $v \in C$ is active. It samples each $c \in [\Delta + 1]$ independently into $L_{2,i}(v)$ with probability $p = 1/(4\Delta)$.
- 2. Each $v \in C$ sampling less or fewer than 1 color $(|L_{2,i}(v)| \neq 1)$ becomes inactive.
- 3. Each $v \in C$ sampling a single color c becomes inactive if c is used by a colored external neighbor, by an anti-edge in M_C , or is sampled by an external neighbor of v.
- 4. Each *active* node retains its color if it has an *active* anti-neighbor with this color. If the same color is used by two or more anti-edges, we keep the anti-edge with the smallest ID.

Lemma 7.3. Let K > 0 be a constant such that $K < 1/(18\varepsilon)$. W.p. $1 - \exp(-\Omega(\bar{d}_C))$ over the randomness in $L_2(C)$, the set M_C produced by MATCHING is a colorful matching in C of size at least $K \cdot \bar{d}_C$.

Proof. Suppose $\operatorname{avail}_D(F) \ge \overline{d}_C \Delta^2/6$. For each color c, define A_c as the indicator random variable of the event that at least one anti-edge in C samples c. For a heavy color $c \in D$, we bound $\Pr[A_c = 1]$ from below by the probability that exactly one anti-edge samples c:

$$\Pr[A_c = 1] \ge \sum_{e \in F} \Pr[\text{endpoints of } e \text{ are the only nodes to sample } c]$$

$$\ge \operatorname{avail}_c(F) \cdot p^2 \cdot (1-p)^{|C|-2}$$

$$\ge \frac{\bar{d}_C \Delta}{20} \cdot \frac{1}{16\Delta^2} \cdot \exp(-2p|C|) \qquad (\text{because } c \text{ is heavy})$$

$$\ge \frac{\bar{d}_C}{320e\Delta} . \qquad (\text{because } |C| \le (1+\varepsilon)\Delta)$$

By Claim 7.2, there exist at least $\Delta/10$ such heavy colors in D. Therefore, $\mathbb{E}[A] = \mathbb{E}[\sum_{c \in D} A_c] \ge \bar{d}_C/320e$. Since each color is sampled independently, random variables A_c are independent and we can apply the classic Chernoff bound. With probability $1 - \exp(-\Omega(\bar{d}_C))$, we have $A \ge \bar{d}_C/640e$.

Random variable A is not the number of anti-edges we can insert in M because we do not account for nodes sampling more than one color (Step 2). We emphasize that nodes can adopt any color available to them: by definition of avail it cannot be conflicting on the inside nor with colored nodes in M. Let B_c be the random variable equal to one if and only if at exactly one anti-edge sampled c (i.e., $A_c = 1$) and at least one endpoint becomes inactive in Step 2. Condition on $A_c = 1$ and let e be the anti-edge that sampled c. Since an endpoint samples each color independently with probability p, it only samples c with probability $(1 - p)^{\Delta} \ge \exp(-1/2) \ge 3/5$ (by Eq. (11)). Therefore, the probability that one endpoint of e sampled more colors is at most $2 \times (1 - 3/5) = 4/5$. We overestimate the number of anti-edges inserted in M by $B = \sum_{c \in D} B_c$ which, in expectation, is $\mathbb{E}[B] = \sum_{c \in D} \mathbb{E}[B_c|A_c = 1]\mathbb{E}[A_c] \le (4/5) \cdot \mathbb{E}[A]$. Random variable B is 2-Lipschitz. It is also 2-certifiable, because to certify that $B_c = 1$ we can point at two colors sampled by one of the endpoints. By Talagrand inequality (Lemma A.4), with probability $1 - \exp(-\Omega(\bar{d}_C))$, we add $A - B \ge \mathbb{E}[A - B]/2 \ge \frac{\bar{d}_C}{5 \cdot 10^3}$ anti-edges to the colorful matching.

This shows that as long as $\operatorname{avail}_D(F) \geq \overline{d}_C \Delta^2/6$, we add $\frac{\overline{d}_C}{5 \cdot 10^3}$ anti-edges to the matching at each iteration. Hence, after $5 \cdot 10^3 \cdot K$ iterations, the matching has $K \cdot \overline{d}_C$ anti-edges. Observe that when we insert an anti-edge in M_C , we remove one color c from D and at most $2\varepsilon\Delta$ anti-edges from F. Color c contributed at most $\operatorname{avail}_c(F) \leq \overline{d}_C \Delta \leq \varepsilon \Delta^2$ to $\operatorname{avail}_D(F)$ and each anti-edge at most Δ . Inserting an anti-edge in M_C decreases $\operatorname{avail}_D(F)$ by at most $3\varepsilon\Delta^2$. If, after some iterations, the number of available colors drops below $\overline{d}_C \Delta^2/6$, we must have inserted at least $\frac{\overline{d}_C \Delta^2/6}{3\varepsilon\Delta^2} = \overline{d}_C/(18\varepsilon) > K \cdot \overline{d}_C$ anti-edges into M.

When \bar{d}_C is small. If $\bar{d}_C \leq O(\log n)$, Lemma 7.3 fails to compute a colorful matching with high probability. In this section, we explain how to compute a large enough matching in cliques with small anti-degree. Note that nodes can count the number of anti-edges in the matching in $O(\log \Delta)$ rounds. If they find fewer than $K\beta$ anti-edges, it must be that $\bar{d}_C < \beta$. We then uncolor all nodes in C and run Algorithm 9.

Intuitively, since $\bar{d}_C \ge \Omega(1)$, if we repeat the previous procedure $\Theta(\log n)$ times, we would get a failure probability of $\left(e^{-\Theta(\bar{d}_C)}\right)^{\Theta(\log n)} = 1/\operatorname{poly}(n)$. This implies that even when \bar{d}_C is a small constant, a colorful matching of size $\Theta(\bar{d}_C)$ exists. This was, in fact, already shown in the first palette sparsification theorem.

Lemma 7.4 ([ACK19, Lemma 3.2]). Let C be a ε -almost clique, $D \subseteq [\Delta + 1]$ and F a subset of anti-edges in C. Fix any partial coloring given by Theorem 3 where nodes of C are uncolored and $\operatorname{avail}_D(F) \ge \overline{d}_C \Delta^2/3$. Suppose each node sample colors in $[\Delta + 1]$ independently with probability $q \stackrel{\text{def}}{=} \gamma \frac{\beta}{\Delta}$ for some constant $\gamma(\varepsilon) = \gamma > 0$ (depending on ε but not n nor Δ). Then there exists a colorful matching of size at least $\overline{d}_C/(414\varepsilon)$ with high probability.

Note that our definition of avail is stronger than the one of [ACK19] because it removes colors sampled by active external neighbor. Regardless of that, Lemma 7.1 shows that we have a large number of available colors, which is the only requirement for the proof of [ACK19].

Algorithm 9. MATCHING (for almost-cliques C with $\bar{d}_C < \beta$ in parallel).

- 1. Each $v \in C$ samples each $c \in [\Delta + 1]$ into $L_2^*(v)$ independently with probability $q \stackrel{\text{def}}{=} \gamma \frac{\beta}{\Delta}$.
- 2. Each node computes $S_v \stackrel{\text{def}}{=} L_2^*(v) \setminus L_2^*(V \setminus C)$, the set of colors that do not collide with those of external neighbors.

Let $S'_v \stackrel{\text{def}}{=} \{c \in S_v : \exists u \in C \setminus N(v) \text{ such that } c \in S_u\}$ be the colors of v sampled by at least one anti-neighbor.

- 3. We count the number of candidate anti-edges in C: the number of pairs (uv, c) where uv is an anti-edge and $c \in S'_v \cap S'_u$. If there are more than $U \stackrel{\text{def}}{=} 4\gamma^2 K \bar{d}_C \beta^2$ 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 (ID_v, c) for each $c \in S'_v \cap D$.

Use RANDOMPUSH to disseminate all messages (ID_v, c) within C.

Each node v with $c \in S'_v \cap D$ forms a message (ID_u, ID_v, c) for each anti-neighbor u with $c \in S'_u \cap D$.

Use RANDOMPUSH to disseminate all messages (ID_u, ID_v, c) within C.

5. Compute locally the colorful matching using anti-edges disseminated in the previous step.

Lemma 7.5. Let $K \in (0, 1/(18\varepsilon))$ be a constant. Consider all cliques with $1/(2\alpha) \leq \overline{d}_C \leq \beta$. If nodes sample each color independently with probability $q = \gamma \frac{\beta}{\Delta}$ where γ is the constant from Lemma 7.4, then in each clique C, with high probability, there exists a colorful matching that does not conflict with nodes on the outside. Moreover, Algorithm 9 finds this matching in $O(\log \Delta)$ rounds.

Proof. We first explain how nodes compute S'_v . Each node v starts by broadcasting $L_2^*(v)$ in $O(\log \Delta)$ rounds (since $|L_2^*(v)| = O(\log n)$). We run a BFS for each color; two hops suffice by Claim 4.7. To learn 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 $v \in C$ know for each $c \in S_v$ how many other nodes $u \in C$ have $c \in S_u$. 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, c) where u and v are anti-neighbor and c is a color such that $c \in S'_v \cap S'_u$. Namely, we could add edge uv to the colorful matching using color c. 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 C and aggregate the total number of candidate edges.

Suppose that the number of candidate edges is at most $U \stackrel{\text{def}}{=} 4\gamma^2 \cdot K \bar{d}_C \beta^2 = O(\beta^3)$. For each candidate edge (uv, c), we craft two messages (ID_u, c) and (ID_v, c) . 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 6.7). 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 7.4, a colorful matching of size $K \bar{d}_C$ 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 4.2, each node is contained in at most $2\gamma^2\beta^2$ 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 $K\bar{d}_C$, 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 $4\gamma^2\beta^2$ edges from F. Since we assumed F contained $U = 4\gamma^2 \cdot K\bar{d}_C\beta^2$ anti-edges, the algorithm always finds a colorful matching of $K\bar{d}_C$ 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 C selects a subset D of the colors such that the number of candidate edges with that color 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 c if the leader for color c 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 C and x_1, \ldots, x_t the number of candidate edges held by each subtree. Let i be the smallest index in [t] such that $\sum_{j \leq i} x_j \geq U$. We select all colors whose leaders are in subtrees 0 to i - 1. We selected $\sum_{j < i} x_j$ 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. At this point each $v \in C$ knows which color belongs to a selected candidate edge, i.e., colors such that $c \in S'_v \cap D$.

8 Lower Bound

As discussed in Section 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 Bbe 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 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 $d \ge 0$, 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 $d \ge 0$, the number of edges of M between nodes in V_d and nodes in V_{d+1} is equal to

$$S_d \stackrel{\text{def}}{=} \sum_{i=0}^i (-1)^i \cdot |V_d|.$$

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, \ldots, V_d . As sketched in Section 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 6. 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 $p \leq \operatorname{poly} \log(n)/n$ and $p \geq \alpha \ln(n)/n$ 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 $p \ge \alpha \ln(n)/n$ and the constant α is chosen sufficiently large, then B has a perfect matching w.h.p. This is well-known [Bol98, 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 $\ge 2/3$, 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 $d \ge 0$, 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 $d \in \{\ell + 1, \ldots, h + T + 1\}$ 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 $(V_0 \cup \cdots \cup V_{\ell-1}) \cup (V_{\ell+1} \cup \cdots \cup V_{h+T+1})$. 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 $(V_0 \cup \cdots \cup V_{\ell-1}) \cup (V_{\ell+1} \cup \cdots \cup V_{h+T+1})$, 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 $(V_0 \cup \cdots \cup V_{\ell-1}) \cup (V_{\ell+1} \cup \cdots \cup V_{h+T+1})$, 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 \mathcal{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

$$p_{\text{est},K} \stackrel{\text{def}}{=} \max_{x_{\ell} > 0} \Pr\left(X_{\ell} = x_{\ell} \,|\, \mathcal{K} = K\right)$$

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, \mathcal{K} takes on a "good" state K for which $p_{\text{est},K} \leq 1/2 + o(1)$. 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 $1/2 + o(1) \leq 2/3$ 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 $\mathcal{K} = K$ more closely. First note that by symmetry, the probability $\Pr(X_\ell = x | \mathcal{K} = K)$ 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 of X_ℓ only depends on the additional edges of the nodes in $X_{\ell-1}$ and $X_{\ell+1}$. The probability $\Pr(X_\ell = x | \mathcal{K} = K)$ 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 $V_{\geq d} = V_d \cup V_{d+1} \cup \ldots$ 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 $\mathcal{X}_{\leq d} \stackrel{\text{def}}{=} \mathcal{X}_0 \cap \cdots \cap \mathcal{X}_{d-1}, \mathcal{X}_{\leq d} = \mathcal{X}_{\leq d} \cap \mathcal{X}_d, \mathcal{X}_{\geq d} \stackrel{\text{def}}{=} \mathcal{X}_{d+1} \cap \cdots \cap \mathcal{X}_{h+T+1}$, as well as $x_{\leq d} \stackrel{\text{def}}{=} x_0 + \cdots + x_{d-1}$ and $x_{\leq d} = x_{\leq d} + x_d$. 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 $V_{\geq d}$ to $V_{d-1} \cup V_{\geq d}$ is revealed, then the size X_d of V_d is binomially distributed with parameters $|V_{\geq d}|$ and $q_d \stackrel{\text{def}}{=} 1 - (1-p)^{x_{d-1}}$. For all $d \geq 1$, we therefore have

$$\Pr(X_d = x_d \,|\, \mathcal{X}_{< d}) = \binom{n - x_{< d}}{x_d} \cdot q_d^{x_d} \cdot (1 - q_d)^{n - x_{\leq d}}, \text{ where } q_d \stackrel{\text{def}}{=} 1 - (1 - p)^{x_{d-1}}.$$
(4)

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 (4) iteratively, we obtain

$$\Pr(\mathcal{X}_{\leq h+T+1}) = \Pr(\mathcal{X}_{<\ell} \land X_{\ell} = x_{\ell} \land \mathcal{X}_{>\ell})$$

$$= \Pr(\mathcal{X}_{<\ell}) \cdot \prod_{i=\ell}^{h+T+1} \Pr(X_i = x_i \mid \mathcal{X}_{
$$= \Pr(\mathcal{X}_{<\ell}) \cdot \prod_{i=\ell}^{h+T+1} \binom{n-x_{ (5)$$$$

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 $d \ge 1$, we define the even $\mathcal{X}'_{< d}$ as follows. If $d \le \ell$, 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 $\mathcal{X}'_{\leq d}$ analogously. We then have

$$\Pr(\mathcal{X}'_{\leq h+T+1}) = \Pr(\mathcal{X}_{<\ell}) \cdot \Pr(X_{\ell} = x_{\ell} - 1 \mid \mathcal{X}_{<\ell}) \cdot \prod_{i=\ell+1}^{h+T+1} \Pr(X_i = x_i \mid \mathcal{X}'_{$$

In order to compare $\Pr(\mathcal{X}_{\leq h+T+1})$ and $\Pr(\mathcal{X}'_{\leq h+T+1})$, we therefore need to compare $\Pr(X_{\ell} = x_{\ell} | \mathcal{X}_{<\ell})$ and $\Pr(X_{\ell} = x_{\ell} - 1 | \mathcal{X}_{<\ell})$, as well as $\Pr(X_i = x_i | \mathcal{X}_{<i})$ and $\Pr(X_i = x_i | \mathcal{X}'_{<i})$ for all $i \geq \ell + 1$. We have

$$\Pr(X_{\ell} = x_{\ell} - 1 \mid \mathcal{X}_{<\ell}) = \binom{n - x_{<\ell}}{x_{\ell} - 1} \cdot q_{\ell}^{x_{\ell} - 1} \cdot (1 - q_{\ell})^{n - x_{\leq \ell} + 1}$$

$$= \frac{\binom{n - x_{<\ell}}{x_{\ell} - 1} \cdot q_{\ell}^{x_{\ell} - 1} \cdot (1 - q_{\ell})^{n - x_{\leq \ell} + 1}}{\binom{n - x_{<\ell}}{x_{\ell}} \cdot q_{\ell}^{x_{\ell}} \cdot (1 - q_{\ell})^{n - x_{\leq \ell}}} \cdot \Pr(X_{\ell} = x_{\ell} \mid \mathcal{X}_{<\ell})$$

$$= \frac{x_{\ell} \cdot (1 - q_{\ell})}{(n - x_{\leq \ell} + 1) \cdot q_{\ell}} \cdot \Pr(X_{\ell} = x_{\ell} \mid \mathcal{X}_{<\ell}). \tag{6}$$

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

$$\Pr(X_{\ell+1} = x_{\ell+1} \mid \mathcal{X}'_{<\ell+1}) = \Pr(X_{\ell+1} = x_{\ell+1} \mid \mathcal{X}_{<\ell} \land X_{\ell} = x_{\ell} - 1)$$

$$= \binom{n - x_{\leq \ell} - 1}{x_{\ell+1}} \cdot (q'_{\ell+1})^{x_{\ell+1}} \cdot (1 - q'_{\ell+1})^{n - x_{\leq \ell+1} + 1}$$

$$= \frac{\binom{n - x_{\leq \ell} - 1}{x_{\ell+1}}}{\binom{n - x_{\leq \ell}}{x_{\ell+1}}} \cdot \left(\frac{q'_{\ell+1}}{q_{\ell+1}}\right)^{x_{\ell+1}} \cdot \left(\frac{1 - q'_{\ell+1}}{1 - q_{\ell+1}}\right)^{n - x_{\leq \ell+1} + 1}} \cdot \Pr(X_{\ell+1} = x_{\ell+1} \mid \mathcal{X}_{<\ell} \land X_{\ell} = x_{\ell})$$

$$= \frac{n - x_{\leq \ell} - x_{\ell+1}}{n - x_{\leq \ell}} \cdot \left(\frac{q'_{\ell+1}}{q_{\ell+1}}\right)^{x_{\ell+1}} \cdot \left(\frac{1}{1 - p}\right)^{n - x_{\leq \ell+1} + 1}} \cdot \Pr(X_{\ell+1} = x_{\ell+1} \mid \mathcal{X}_{<\ell+1}).$$
(7)

Finally, for $i > \ell + 1$, we have

$$\Pr(X_{i} = x_{i} | \mathcal{X}_{
$$= \frac{n - x_{(8)$$$$

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 \mathcal{W} . We next define the event \mathcal{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 $f(n) \ge 32 \ln n$ and $f(n) \le \text{polylog } n$. The event \mathcal{W} is defined as follows. For all $d \in \{1, \ldots, h + T + 1\}$, it must hold that

$$|X_d - f(n) \cdot X_{d-1}| \leq \sqrt{7 \cdot f(n) \cdot X_{d-1} \cdot \ln n}.$$
(9)

Note that condition (9) and the assumption that $f(n) \ge 32 \ln n$ directly imply that $X_d \le 1.5f(n) \cdot X_{d-1}$ (even if $X_{d-1} = 1$). Therefore in well-behaved instances, X_d/X_{d-1} is at most polylog n. We choose the parameter $T = \Theta(\ln(n)/\ln\ln(n))$ small enough such that in well-behaved instances, $X_0 + \cdots + X_{h+T+1} \le n^{1/3}$. Similarly, (9) implies that $X_d \ge 0.5f(n) \cdot X_{d-1}$ and thus for any $d = \Theta(\log n/\log \log n)$, we have $X_d \ge n^{\nu}$ 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 $d \ge 1$. 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

$$\Pr(|X_d - \mathbb{E}[X_d | \mathcal{X}_{< d}]| > \delta \cdot \mathbb{E}[X_d | \mathcal{X}_{> d}] | \mathcal{X}_{< d}) \leq 2e^{-\delta^2/3 \cdot \mathbb{E}[X_d]}.$$
(10)

We will see that (10) implies that for every d, Inequality (9) holds with probability at least $1-2/n^2$. To achieve this, we first have to understand what the value of $\mathbb{E}[X_d|\mathcal{X}_{\leq d}]$ is. We first have a look at the probability q_d of the binomial distribution underlying X_d . We have

$$q_d = 1 - (1 - p)^{x_{d-1}} \leq p \cdot x_{d-1} \text{ and}$$

$$q_d = 1 - (1 - p)^{x_{d-1}} \geq 1 - e^{-px_{d-1}} \geq p \cdot x_{d-1} - (p \cdot x_{d-1})^2 \geq p \cdot x_{d-1} \cdot \left(1 - \frac{f(n)}{n^{2/3}}\right)$$

We used that for any real value $y \in [0,1]$ and any $k \ge 1$, we have $(1-y)^k \ge 1-ky$ and $1-y \le e^{-y} \le 1-y+y^2$. In the last inequality, we have further used that in well-behaved executions, $x_{d-1} \le n^{1/3}$. We have $\mathbb{E}[X_d|\mathcal{X}_{\le d}] = q_d \cdot (n-x_{\le d})$. We therefore have

$$\mathbb{E}[X_d|\mathcal{X}_{< d}] \leq px_{d-1} \cdot n = f(n) \cdot x_{d-1},$$

$$\mathbb{E}[X_d|\mathcal{X}_{< d}] \geq \left(1 - \frac{f(n)}{n^{2/3}}\right) \cdot px_{d-1} \cdot (n - n^{1/3}) \geq \left(1 - \frac{2f(n)}{n^{2/3}}\right) \cdot f(n) \cdot x_{d-1}.$$

Let ξ denote the maximum absolute deviation from the expectation that is still guaranteed to be well-behaved by (9). We can lower bound ξ as follows

$$\begin{aligned} \xi & \ge \sqrt{7f(n) \cdot x_{d-1} \cdot \ln n} - \frac{2f(n)}{n^{2/3}} \cdot f(n) \cdot x_{d-1} \\ & \ge \sqrt{7\mathbb{E}[X_d|\mathcal{X}_{$$

The last inequality holds if $n \ge n_0$ for a sufficiently large constant n_0 . Together with (10), this now implies that for all d, (9) 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 $n \ge n_0$ 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 \mathcal{W} holds). Consider some assignment to the random variables $X_1, \ldots, X_{\ell-1}$ and $X_{\ell+1}, \ldots, X_{h+T+1}$ that are consistent with (9). 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 $X_1, \ldots, X_{\ell-1}$ and $X_{\ell+1}, \ldots, X_{h+T+1}$ only allow one single value of X_ℓ such that (9) is satisfied. We need to show that even when conditioning on \mathcal{W} , this only happens with a very small probability. Let us therefore define an even $\mathcal{W}' \subseteq \mathcal{W}$ as an instance in which replacing X_ℓ by $X_\ell - 1$ or $X_\ell + 1$ still satisfies (9). Note that the above analysis has enough slack to ensure that also $\Pr(\mathcal{W}') \ge 1 - \ln(n)/n^2$ if $n \ge n_0$ for a sufficiently large constant n_0 . The same analysis for example also works if the fixed constant 7 in (9) is replaced by any smaller fixed constant that is larger than 6. We therefore have

$$\Pr(\mathcal{W}' \mid \mathcal{W}) = \frac{\Pr(\mathcal{W}' \cap \mathcal{W})}{\Pr(\mathcal{W})} = \frac{\Pr(\mathcal{W}')}{\Pr(\mathcal{W})} \ge \Pr(\mathcal{W}') \ge 1 - \frac{\ln n}{n^2}.$$

We use $x_1, \ldots, x_{\ell-1}$ and $x_{\ell+1}, \ldots, x_{h+T+1}$ 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 \mathcal{W}' holds, there is at least one such value x_ℓ and we have seen that even conditioning on \mathcal{W} , the probability of \mathcal{W}' is still at least $1 - \ln(n)/n^2$. Because we are assuming \mathcal{W} , we know that the value x_ℓ satisfies the following condition.

$$x_{\ell} = x_{\ell-1} \cdot np + \theta, \quad \text{where} \quad |\theta| \in O(\sqrt{x_{\ell-1}f(n)\log n}) = O\left(\sqrt{\frac{f(n)\log n}{x_{\ell-1}}}\right) \cdot x_{\ell-1}$$

We first look at the ratio between $\Pr(X_{\ell} = x_{\ell} - 1 | \mathcal{X}_{<\ell})$ and $\Pr(X_{\ell} = x_{\ell} | \mathcal{X}_{<\ell})$. By Equation (6), this ratio is equal to $\frac{x_{\ell} \cdot (1-q_{\ell})}{(n-x_{<\ell}+1) \cdot q_{\ell}}$. In the following, we denote this ratio by ρ_1 . By using that $x_{< h+T+1} < n^{1/3}$, we can then bound ρ_1 as follows.

$$\rho_1 = \frac{x_{\ell} \cdot (1 - q_{\ell})}{(n - x_{\leq \ell} + 1) \cdot q_{\ell}} \leq \frac{f(n) \cdot x_{\ell-1} \cdot \left(1 + O\left(\sqrt{f(n)\log(n)/x_{d-1}}\right)\right)}{\left(1 - \frac{1}{n^{2/3}}\right) \cdot n \cdot px_{\ell-1} \cdot (1 - px_{\ell-1})} \leq 1 + \frac{1}{n^{\Omega(1)}}$$

In the last inequality, we used that $x_{\ell} = x_{\ell-1} \cdot np$, that $x_{\ell-1} \leq n^{1/3}$, and that $x_{\ell-1} \geq n^{\kappa}$ for some constant $\kappa > 0$. Similarly, we have

$$\rho_1 = \frac{x_{\ell} \cdot (1 - q_{\ell})}{(n - x_{\leq \ell} + 1) \cdot q_{\ell}} \ge \frac{x_{\ell-1} \cdot \left(1 - O\left(\sqrt{f(n)\log(n)/x_{d-1}}\right)\right) \cdot (1 - px_{\ell-1})}{n \cdot px_{\ell-1}} \ge 1 - \frac{1}{n^{\Omega(1)}}.$$

We next look at the ratio between $\Pr(X_{\ell+1} = x_{\ell+1} \mid \mathcal{X}'_{<\ell+1})$ and $\Pr(X_{\ell+1} = x_{\ell+1} \mid \mathcal{X}_{<\ell+1})$. We

denote this ratio by ρ_2 . By Equation (7), ρ_2 can be written as

$$\rho_{2} = \frac{n - x_{\leq \ell} - x_{\ell+1}}{n - x_{\leq \ell}} \cdot \left(\frac{q_{\ell+1}'}{q_{\ell+1}}\right)^{x_{\ell+1}} \cdot \left(\frac{1}{1-p}\right)^{n - x_{\leq \ell+1}+1} \\
\leqslant \left(\frac{p \cdot (x_{\ell} - 1)}{p \cdot x_{\ell} \cdot (1 - p \cdot x_{\ell})}\right)^{x_{\ell+1}} \cdot \left(\frac{1}{1-p}\right)^{n - x_{\leq \ell+1}+1} \\
\leqslant \left(1 + \frac{O(f(n))}{n^{1/3}}\right) \cdot \left(1 - \frac{1}{x_{\ell}}\right)^{x_{\ell+1}} \cdot \left(1 + \frac{p}{1-p}\right)^{n} \\
\leqslant \left(1 + \frac{O(f(n))}{n^{1/3}}\right) \cdot e^{-\frac{x_{\ell+1}}{x_{\ell}}} \cdot e^{np + O(np^{2})} \\
\leqslant \left(1 + \frac{O(f(n))}{n^{1/3}}\right) \cdot e^{np - np + O(\sqrt{f(n)\log(n)/x_{\ell}})} \leqslant 1 + \frac{1}{n^{\Omega(1)}}.$$

In the last inequality, we used that because \mathcal{W} holds, we have $x_{\ell+1} \leq x_{\ell} \cdot np + O(\sqrt{f(n)x_{\ell}\log n})$ and that $x_{\ell} \geq n^{\nu}$ for some constant $\nu > 0$. We can similarly lower bound ρ_2 as follows.

$$\begin{split} \rho_2 &\geq \frac{n-n^{1/3}}{n} \cdot \left(\frac{p(x_{\ell}-1)(1-p(x_{\ell}-1))}{p \cdot x_{\ell}}\right)^{x_{\ell+1}} \cdot \left(\frac{1}{1-p}\right)^{n-n^{1/3}} \\ &\geq \left(1 - \frac{O(f(n))}{n^{1/3}}\right) \cdot \left(1 - \frac{1}{x_{\ell}}\right)^{x_{\ell+1}} \cdot (1+p)^n \\ &\geq \left(1 - \frac{O(f(n))}{n^{1/3}}\right) \cdot e^{-\frac{x_{\ell+1}}{x_{\ell}}} \cdot \left(1 - \frac{1}{x_{\ell}^2}\right)^{x_{\ell+1}} \cdot e^{pn} \cdot (1-p^2)^n \\ &\geq \left(1 - \frac{O(f(n))}{n^{1/3}}\right) \cdot \left(1 - \frac{x_{\ell+1}}{x_{\ell}^2}\right) \cdot (1-p^2n) \cdot e^{np-np-O(\sqrt{np\log(n)/x_{\ell})}} \\ &\geq \left(1 - \frac{O(f(n))}{n^{1/3}}\right) \cdot \left(1 - \frac{O(f(n))}{x_{\ell}}\right) \cdot \left(1 - O\left(\frac{\sqrt{np\log(n)}}{\sqrt{x_{\ell}}}\right)\right) \\ &\geq 1 - \frac{1}{n^{\Omega(1)}}. \end{split}$$

In the third inequality, we use that for $y \in [-1, 1]$, it holds that $1 + y \ge e^y \cdot (1 - y^2)$. In the last inequality, we again used that $x_{\ell-1}$ and x_{ℓ} are both of size $\ge n^{\nu}$ for some constant $\nu > 0$.

Finally, let us look at the ratio between the probabilities $Pr(X_i = x_i | \mathcal{X}'_{< i})$ and $Pr(X_i = x_i | \mathcal{X}_{< i})$ for $i > \ell + 1$. We denote this ratio by $\rho_{3,i}$, and by using Equation (8), we can bound $\rho_{3,i}$ as follows.

$$1 - O\left(\frac{f(n)}{n^{2/3}}\right) \le \rho_{3,i} = \frac{n - x_{\le i} + 1}{n - x_{\le i} + 1} \cdot (1 - q_i) \le 1 + O\left(\frac{f(n)}{n^{2/3}}\right).$$

We therefore have

$$\frac{\Pr(\mathcal{X}'_{\leq h+T+1} | \mathcal{W}')}{\Pr(\mathcal{X}_{\leq h+T+1} | \mathcal{W}')} = \rho_1 \cdot \rho_2 \cdot \prod_{i=\ell+2}^{h+T+1} \rho_i = 1 \pm o(1).$$

Recall that $\mathcal{X}_{\leq h+T+1} = \mathcal{X}_{<\ell} \cap \{X_{\ell} = x_{\ell}\} \cap \mathcal{X}_{>\ell}$ and $\mathcal{X}'_{\leq h+T+1} = \mathcal{X}_{<\ell} \cap \{X_{\ell} = x_{\ell} - 1\} \cap \mathcal{X}_{>\ell}$. We therefore have

$$\Pr(\mathcal{X}_{\leq h+T+1} | \mathcal{W}') = \Pr(X_{\ell} = x_{\ell} | \mathcal{X}_{<\ell} \cap \mathcal{X}_{>\ell} \cap \mathcal{W}') \cdot \Pr(\mathcal{X}_{<\ell} \cap \mathcal{X}_{>\ell} | \mathcal{W}') \text{ and} \\ \Pr(\mathcal{X}_{\leq h+T+1} | \mathcal{W}') = \Pr(X_{\ell} = x_{\ell} - 1 | \mathcal{X}_{<\ell} \cap \mathcal{X}'_{>\ell} \cap \mathcal{W}') \cdot \Pr(\mathcal{X}_{<\ell} \cap \mathcal{X}'_{>\ell} | \mathcal{W}')$$

and thus

$$\frac{\Pr(X_{\ell} = x_{\ell} - 1 \mid \mathcal{X}_{<\ell} \cap \mathcal{X}'_{>\ell} \cap \mathcal{W}')}{\Pr(X_{\ell} = x_{\ell} \mid \mathcal{X}_{<\ell} \cap \mathcal{X}_{>\ell} \cap \mathcal{W}')} = \frac{\Pr(\mathcal{X}'_{\le h+T+1} \mid \mathcal{W}')}{\Pr(\mathcal{X}_{\le h+T+1} \mid \mathcal{W}')} = 1 \pm o(1).$$

However, this means that if \mathcal{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 $\Pr(X_{\ell} = x_{\ell}^* | \mathcal{X}_{<\ell} \cap \mathcal{X}_{>\ell} \cap \mathcal{W}')$, we have $\Pr(X_{\ell} = x_{\ell}^* | \mathcal{X}_{<\ell} \cap \mathcal{X}_{>\ell} \cap \mathcal{W}') \leq 1/2 + o(1)$. Thus, if \mathcal{W}' holds, the nodes in V_h exactly estimate X_{ℓ} with probability better than 1/2 + o(1). Because \mathcal{W}' holds with probability 1 - o(1), even if the nodes in V_h always succeed in case \mathcal{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 $n \geq n_0$ 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 $\nu > 0$, 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 2.

Theorem 7. 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 $\{1, \ldots, n\}$ as follows. Each color x is included in S_v independently with probability f(n)/n, where $f(n) \ge c \ln(n)$ for a sufficiently large constant c and $f(n) \le \text{poly} \log n$. Let G be the subgraph of K_n defined by all n nodes and the set of edges between nodes u and v with $S_u \cap S_v \ne \emptyset$. 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 $V = \{1, \ldots, n\}$ and the set of color $C = \{1, \ldots, n\}$. 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 6.

References

- [AA20] Noga Alon and Sepehr Assadi. Palette sparsification beyond $(\Delta + 1)$ vertex coloring. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM), volume 176 of LIPIcs, pages 6:1–6:22. LZI, 2020. 4, 5
- [ACK19] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta + 1)$ vertex coloring. In the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms

(SODA), pages 767–786, 2019. Full version at arXiv:1807.08886. 1, 2, 3, 4, 5, 6, 7, 11, 12, 16, 26, 28, 43, 44

- [AGG⁺19] 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 Symposium on Parallelism in Algorithms and Architectures*, pages 69–79, 2019. 1, 4
- [AGM12] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *SODA*, pages 459–467. SIAM, 2012. 3, 4
- $[AKM22] Sepehr Assadi, Pankaj Kumar, and Parth Mittal. Brooks' theorem in graph streams: a single-pass semi-streaming algorithm for <math>\Delta$ -coloring. In *STOC*, pages 234–247. ACM, 2022. 4, 5
- [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). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. 7
- [AKZ22] Sepehr Assadi, Gillat Kol, and Zhijun Zhang. Rounds vs communication tradeoffs for maximal independent sets. arXiv preprint arXiv:2209.09049, 2022. 4
- [AMS96] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In Gary L. Miller, editor, Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996, pages 20–29. ACM, 1996. 3
- [AW22] Sepehr Assadi and Chen Wang. Sublinear time and space algorithms for correlation clustering via sparse-dense decompositions. In Mark Braverman, editor, 13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 February 3, 2022, Berkeley, CA, USA, volume 215 of LIPIcs, pages 10:1–10:20. Schloss Dagstuhl Leibniz-Zentrum für Informatik, 2022. 11
- [Bar15] L. Barenboim. Deterministic $(\Delta + 1)$ -coloring in sublinear (in Δ) time in static, dynamic and faulty networks. In Proc. 34th ACM Symposium on Principles of Distributed Computing (PODC), pages 345–354, 2015. 4
- [BEG18] Leonid Barenboim, Michael Elkin, and Uri Goldenberg. Locally-Iterative Distributed $(\Delta + 1)$ -Coloring below Szegedy-Vishwanathan Barrier, and Applications to Self-Stabilization and to Restricted-Bandwidth Models. In the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), pages 437–446, 2018. 4
- [BEPS16] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *Journal of the ACM*, 63(3):20:1–20:45, 2016. 1, 3, 4, 5, 10, 18
- [BJK⁺02] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In José D. P. Rolim and Salil P. Vadhan, editors, Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings, volume 2483 of Lecture Notes in Computer Science, pages 1–10. Springer, 2002. 3

- [Bol98] Béla Bollobás. Random graphs. In *Modern graph theory*, pages 215–252. Springer, 1998. 5, 9, 31
- [CCF02] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan J. Eidenbenz, and Ricardo Conejo, editors, Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings, volume 2380 of Lecture Notes in Computer Science, pages 693–703. Springer, 2002. 3
- [CK10] Alejandro Cornejo and Fabian Kuhn. Deploying wireless networks with beeps. In International Symposium on Distributed Computing, pages 148–162. Springer, 2010. 4
- [CLP18] Yi-Jun Chang, Wenzheng Li, and Seth Pettie. An optimal distributed $(\Delta+1)$ -coloring algorithm? In the Proceedings of the ACM Symposium on Theory of Computing (STOC), pages 445–456, 2018. 5
- [CLP20] Yi-Jun Chang, Wenzheng Li, and Seth Pettie. Distributed $(\Delta + 1)$ -coloring via ultrafast graph shattering. SIAM Journal of Computing, 49(3):497–539, 2020. 1, 4, 5, 10, 11, 12
- [CM04] Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In Martin Farach-Colton, editor, LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings, volume 2976 of Lecture Notes in Computer Science, pages 29–38. Springer, 2004. 3
- [DDG⁺14] Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Amoebot – A new model for programmable matter. In Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures, pages 220–222, 2014. 4
- [Doe20] Benjamin Doerr. Probabilistic Tools for the Analysis of Randomized Optimization Heuristics, pages 1–87. Springer International Publishing, 2020. 41
- [DP09] Devdatt P. Dubhashi and Alessandro Panconesi. Concentration of Measure for the Analysis of Randomized Algorithms. Cambridge University Press, 2009. 42
- [EPS15] Michael Elkin, Seth Pettie, and Hsin-Hao Su. $(2\Delta 1)$ -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, pages 355–370, 2015. 5, 6, 7
- [EW13] Yuval Emek and Roger Wattenhofer. Stone age distributed computing. In Proceedings of the 2013 ACM symposium on Principles of distributed computing, pages 137–146, 2013. 4
- [FGH⁺23] Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. Coloring fast with broadcasts. To appear at SPAA'23, 2023. 4
- [FHK16] Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local Conflict Coloring. In the Proceedings of the Symposium on Foundations of Computer Science (FOCS), pages 625–634, 2016. 1

- [GG23] Mohsen Ghaffari and Christoph Grunau. Faster deterministic distributed MIS and approximate matching. In *ACM Symposium on Theory of Computing (STOC)*, pages to appear, arXiv:2303.16043, 2023. 1, 4
- [GGR21] Mohsen Ghaffari, Christoph Grunau, and Václav Rozhoň. Improved deterministic network decomposition. In the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), 2021. 1
- [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, pages 202–219. SIAM, 2016. 1, 3, 47
- [GK13] Mohsen Ghaffari and Fabian Kuhn. Distributed minimum cut approximation. In International Symposium on Distributed Computing, pages 1–15. Springer, 2013. 1, 3, 47
- [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, pages 1009–1020. IEEE, 2021. 1, 2, 4, 18
- [GKK⁺15] Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. Near-optimal distributed maximum flow. In Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, pages 81–90, 2015. 1, 3, 47
- [GMT15] Sudipto Guha, Andrew McGregor, and David Tench. Vertex and hyperedge connectivity in dynamic graph streams. In Tova Milo and Diego Calvanese, editors, Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015, pages 241–247. ACM, 2015. 3
- [GZ22] Mohsen Ghaffari and Goran Zuzic. Universally-optimal distributed exact mincut. In ACM Symposium on Principles of Distributed Computing, pages to appear, arXiv:2205.14967, 2022. 1, 3, 47
- [HK73] John E Hopcroft and Richard M Karp. An n⁵/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973. 7
- [HKMT21] Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Tigran Tonoyan. Efficient randomized distributed coloring in CONGEST. In the Proceedings of the ACM Symposium on Theory of Computing (STOC), pages 1180–1193. ACM, 2021. Full version at CoRR abs/2105.04700. 1, 4, 6, 11, 12, 45
- [HKNT22] Magnús M. Halldórsson, Fabian Kuhn, Alexandre Nolin, and Tigran Tonoyan. Nearoptimal distributed degree+1 coloring. In STOC, pages 450–463. ACM, 2022. 1, 2, 4, 5, 10, 11, 12, 42
- [HNT22] Magnús M. Halldórsson, Alexandre Nolin, and Tigran Tonoyan. Overcoming congestion in distributed coloring. In the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), pages 26–36. ACM, 2022. 1, 2, 12, 13, 43
- [HSS16] S. G. Harris, J. Schneider, and H.-H. Su. Distributed $(\Delta + 1)$ -coloring in sublogarithmic rounds. In Proc. 48th Symp. on the Theory of Computing (STOC), 2016. 5

- [HSS18] David G. Harris, Johannes Schneider, and Hsin-Hao Su. Distributed $(\Delta + 1)$ -coloring in sublogarithmic rounds. Journal of the ACM, 65:19:1–19:21, 2018. 1, 4, 5, 11, 12
- [Joh99] Öjvind Johansson. Simple distributed $\Delta + 1$ -coloring of graphs. Inf. Process. Lett., 70(5):229–232, 1999. 1, 3, 4, 5, 10
- [KLM⁺14] 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, pages 561–570. IEEE Computer Society, 2014. 3
- [Lin92] Nati Linial. Locality in distributed graph algorithms. SIAM Journal on Computing, 21(1):193–201, 1992. 1, 3, 4
- [Mot94] Rajeev Motwani. Average-case analysis of algorithms for matchings and related problems. Journal of the ACM (JACM), 41(6):1329–1356, 1994. 7
- [MT20] Yannic Maus and Tigran Tonoyan. Local conflict coloring revisited: Linial for lists. In the Proceedings of the International Symposium on Distributed Computing (DISC), pages 16:1–16:18, 2020. 4
- [Pel00] David Peleg. Distributed Computing: A Locality-Sensitive Approach. SIAM, 2000. 1
- [PS97] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. SIAM J. Comput., 26(2):350–368, 1997. 1
- [Ree98] Bruce A. Reed. ω , Δ , and χ . J. Graph Theory, 27(4):177–212, 1998. 5, 11
- [RGH⁺22] Václav Rozhoň, Christoph Grunau, Bernhard Haeupler, Goran Zuzic, and Jason Li. Undirected $(1 + \varepsilon)$ -shortest paths via minor-aggregates: near-optimal deterministic parallel and distributed algorithms. In *Proceedings of the 54th Annual ACM SIGACT* Symposium on Theory of Computing, pages 478–487, 2022. 1, 3, 47
- [SW10] Johannes Schneider and Roger Wattenhofer. A new technique for distributed symmetry breaking. In the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), pages 257–266. ACM, 2010. 1, 4, 5, 6, 10
- [Tal95] Michel Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. Publications Mathématiques de l'Institut des Hautes Etudes Scientifiques, 81(1):73-205, 1995. 42

A Concentration Bounds

Some useful inequalities. We use the following classic inequalities:

Lemma A.1 ([Doe20]). For $x \in [0,1]$ and y > 0, we have

$$1 - x \leqslant e^{-x} \leqslant 1 - \frac{x}{2}$$
 and $(1 - x)^y \leqslant \frac{1}{1 + xy}$. (11)

Chernoff bound with domination. The classical version of the Chernoff bound shows concentration for sum of binary random variables and assumes independence between each variable. We use a more general form allowing for some dependencies and non-binary variables.

Lemma A.2 (Martingales). Let $\{X_i\}_{i=1}^r$ be random variables distributed in [0,1], and $X = \sum_i X_i$. Suppose that for all $i \in [r]$ and $(x_1, \ldots, x_{i-1}) \in \{0,1\}^{i-1}$ with $\Pr[X_1 = x_1, \ldots, X_r = x_{i-1}] > 0$, $\Pr[X_i = 1 \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq q_i \leq 1$, then for any $\delta > 0$,

$$\Pr\left[X \ge (1+\delta)\sum_{i=1}^{r} q_i\right] \le \exp\left(-\frac{\min(\delta,\delta^2)}{3}\sum_{i=1}^{r} q_i\right).$$
(12)

Suppose instead that $\Pr[X_i = 1 \mid X_1 = x_1, \dots, X_{i-1} = x_{i-1}] \ge q_i, q_i \in (0, 1)$ holds for i, x_1, \dots, x_{i-1} over the same ranges, then for any $\delta \in [0, 1]$,

$$\Pr\left[X \leqslant (1-\delta)\sum_{i=1}^{r} q_i\right] \leqslant \exp\left(-\frac{\delta^2}{2}\sum_{i=1}^{r} q_i\right)$$
(13)

Talagrand inequality. A function $f(x_1, \ldots, x_n)$ is *c*-Lipschitz iff changing any single x_i affects the value of f by at most c, and f is *r*-certifiable iff whenever $f(x_1, \ldots, x_n) \ge s$ for some value s, there exist $r \cdot s$ inputs $x_{i_1}, \ldots, x_{i_{r\cdot s}}$ such that knowing the values of these inputs certifies $f \ge s$ (i.e., $f \ge s$ whatever the values of x_i for $i \notin \{i_1, \ldots, i_{r \cdot s}\}$).

Lemma A.3 (Talagrand's inequality [Tal95, DP09]). Let $\{X_i\}_{i=1}^n$ be n independent random variables and $f(X_1, \ldots, X_n)$ be a c-Lipschitz r-certifiable function; then for $t \ge 1$,

$$\Pr\left[|f - \mathbb{E}[f]| > t + 30c\sqrt{r \cdot \mathbb{E}[f]}\right] \leq 4 \cdot \exp\left(-\frac{t^2}{8c^2r \mathbb{E}[f]}\right)$$

In the next lemma, \mathbb{I}_X denotes the indicator random variable of an event X.

Lemma A.4 ([HKNT22]). Let $\{X_i\}_{i=1}^n$ be *n* independent random variables. Let $\{A_j\}_{j=1}^k$ and $\{B_j\}_{j=1}^k$ be two families of events that are functions of the X_i 's. Let $f = \sum_{j \in [k]} \mathbb{I}_{A_j}, g = \sum_{j \in [k]} \mathbb{I}_{A_j \cap \overline{B}_j},$ and h = f - g be such that f and g are c-Lipschitz and r-certifiable w.r.t. the X_i 's, and $\mathbb{E}[h] \ge \alpha \mathbb{E}[f]$ for some constant $\alpha \in (0, 1)$. Let $\delta \in (0, 1)$. Then for $\mathbb{E}[h]$ large enough:

$$\Pr[|h - \mathbb{E}[h]| > \delta \mathbb{E}[h]] \leq \exp(-\Omega(\mathbb{E}[h])).$$

B Omitted Proofs

B.1 Computing the Almost-Clique Decomposition

In this section, we show the following lemma:

Lemma 4.4. There is a $O(\log \Delta)$ -round algorithm computing an ε -almost-clique decomposition. It only broadcasts $O(\log n/\varepsilon^4)$ -bit messages and samples $O(\log n/\varepsilon^2)$ edges per node.

Algorithm 10. Algorithm computing a ε -almost-clique decomposition. Parameters. Define

 $\delta = \frac{\varepsilon}{12}, \qquad \lambda = \frac{16\Delta}{\delta} \qquad \text{and} \qquad \sigma = \frac{384\beta}{\delta^4}.$

When sparsifying the input graph. Each node v samples a value $r(v) \in [\lambda]$ uniformly at random. Each node v then computes

- the set F(v) containing all values r(u) for neighbors $u \in N(v)$ such that $r(v) \leq \sigma$.
- 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 $E_s(v)$.
 - If $|F(v) \cap F(u)| \ge (1-\delta)\Delta\sigma/\lambda$, then u and v are friends.
- 2. By counting its number of friends in $E_s(v)$, v learns if it is popular.

Definition 11 (Friendly edges). For any $\delta \in (0, 1)$, we say that nodes u and v are *friends* if they are connected, i.e., $uv \in E$, and share a $(1 - \delta)$ -fraction of their neighborhood, i.e., $|N(u) \cap N(v)| \ge (1 - \delta)\Delta$.

To detect friendly edges, the approach of [ACK19] was to sample nodes with probability $O(\frac{\log n}{\delta^2 \Delta})$ and, for each sampled edge uv, compare the set of nodes sampled in N(v) to that of N(u). This approach requires nodes to communicate $O(\log^2 n)$ bits with their neighbors ($O(\log n)$ -bits identifiers for $O(\log n)$ sampled neighbors); hence, it exceeds the bandwidth requirements of our model.

In recent work, [HNT22] proposed a CONGEST algorithm for solving this task in O(1) rounds. They devise an algorithm ([HNT22, Algorithm 1]) using families of pseudo-random hash functions to estimate up to $\delta\Delta$ precision the similarities of two $\Theta(\Delta)$ -sized sets. They observed it could be used to compute ACD in a rather straightforward way by using this primitive to compare neighborhoods. The main obstacle to implement this algorithm in our model are memory constraints: their families of hash function are non-constructive and of size poly(n).

We note, however, that for this specific use, we can sample a truly random function. To sample a random function r mapping nodes to values in $[\lambda]$, it is enough if each node v samples a value $r(v) \in [\lambda]$ independently. By [HNT22, Claim 1], the induced random function has few enough collisions for nodes to estimate the size of their shared neighborhood with sufficient accuracy with high probability. Furthermore, to do so, nodes need only to know the hash value of their neighbors. The parameters of Algorithm 10 are set to match the ones of [HNT22, Algorithm 1] with up to $(\delta/2)\Delta$ error.

Lemma B.1 (Detecting Friendly Edges, [HNT22, Claim 1 + Lemma 2]). Let $\delta \in (0, 1/10)$. For every pair of adjacent nodes uv, with high probability, we have that

- if u and v are δ -friends, we have $|F(v) \cap F(u)| \ge (1 1.5\delta)\Delta\sigma/\lambda$; and
- if u and v are not 2δ -friends, we have $|F(v) \cap F(u)| < (1 1.5\delta)\Delta\sigma/\lambda$.

The other primitive required to compute the almost-clique decomposition is for distinguishing between δ -popular nodes and those that are not 2δ -popular.

Definition 12 (Popular Nodes). For any $\delta \in (0, 1/10)$, we say u is δ -popular if it has $(1 - \delta)\Delta$ friendly edges.

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 A.2) 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 B.2 (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 $(1 1.5\delta)\Delta p \ \delta$ -friendly edges in $E_s(u)$;
- if u is not 2δ -popular, it samples fewer than $(1 1.5\delta)\Delta p \ 2\delta$ -friendly edges in $E_s(u)$.

Lemmas B.1 and B.2 are sufficient to find a δ -almost-clique decomposition.

Lemma B.3 ([ACK19]). Let H be the subgraph of G with 2δ -popular nodes. Let C_1, \ldots, C_t be the connected components of H with at least one $\delta/2$ -popular node and $V_{\text{sparse}} = V \setminus \bigcup_{i \in [t]} C_i$. This decomposition is a 12 δ -almost-clique decomposition.

We are now ready to prove Lemma 4.4.

Proof of Lemma 4.4. Let $V_{\text{sparse}}, C_1, \ldots, C_t$ be the decomposition described in Lemma B.3. Consider a cluster C_i for some $i \in [t]$. By Lemma B.2, nodes can tell if they are $\delta/2$ -popular. Moreover, the subgraph $\tilde{H}(C)$ of C_i , consisting of the sampled edges in $E_s(\cdot)$, is a random graph where edges are sampled with probability $O(\log n/\Delta)$. This means that $\tilde{H}(C)$ has a constant rate vertex expansion (see Lemma 4.5 for a proof of a similar fact). Therefore, in $O(\log \Delta)$ rounds, every node v in the connected component C_i knows it belongs to an almost-clique, as well as the identifier of the $\delta/2$ -popular node in C_i with minimal ID (used as identifier for the clique) and which edges in $E_s(v)$ are connecting it to C_i .

B.2 Preconditioning Almost-Clique

Theorem 3. 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 $\Delta/\eta \ge K \log n$ for a large enough constant K > 0. There exists an algorithm computing a partial coloring of G where all uncolored nodes are partitioned in almost-cliques C_1, \ldots, C_t for some t such that for any $i \in [t]$, almost-clique C_i is such that:

- 1. $|C_i| \leq (1+\varepsilon)\Delta;$
- 2. $|N(v) \cap C_i| \ge (1 \varepsilon)\Delta$ for all nodes $v \in C_i$;
- 3. $|N(v) \cap \bigcup_{j \neq i} C_j| \leq e_{\max} = \Delta/\eta$

Furthermore, the algorithm runs in $O(\log \Delta + \log \eta)$ rounds, uses $O(\eta \log n)$ colors from lists L_1 , and samples $O(\eta \log n)$ edges per node.

Assume we computed an ε' -almost-clique decomposition using Algorithm 10 for $\varepsilon' = \varepsilon/3$. In this section, we use this decomposition to compute the partial coloring described in Theorem 3.

Sparse nodes receive slack after a single randomized color trial. Intuitively, this happens because each non-edge in the neighborhood of a ζ -sparse node v has both its endpoints colored the same with probability $\Omega(1/\Delta)$. Since it has $\Delta\zeta$ such non-edges in E(N(v)) (see Definition 3), it receives $\Omega(\zeta)$ slack in expectation. Formally, this gives the following lemma. **Lemma B.4** ([HKMT21, Lemma 6.3]). Let v be a ζ -sparse node. After a random fraction of its neighbors try colors, it has slack $\Omega(\zeta)$ with probability $1 - e^{-\Omega(\zeta)}$. Furthermore, if v is a dense node, it receives slack $\Omega(e_v)$ with probability $1 - e^{-\Omega(e_v)}$.

In Theorem 3, we want to get rid of high external degree nodes. By Lemma B.4, if a node has a high external degree, it should also have a lot of slack. Algorithm 11 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 B.5. There is an algorithm partitioning the dense nodes into two classes: extroverted nodes of external degree at most Δ/η , and introverted nodes of external degree at least $\Delta/(2\eta)$. The algorithm samples $O(\eta \log n)$ edges per node.

Proof. Let $e_{\text{max}} = \Delta/\eta$. During the streaming phase, 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.

- Consider a node with external degree at most $e_{\max}/2$. In expectation, it samples $pe_{\max}/2 \leq \beta/2$ 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 nodes, i.e., with external degree at least e_{max} . In expectation, it samples at least $pe_{\text{max}} \ge \beta$ 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 $e_{\text{max}}/2$ and e_{max} can be arbitrarily classified as introvert or extrovert.

Definition 13 (Extrovert/Introvert). An almost-clique is *extrovert* if it has more than $2\varepsilon'\Delta$ extroverted nodes, and *introvert* otherwise.

Algorithm 11. The algorithm preconditioning almost-cliques. Input: an ε' -almost-clique decomposition $V_{\text{sparse}}, C_1, \ldots, C_t$ for some t.

- 1. In each clique C_i , let $W_i \subseteq C_i$ be its set of extroverted nodes. Each 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 C_i is extrovert.
- 2. (Generate Slack) With probability 1/20, sparse nodes and dense nodes from extroverted cliques $V_{\text{sparse}} \cup \bigcup_{i \in J} C_i$ independently try a random color.
- 3. Let $V' = V_{\text{sparse}} \cup \bigcup_{i \notin J} W_i \cup \bigcup_{i \in J} (C_i \setminus W_i)$ be the set containing sparse nodes, extroverted nodes from introverted cliques and introvertednodes from extroverted 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 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 cliques using SLACKCOLOR.

Proof of Theorem 3. 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 Lemma B.4). Let $J \subseteq [t]$ the set of extroverted cliques. In Step 3, we color nodes of V' where

$$V' = V_{\text{sparse}} \cup \bigcup_{i \notin J} W_i \cup \bigcup_{i \in J} (C_i \setminus W_i) \;.$$

Dense nodes of V' receive slack from their inactive neighbors in $V \setminus V'$.

- An extroverted nodes $v \in W_i$ in some introverted clique C_i with $i \notin J$ has $|N(v) \cap (C \setminus W_i)| \ge (1 3\varepsilon')\Delta$ introverted neighbors in C_i . Note that none of them was colored in Step 2.
- A introverted node $v \in C$ in an extroverted clique C_i with $i \in J$ has at least $|N(v) \cap W_i| \ge (2\varepsilon' \varepsilon')\Delta = \varepsilon'\Delta$ extroverted neighbors in C_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 $\Omega(\Delta)$ for a small enough universal constant. Hence, by Lemma 3.1, we can color all nodes in V' in $O(\log \Delta)$ rounds and $O(\log n)$ fresh colors with high probability.

After Step 3, the only extroverted nodes to remain uncolored are in extroverted cliques. We now explain how we color these nodes. Nodes have $\Omega(\Delta/\eta)$ slack. By an argument similar to Lemma 5.3, w.h.p., we reduce the degree of each node by a constant factor. After $O(\log \eta)$ rounds, each node has uncolored degree $O(\Delta/\eta)$. It samples $O(\eta \log \log n)$ colors. Nodes now have slack proportional to their degree and can be colored by SLACKCOLOR in $O(\log \Delta)$ rounds and using $O(\eta \log n)$ colors.

We now prove that our coloring verifies the properties of Theorem 3. The crux is that the only uncolored nodes remaining are introverted nodes in introverted almost-cliques. For each introverted almost-clique C in the ε' -almost-clique decomposition, we get an ε -almost-clique C' with the claimed properties by simply removing colored nodes. This is because C' is an ε' -almost-cliques from which we removed at most $2\varepsilon'\Delta$ extroverted nodes. Hence, the upper bound $|C'| \leq (1 + \varepsilon')\Delta \leq (1 + \varepsilon)\Delta$ trivially holds (recall $\varepsilon' = \varepsilon/3$) and for all $v \in C'$, we have $|N(v) \cap C'| \geq (1 - 3\varepsilon')\Delta = (1 - \varepsilon)\Delta$. Furthermore, all nodes of C' are introverted, therefore they are connected to at most Δ/η nodes in other cliques. Note however that they can be connected to $\varepsilon\Delta$ colored nodes (as they include sparse nodes and extroverted nodes from C).

B.3 Analysis of RandomPush

Lemma 6.7. Let $x \leq \beta^3$ messages of $O(\log n)$ bits each known by exactly one node in the sparsified almost-clique \tilde{C} . 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 $\overline{S_i} = C \setminus S_i$.

Initially, $|S_1| \ge 1$. Each node has degree $\Theta(\beta^4)$, w.h.p., by Claim 4.2. Thus, nodes forward the message to $\Omega(\beta)$ of its neighbors, so $|S_2| \ge \beta$, w.h.p. We now show that S_i grows geometrically while $|S_i| \le 3\Delta/4$, then afterwards $\overline{S_i}$ decreases geometrically.

By Lemma 4.5, while $|S_i| \leq 3\Delta/4$, there are at least $|S_i|\beta^4/40$ edges between S_i and $\overline{S_i}$. For an uninformed node v in $\overline{S_i}$, let $d_v^{S_i} = |N_{\widetilde{C}}(v) \cap S_i|$ be its number of informed neighbors. Letting X_v be the event that v learns the message in this iteration, we have that

$$\Pr[X_v] = 1 - \left(1 - \frac{1}{x}\right)^{d_v^{S_i}} \ge 1 - \frac{1}{1 + d_v^{S_i}/x} = \frac{d_v^S}{x + d_v^{S_i}}$$

where we used Lemma A.1. Hence, the expected number of nodes that learn the message is at least

$$\sum_{v \in \overline{S_i}} \frac{d_v^{S_i}}{x + d_v^{S_i}} \ge \frac{\widetilde{\Delta}}{x + \widetilde{\Delta}} \cdot \frac{|S_i|\beta^4}{8\widetilde{\Delta}} \ge \Omega(|S_i|) ,$$

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 $\frac{|\overline{S_i}|\beta^4/40}{\widetilde{\Delta}}$ nodes satisfying $d_v^S = \widetilde{\Delta}$ and the rest satisfying $d_v^S = 0$.

Since the X_v are independent, by Lemma A.2 (Chernoff bound) it holds w.h.p. that $|S_{i+1}| \ge (1 + \Omega(1))|S_i|$ while $|S_i| \in [\beta, 3\Delta/4]$. Therefore, after $i \in \Theta(\log \Delta)$ iterations, $|S_i| \le 3\Delta/4$.

The rest of the argument is similar. The nodes in $\overline{S_i}$ have at least $|S_i|\beta^4/40$ edges with S_i . In expectation, $\Theta(|\overline{S_i}|)$ of them get colored in each iteration in expectation, and this holds w.h.p. while $|\overline{S_i}| \ge \beta$. When $|\overline{S_i}|$ drops below $O(\beta^3)$, each node in $\overline{S_i}$ is adjacent to $\Omega(\beta^4)$ nodes in S_i . Therefore, it receives the message $\Omega(\beta)$ times in expectation, and thus receives it w.h.p.

C Corollaries for Other Models

C.1 Coloring in Distributed Streaming

Definition 14 (Local Streaming Model). In the LocalStream model, there are n nodes with unique $O(\log n)$ -bit identifiers and $p(n) = poly(\log n)$ 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^{-6} .
- (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 poly log n messages per rounds.

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 C.1. There exists a LocalStream algorithm using $O(\log^4 n)$ memory per node and $O(\log^2 \Delta)$ rounds of communication.

C.2 Coloring in the Cluster Graph Model

We first define the cluster graph model (a variant appears in [GKK⁺15] and similar concepts appear in other places in the literature, see e.g., [RGH⁺22, GH16, GK13, GZ22]). Then, we state our result.

Definition 15 (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. Definite 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:

⁶An alternative would be to supply the nodes with shared randomness. Then the ID of the other node would suffice to learn its random bits.

- (Intra-cluster broadcast) Each cluster center starts with a 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 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 $poly(\log n)$ -bit message and, in each cluster, we deliver a $poly(\log n)$ -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 $poly(\log n)$ 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 $poly(\log n)$ -bit messages.

Theorem 8. There is a distributed randomized algorithm that computes a Δ + 1-coloring in 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 1. We just need to discuss how the cluster graph computes and simulates the corresponding sparsified graph.

Each cluster center samples the 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 C that notices a neighboring cluster C' that sampled a common color remembers the cluster identifier of 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 poly(log n) neighboring clusters after the sparsification, this can be done as a poly(log n)-bit aggregation.

In the course of this process, we could also elect for each pair of neighboring clusters C and C' in this sparsified graph one physical edge from node $v \in C$ to a node $u \in C'$. For instance, that can be the edge (v, u) with the highest ID tuple. Again, this fits easily as a poly $(\log n)$ -bit aggregation.

At this point, each cluster center knows all its 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 1.

C.3 Coloring in the Node Capacitated Clique

We show, in fact, that any 1-pass LocalStream algorithm with $poly \log n$ memory and bandwidth can be turned into a poly log n rounds NCC algorithm.

Theorem 9. Let \mathcal{A} be a randomized LocalStream algorithm using one streaming pass and Tcommunication rounds. If it has bandwidth B and communication with at most D different neighbors
within a round, then there is an algorithm emulating \mathcal{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 LocalStream. 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 LocalStream 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 C.2. If nodes know the poly log n random bits of their neighbors, then emulating \mathcal{A} requires $O\left(\frac{T \cdot (BD)}{\log n}\right)$.

The only information missing to nodes in order to run the LocalStream 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 poly log n bits binary strings, we write $\mathcal{A}[S]$ for the algorithm \mathcal{A} where each node u has the string S(u) as random bits.

Lemma C.3. For any fixed n, there is a family S of poly(n) functions mapping nodes to initial states such that for any n-node graph input, if we run A on a random function in S, the streaming is correct with high probability.

Proof of Lemma C.3. Fix a *n*-node graph G. Sample t functions S_1, \ldots, S_t assigning poly log *n*-bits binary string to nodes.

Since \mathcal{A} is correct with high probability, it means that on a random S_i , algorithm $\mathcal{A}[S_i]$ fails with probability at most 1/n. For a fixed G, call X_i the random variable equal to one iff algorithm $\mathcal{A}[S_i]$ fails on G with probability more than 1/n. In expectation, the number of bad assignments is $\mathbb{E}\left[\sum_{i \in [t]} X_u\right] \leq t/n$. Samples are independent; hence, by Chernoff, we get

$$\Pr\left[\sum_{i\in[t]} X_i > \frac{2t}{n}\right] \leqslant \exp\left(-\frac{2t}{3n}\right).$$
(14)

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. (14) is strictly less than 1; hence, there is a family $S = \{S_1, \ldots, S_t\}$ such that the probability that $\mathcal{A}[S_i]$ fails for a random $i \in [t]$ is at most 2/n for all n-nodes graphs.

Proof of Theorem 9. For a fixed n-sized network. Nodes can locally compute the family S described in Lemma C.3 (recall there is not memory or local time constraints on nodes in the NCC model). The node of minimum ID then sample a random index $i \in [|S|]$ and broadcast it. Since $|S| \leq \text{poly}(n)$, 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 \mathcal{A} in $O(TBD/\log n)$ rounds. By Lemma C.3, it fails with probability $1/\operatorname{poly}(n)$.