Distributed Lovász Local Lemma under Bandwidth Limitations

Magnús M. Halldórsson · mmh@ru.is · Reykjavik University, Iceland Yannic Maus¹ · yannic.maus@ist.tugraz.at · TU Graz, Austria Saku Peltonen · saku.peltonen@gmail.com · Aalto University, Finland

Abstract

The constructive Lovász Local Lemma has become a central tool for designing efficient distributed algorithms. While it has been extensively studied in the classic LOCAL model that uses unlimited bandwidth, much less is known in the bandwidth-restricted CONGEST model.

In this paper, we present bandwidth- and time-efficient algorithms for various subclasses of LLL problems, including a large class of subgraph sampling problems that are naturally formulated as LLLs. Lastly, we use our LLLs to design efficient CONGEST algorithms for coloring sparse and triangle-free graphs with few colors. These coloring algorithms are exponentially faster than previous LOCAL model algorithms.

¹Supported by the Austrian Science Fund (FWF), Grant P36280-N.

Contents

| 1 | Introduction | | 1 |
|----------|--|--------------------|--|
| | 1.1 Our Contributions: LLL solvers | | 3 |
| | 1.1.1 Disjoint variable set LLLs | | 3 |
| | 1.1.2 Binary LLLs with low risk | | 4 |
| | 1.2 Our Contribution: Coloring Sparse and Triangle-Free Graphs | | 4 |
| | 1.3 Further Related Work | | 5 |
| | 1.4 Outline of the rest of the paper | | 6 |
| 2 | Distributed Lovász Local Lemma (Definitions) | | 6 |
| | 2.1 Constructive Lovász Local Lemma (LLL) | | 6 |
| | 2.2 Constructive Distributed Lovász Local Lemma | | 6 |
| | 2.3 Simulatable Distributed Lovász Local Lemma (CONGEST) $\ldots \ldots \ldots$ | | 7 |
| 3 | Technical Overview & Technical Contributions | | 8 |
| 0 | 3.1 Disjoint Variable Set LLLs | | 9 |
| | 3.2 Binary LLLs with low risk | | 9 |
| | 3.3 Post-shattering in CONGEST | | 13 |
| | 3.4 Coloring Sparse Graphs | | 13 |
| 4 | Binary LLLs with low Risk | | 14 |
| 5 | Disjoint Variable Set LLLs | | 18 |
| 6 | Efficient Post-shattering in CONGEST | | 21 |
| | 6.1 Network Decomposition | | 22 |
| | 6.2 The LLL algorithm of Chung, Pettie, Su | | 22 |
| | 6.3 Efficient Post-shattering in CONGEST (details) | | 23 |
| 7 | Applications and Bounding Risks | | 25 |
| | 7.1 Example of Disjoint Variable Set LLL: Slack Generation | | 26 |
| | 7.2 Techniques to Bound Risk | | 27 |
| | 7.3 Example IIIs with Low Rick | | |
| | 1.5 Example LLLS with Low Risk | | 29 |
| 8 | Applications II: Coloring Sparse Graphs and Slack Generation | | 29 31 |
| 8 | Applications II: Coloring Sparse Graphs and Slack Generation 8.1 Degree+1 List Coloring (d1LC) | | 29 31 31 |
| 8 | Applications II: Coloring Sparse Graphs and Slack Generation 8.1 Degree+1 List Coloring (d1LC) 8.2 Sparsity-preserving Degree Reduction | · · · · | 29 31 31 31 |
| 8 | Applications II: Coloring Sparse Graphs and Slack Generation 8.1 Degree+1 List Coloring (d1LC) 8.2 Sparsity-preserving Degree Reduction 8.3 Slack Generation for Sparse Nodes | · · · · | 29 31 31 31 33 |
| 8 | Applications II: Coloring Sparse Graphs and Slack Generation 8.1 Degree+1 List Coloring (d1LC) 8.2 Sparsity-preserving Degree Reduction 8.3 Slack Generation for Sparse Nodes 8.4 Coloring Sparse Graphs | · · · · · · · · | 29 31 31 31 33 36 |
| 8 | Applications II: Coloring Sparse Graphs and Slack Generation 8.1 Degree+1 List Coloring (d1LC) 8.2 Sparsity-preserving Degree Reduction 8.3 Slack Generation for Sparse Nodes 8.4 Coloring Sparse Graphs 8.5 Coloring Triangle-free Graphs | · · · · | 29 31 31 33 36 37 |

1 Introduction

The Lovász Local Lemma (LLL) is a powerful probabilistic tool that provides conditions under which many mildly locally dependent "bad events" defined over some random variables can simultaneously be avoided. In its computational version, one aims at also computing an assignment of the random variables avoiding all these events. Due to its local nature, it has become an important cornerstone for distributed computation, e.g., [BFH⁺16a, CP19, Bra19, CPS17, MU21, FG17, RG20, GHK18, Dav23]. For example, it plays a key role in the complexity theory of local graph problems, as it is complete for sublogarithmic computation [CP19, MU21], in the sense that any local graph problem (formally, any locally checkable labeling problem [NS95]) that admits a $o(\log n)$ -round algorithm can be solved in the same asymptotic runtime as LLL. Other examples are its usage as a tool for splitting graphs into smaller subgraphs while satisfying certain constraints for divide-and-conquer approaches [Dav23], or to solve concrete problems such as computing edge colorings with few colors [CHL⁺20, HMN22, Dav23]. It has also been imperative for developing the award-winning round elimination lower bound technique [BFH⁺16a, Bra19, BBH⁺21].

Classically, research on local distributed graph problems has a strong focus on problems that decompose nicely and are trivially solvable sequentially by greedy algorithms, such as finding maximal independent sets. For many other problems, especially those that go beyond the greedy regime such as many partitioning problems or coloring problems with few colors, easy randomized algorithms hold for high-degree graphs, but low-to-medium ranged degrees prove to be challenging. Combinatorially, LLL is the go-to method for such problems, and efficient LLL algorithms allow for many non-trivial results to immediately carry over [CPS17, HMN22, MU21].

The breakthrough Moser-Tardos algorithm and a more efficient distributed implementation by Chang, Pettie, and Su yield logarithmic-time distributed algorithms [MT10, CPS17]. After the publication of a seminal lower bound of $\Omega(\log \log n)$ -rounds [BFH⁺16a], the quest for understanding under which circumstances sublogarithmic time, optimally poly log log n time, LLL algorithms exist has begun [FG17, CP19, GHK18, Dav23]. This algorithmic success has almost exclusively been in the bandwidth-unrestricted setting, with progress on bandwidth-restricted algorithms being significantly more limited.

In this paper, we provide bandwidth- and time-efficient distributed algorithms for important subclasses of LLL problems and exemplify their usefulness via several applications mainly in the domains of subgraph sampling and coloring sparse graphs with few colors.

To explain the challenges that occur in developing bandwidth-efficient LLL algorithms we begin with the necessary background.

Distributed Lovász Local Lemma (LLL) An instance $\mathcal{L} = (\mathcal{V}, \mathcal{B})$ of the distributed Lovász Local Lemma (LLL) is given by a set of independent random variables \mathcal{V} and a family of "bad" events \mathcal{B} over these variables. The Lovász Local Lemma [EL74] states (in its basic form) that there exists an assignment to the variables that avoids all bad events as long as an appropriate relationship holds between the probability of each bad event and the number of events that any given event depends on. This relationship is called the LLL criterion, with many algorithms assuming stronger criteria than the one required for the existence of a solution. The dependency graph \mathcal{H} of an LLL has a node for each bad event in \mathcal{B} with two bad event nodes adjacent if they share a variable.

LOCAL and CONGEST model [Lin92, Pel00]. In the LOCAL model of distributed computing a communication network is abstracted as an *n*-node graph G = (V, E) of maximum degree Δ .

Nodes serve as computing entities and edges represent communication links. Nodes communicate with neighbors in synchronous rounds, where in each round a node can perform arbitrary local computations and send one message of *unbounded size* over each incident edge. The objective is to solve the problem at hand in the fewest rounds, e.g., with each node outputting its own color in a coloring problem. In the *distributed LLL* in the LOCAL model one generally assumes that the communication network is identical with the dependency graph \mathcal{H} . This is motivated by the fact that in most applications of LLL the communication network and the dependency graph are in close resemblance and communication in \mathcal{H} can be simulated in the original communication network within a constant factor overhead in the round complexity. In the LOCAL model, there are poly log log n LLL algorithms for certain special types of LLLs, e.g., for LLLs with small maximum degree $\Delta \leq$ poly log log n [FG17, RG20], for LLLs with very strong LLL criteria [Dav23], or for LLLs satisfying some additional technical properties [GHK18].

The CONGEST model is identical to LOCAL, except for the important difference that messages are restricted in size; each message can only contain $O(\log n)$ bits, which fits only a constant number of node identifiers. As a result, one has to be much more careful when modeling distributed LLL instances and precise on which event and variable are simulated by which node of the communication network. We illustrate these challenges with a key example central to this paper.

Example LLL (Slack Generation). One prime application in our work is the slack generation LLL for graph coloring. The *slack* of a node in a partial coloring of a graph is the number of colors that are available to it minus its uncolored degree in the graph. If every node has (positive) slack, the respective coloring problem can be solved via a sequential greedy algorithm and there are also poly log log *n*-round distributed algorithms, even under the presence of bandwidth restrictions [HKNT22, HNT22]. For the rest of this example, assume a sparse Δ -regular graph G = (V, E), i.e., a graph in which every node $v \in V$ has many non-edges in its neighborhood, that is, G[N(v)] is far from being a clique. Such graphs can be colored with $\ll \Delta$ colors, but this cannot be done greedily as nodes may not have slack. We show that coloring some of the nodes such that every remaining node has slack (forming an easy greedily-solvable residual problem) can be modeled as an LLL (see Algorithm 2 for the random process and Lemmas 7.1 and 8.6 for the formal statements):

Each node is activated with constant probability and each activated node picks a random candidate color. Nodes with no neighbor with the same candidate color keep their color, and otherwise relinquish it.

A node v gets slack in this process if two of its neighbors happen to get colored with the same color, as in that case v only loses one available color from its color palette but two competing uncolored neighbors. Now, introduce a bad event for each node that holds if v does not have slack after this process. Despite many dependencies between the final colors of different nodes, one can show that this forms an LLL, but the bandwidth constraints of the CONGEST model make it extremely challenging to design efficient algorithms for this LLL. The bad event of a node v depends on the randomness of nodes in its two-hop neighborhood, as these nodes' color choices determine which colors are finally retained in v's neighborhood. Under bandwidth restrictions, v can't learn about the candidate colors of all of these nodes. To exploit parallelism, all existing sublogarithmic-round distributed LLL algorithms gradually and carefully set more and more of the variables. To steer decisions on future variables in these processes, it is pivotal that event nodes learn <u>all the information</u> about partial assignments of their variables. Hence none of these algorithms work efficiently in the CONGEST model.²

²The only exception is the implementation of [FG17] in [MU21] which is efficient for small maximum degree Δ the paper is formulated for $\Delta = O(1)$, but remains efficient for Δ up to poly log log *n*—, as one can still learn the required information for a single step of their algorithm in poly Δ rounds.

1.1 Our Contributions: LLL solvers

First, as a conceptual contribution, we formalize LLL problems in the CONGEST model.

While it is straightforward to assume that each event and variable is simulated by some node of the communication network, it is *a priori* unclear what knowledge nodes need regarding their events/variables. We reduce that knowledge or ability to a few simple primitives that can typically be implemented efficiently. Primarily, nodes need to be able to sample variables according to their distribution, measure how bad a partial solution is for events, and perform certain restricted communication between events and their variables. We refer to these LLLs as *simulatable* (see Definition 2.3). Note that the slack generation LLL as presented is not directly simulatable as one cannot measure the quality of partial variable assignments.

1.1.1 Disjoint variable set LLLs

As our second contribution, we present an LLL algorithm for the setting where intuitively the variables of each bad event are split into two sets, and a good assignment for at least one of the two variable sets is sufficient to avoid the bad event (formal statement in Theorem 5.2).

These LLLs appear frequently, e.g., when solving coloring LLL problems one may have colors from two different color spaces available. Another simple example is given by the sinkless orientation problem whose objective is to orient the edges of a graph such that every node has at least one outgoing edge [BFH⁺16a, GS17]. The probability that a degree Δ node has no outgoing edge is upper bounded by $2^{-\Delta}$ when orienting the edges randomly, proving that this is an LLL. Now, one can use the splitting algorithm from [HMN22] to split the edges into two sets such that every node has roughly $\Delta/2$ edges incident in each set, aligning the problem with our LLL framework and, to the best of our knowledge, yielding the first (published) CONGEST algorithm for the problem. Our approach in a nutshell. Our algorithm for disjoint variable set LLLs uses the influential shattering technique [Bec91, BEPS16]. First, we flip the variables in the first set at random according to their distribution to ensure that most of the events are avoided. A standard analvsis shows that this *shatters* the graph into small connected components—think of components of size $N = \text{poly} \log n$. Then, we use the second set of variables to avoid the remaining bad events. In the LOCAL model, the latter can be done with the known deterministic LLL algorithm from [FG17, RG20] applied on all components independently and in parallel. This post-shattering *phase* runs in poly $\log N = \operatorname{poly} \log \log n$ rounds, exploiting the components' small sizes. Our core technical contribution is an algorithm for the post-shattering phase in the CONGEST model. A deterministic LLL CONGEST algorithm is not known. Also, randomized algorithms are insufficient since their failure probability is $1/\operatorname{poly}(N) = 1/\operatorname{poly}(\log n)$ on each component, and almost surely one of the possibly many components will fail. Instead, we run $\Theta(\log n)$ independent executions of a randomized algorithm in parallel to amplify the error probability. That, however, places an additional burden on the bandwidth and becomes the central challenge to overcome. We leverage the small component size for coordination and information learning in a more efficient manner, e.g., by using significantly smaller ID spaces so that a single CONGEST message can encode $\Theta(\log n)$ IDs. There are several technical details that we spare in this introduction. For example, we cannot set all variables of a small component in one go, and partially setting only some of the variables comes

with the responsibility to ensure that there even exists a feasible assignment for the remaining unset variables.

1.1.2 Binary LLLs with low risk

In the absence of an alternative set of variables, the basic approach is to randomly sample all the variables, and then retract the variables around the events that fail under the initial assignment. The aim is then to solve another LLL on the subinstance induced by the retracted variables, redefining the events in terms of their marginal probability in the new instance, namely the probability that they hold conditioned on the assignment to the variables that are fixed. The good news is that this instance would be small (poly-log size), due to shattering, so we can afford to apply more powerful LLL solvers. The bad news is that this approach alone is seldom sufficient by itself since events that previously were not failing may now become highly unsatisfied by the retractions of adjacent events and there may not even exist an assignment of the retracted variables that avoids all bad events. Recursive retractions may lead to long chains, leading to at least $\Omega(\log n)$ rounds. The challenge is then how to limit retractions while ensuring a low conditional probability of bad events.

We treat a class of LLL with binary variables that occurs frequently in *sampling*, where the sampled nodes are black and the others white. Our approach is to perform a second round of retractions but only to the white variables. We bound from above the marginal probabilities of the shattered instance in terms of a parameter that we call *risk*.

As our third contribution, we present a bandwidth- and time-efficient algorithm for simulatable binary LLLs with low risk (formal statement in Theorem 4.4).

One example of such a problem is the sampling of a subset S of the nodes of a sparse graph G such that every node v of the graph has few neighbors in S but $G[N(v) \cap S]$ proportionally preserves the sparsity of v, i.e., the number of non-edges in its neighborhood. This Degree-Sparsity-Sampling problem (DSS) is also essential to our coloring results, where we sample two (or more) such sets that can serve as alternative sets of variables for the slack generation LLL, effectively enabling us to solve the slack generation problem via our first LLL solver. In Section 3, we explain the details of why the problem fits our LLL framework, including its non-trivial simulatability. Its formal solution is presented in Section 8. This example illustrates also that our LLL solvers are most powerful when used in tandem. In Section 7, we present several additional examples of problems (and schemas of problems) that can be solved efficiently with our LLL algorithms in CONGEST. Additionally, we show in Section 7 that any LLL that can be solved by the main LLL algorithm of [GHK18] has low risk and hence, in can also be solved with our framework in LOCAL.

We point out to the knowledgeable reader that the criteria of our LLL solvers are crucially in between polynomial and exponential whenever $\Delta \geq \text{poly} \log \log n$. This is the main regime of interest as there are poly $\log \log n$ -round LLL solvers for smaller Δ that work with a polynomial criterion and in CONGEST [MU21], and often the bounds on the error probability in LLLs turn into with high probability guarantees for larger Δ .

1.2 Our Contribution: Coloring Sparse and Triangle-Free Graphs

Graph coloring is fundamental to distributed computing, as an elegant way of breaking symmetry and avoiding contention, and was in fact the topic of the original paper introducing the LOCAL model [Lin92]. The typical setting that has been extensively studied is coloring a graph with $\Delta + 1$ colors; in the centralized setting, such a coloring can be computed via a simple greedy algorithm. Importantly for the distributed setting, any partial solution to the problem can be extended to a full solution without ever needing to revert any coloring choice, a property that evidently does not hold when coloring with fewer colors and inherently makes it much more difficult to color large parts of the input graph in parallel. Sparse graphs admit colorings with $\ll \Delta$ colors, and logarithmic-time LOCAL algorithms are known, e.g., to color triangle-free graphs with $\ll \Delta$ colors [CPS17].

We use our LLL algorithms to improve upon this result in two ways: First, our runtime is exponentially faster, second, in contrast to the prior algorithms our algorithms work in the CONGEST model. We summarize our results in the following theorem. Recall, that a node is *sparse* if there are many non-edges in its induced neighborhood (see Section 8 for the precise sparsity requirement).

There is a randomized CONGEST algorithm that w.h.p. colors any triangle-free graph with $\Delta - \Omega(\Delta)$ colors and any locally sparse graph with $\Delta - \text{poly} \log \log n$ colors. The algorithms run in poly log log n rounds (formal statements in Theorems 8.8 and 8.9).

1.3 Further Related Work

Due to its local nature, LLLs are a powerful tool in distributed computing, with several papers tackling its distributed complexity, e.g., [BFH⁺16b, CPS17, FG17, BMU19, BGR20, GHK18, CP19, CHL⁺20, RG20, MU21, Dav23]. The prime characteristic of a (symmetric) LLL is its *LLL criterion* that is the relation of the dependency degree d (the maximum degree in \mathcal{H}), and the global upper bound p on the probability for each bad event to hold. The original lemma of Lovász and Erdös [EL74] showed the existence of a feasible assignment as long as ep(d + 1) < 1 holds. A simplified summary is that essentially all LLLs of interest $((1 + \varepsilon)epd < 1$ is required) can be solved in $O(\log n)$ rounds of LOCAL [MT10, CPS17], while superfast poly(log log n) algorithms are only known for a special class of "(near) exponential" LLLs $(p2^{O(d/\operatorname{poly}\log\log n)} < 1)$ or [Dav23] or very low-degree (i.e., poly(log log n)-degree) graphs under polynomial LLL criteria $(pd^{32} < 1)$ [FG17], or when events satisfy certain additional robustness conditions [GHK18]. See [RG20, GHK18] for deterministic algorithms for polynomial LLLs. If the strong condition $p < 2^{-d}$ holds LLLs can be solved deterministically in poly $\Delta + O(\log^* n)$ rounds [BMU19, BGR20]. In CONGEST, the only algorithms known are randomized and for low-degree graphs [MU21] and for "vertex splitting" problems [HMN22].

There are countless publications on the classic topic of distributed graph coloring with $\Delta + 1$ colors focusing on different aspects of the problem, e.g., for coloring small degree graphs efficiently [Bar15, BEG18, FHK16, MT20, Mau21, FK23], for efficient deterministic coloring in LOCAL [RG20, GK21, GG23] and in CONGEST [BKM20, GK21], and for randomized coloring algorithms in LOCAL [Joh99, HSS16, CLP20, HKNT22] and in CONGEST [HKMT21, HN21, HNT22]. See also [BE13] as a great resource covering many early results on the topic.

Coloring with fewer colors: Recently, highly involved algorithms were designed to color nonclique graphs with $\Delta \geq 3$ with one fewer color, that is, with Δ colors [GHKM18, FHM23]. The resulting designed poly log log n algorithm works in the LOCAL model, but the algorithm inherently does not work in the CONGEST model as one subroutine is based on learning the full topology of $\omega(1)$ -diameter subgraphs.

Chung, Pettie, and Su [CPS17] give a LOCAL algorithm for Δ/k -coloring triangle-free graphs for any $k = O(\log \Delta)$, running in $O(\log n)$ time (faster for very large Δ). A much more constrained notion of sparsity is the *arboricity* of a graph, that is, the number of forests into which one can partition the graph. For any constant $\varepsilon > 0$, there is a $O(\log n)$ -round deterministic algorithms to color graphs with arboricity α with $O(2 + \varepsilon)\alpha$ colors, and it is known that the runtime is tight, even for randomized algorithms [BE10].

Notation. For a graph G = (V, E) and two nodes $u, v \in V$ let $\operatorname{dist}_G(u, v)$ denote the length of a shortest (unweighted) path between u and v in G. For a set $S \subseteq V$ we denote $\operatorname{dist}_G(v, S) = \min_{u \in S} \operatorname{dist}_G(v, u)$. For an integer $k \geq 0$ and a node $v \in V$ of a graph G = (V, E) let $N_G^k(v) = \{u \in V : \operatorname{dist}_G(v, u) \leq k\}$. For a set S, we define $N_G^k(S) = \bigcup_{v \in S} N_G^k(v)$.

1.4 Outline of the rest of the paper

Section 2 contains our LLL formalization in CONGEST, followed by Section 3 in which we present a technical overview of all results and techniques. Section 4 contains our LLL algorithm for LLLs with low risk. Section 5 presents our LLL algorithms working with two alternating sets of variables. Both of our LLL algorithms use the explained shattering framework that consists of a pre-shattering phase and a post-shattering phase. Our solution to the post-shattering phase is presented in Section 6. In Section 7, we present several applications of our LLL algorithms. In Section 8, we present our algorithm for DSS and our algorithms for coloring triangle-free and sparse graphs with few colors.

2 Distributed Lovász Local Lemma (Definitions)

In this section, we present our formalization of distributed LLL in the CONGEST model.

2.1 Constructive Lovász Local Lemma (LLL)

An instance $\mathcal{L} = (\mathcal{V}, \mathcal{B})$ of the distributed Lovász local lemma (LLL) is given by a a set $\mathcal{V} = \{x_1, \ldots, x_{k_{\mathcal{V}}}\}$ of independent random variables and a family \mathcal{B} of "bad" events $\{\mathcal{E}_1, \ldots, \mathcal{E}_{k_{\mathcal{B}}}\}$ over these variables. Let $\mathsf{vbl}(\mathcal{E})$ denote the set of variables involving the event \mathcal{E} and note that \mathcal{E} is a binary function of $\mathsf{vbl}(\mathcal{E})$. The dependency graph $\mathcal{H}_{\mathcal{L}} = (\mathcal{B}, F)$ is a graph with a vertex for each event and an edge $(\mathcal{E}, \mathcal{E}') \in F$ whenever $\mathsf{vbl}(\mathcal{E}) \cap \mathsf{vbl}(\mathcal{E}') \neq \emptyset$. The dependency degree $d = d_{\mathcal{L}}$ is the maximum degree of $\mathcal{H}_{\mathcal{L}}$. We omit the subscript \mathcal{L} when the considered LLL is unambiguous. Additionally, we use the parameters $d_{\mathcal{E}} = \max_{\mathcal{E}\in\mathcal{B}} |\mathsf{vbl}(\mathcal{E})|$ as the event degree and $d_{\mathcal{V}} = \max_{x \in \mathcal{V}} |\{\mathcal{E} \in \mathcal{B} | x \in \mathsf{vbl}(\mathcal{E})\}|$ as the variable degree. Define $p = \max_{\mathcal{E}\in\mathcal{B}} \Pr(\mathcal{E})$ (or let p simply be an upper bound on the term on the right-hand side). The Lovász Local Lemma [EL74] states that $\Pr(\cap_{\mathcal{E}\in\mathcal{B}}\bar{\mathcal{E}}) > 0$ holds if ep(d+1) < 1, or in other words, there exists an assignment to the variables that avoids all bad events.

In the constructive Lovász local lemma one aims to compute such an feasible assignment, avoiding all bad events. This is often under much stronger conditions on the relation of p and d. The relation of p and d is referred to as the *LLL criterion*. If $pd^c < 1$ for some constant c > 1 we speak of a polynomial criterion, while if $p2^d < 1$, we have an exponential criterion. The problems we consider have criteria that are between the polynomial and exponential.

2.2 Constructive Distributed Lovász Local Lemma

In the distributed setting, the LLL instance \mathcal{L} is mapped to a communication network G = (V, E). We are given a function $\ell : \mathcal{B} \cup \mathcal{V} \to V$ that assigns each variable and each bad event to a node of the communication network. We assume that for each variable $x \in \mathcal{V}$, the node $\ell(x)$ knows the distribution of x including the range range(x) of the variable. We also say that node $\ell(x)$ simulates the variable/event x. For a vertex $v \in V$, we call $l(v) = |\ell^{-1}(v)|$ the load of vertex v. The (maximum) vertex load of an LLL instance is $l = \max_{v \in V} l(v)$. In the constructive distributed LLL we execute a LOCAL or CONGEST algorithm on G to compute a feasible assignment φ . Afterwards, for each variable $x \in \mathcal{V}$, node $\ell(x)$ has to output $\varphi(x)$.

In general, the graph G and the dependency graph $H_{\mathcal{L}}$ do not have to coincide and d, $d_{\mathcal{E}}$, $d_{\mathcal{V}}$ and the maximum degree Δ of G shall not be confused with each other. However, distances between events in $H_{\mathcal{L}}$ and the corresponding nodes in G should be closely related.

Definition 2.1. A triple (\mathcal{L}, G, ℓ) has *locality* ν if $\operatorname{dist}_G(\ell(\mathcal{E}), \ell(x)) \leq \nu$ for all events \mathcal{E} of \mathcal{L} and variables $x \in \operatorname{vbl}(\mathcal{E})$.

Observation 2.2. We have $\operatorname{dist}_G(\ell(\mathcal{E}), \ell(\mathcal{E}')) \leq 2\nu$ for all dependent events $\mathcal{E}, \mathcal{E}'$.

Note that the splitting problems of [HMN22] modeled as LLLs have unit locality and unit distance between dependent events, which makes them particularly amenable to CONGEST implementations.

If (\mathcal{L}, G, ℓ) has small locality, then in the LOCAL model we can perform all natural basic operations on events and variables of \mathcal{L} efficiently. Examples of such operations are testing whether an event holds under a variable assignment [MT10, CPS17] or computing conditional event failure probabilities under a partial assignment [FG17, Dav23]. Any LOCAL algorithm for the dependency graph can be simulated in the communication network with a multiplicative overhead of $O(\nu)$ in the round complexity, where ν is the locality of the LLL instance.

(Partial) Assignments. An assignment of a set of variables \mathcal{V} is a function that assigns each variable $x \in \mathcal{V}$ a value in range(x). We use the value \perp for variables that have not been set. A partial assignment φ of a set of variables \mathcal{V} is a function with domain \mathcal{V} satisfying $\varphi(x) \in \operatorname{range}(x) \cup \{\perp\}$ for all $x \in \mathcal{V}$. A partial assignment ψ agrees with another (partial) assignment φ if $\psi(x) = \varphi(x)$ for all $x \notin \psi^{-1}(\perp)$, i.e., if all proper values assigned by ψ match those of φ . For two partial assignments φ_1, φ_2 with $\perp \in \{\varphi_1(x), \varphi_2(x)\}$ for all variables x, let $\varphi(x) = \varphi_1 \cup \varphi_2$ be the assignment with $\varphi(x) = \varphi_1(x)$ whenever $\varphi_1(x) \neq \perp$ and $\varphi(x) = \varphi_2(x)$, otherwise. For an event \mathcal{E} and a partial assignment with $\mathsf{vbl}(\mathcal{E}) \cap \varphi^{-1}(\perp) = \emptyset$, the term $\mathcal{E}(\varphi)$ states whether \mathcal{E} holds under φ .

A retraction ψ of a partial assignment φ is a partial assignment that agrees with φ . We say that we retract a variable x of φ if we set its value to \bot (formally this creates a new partial assignment). For an event \mathcal{E} and a partial assignment φ , we use the notation $\Pr(\mathcal{E} \mid \varphi)$ for the conditional probability over assignments with which φ agrees (the randomness is only over the variables in $\varphi^{-1}(\bot)$).

2.3 Simulatable Distributed Lovász Local Lemma (CONGEST)

In the CONGEST model, a small locality of the function ℓ does not ensure that basic primitives can be executed efficiently. In Section 3.2, we discuss the challenge of even evaluating the status of events via the example of the degree-bounded sparsity splitting problem (DSS).

A second challenge that we quickly touched upon appears from the need to make progress in large parts of the graph in parallel which requires us to set many (but not all) variables in parallel. The main difficulty is to ensure that we never run into an unsolvable remaining problem, that is, we need to ensure that the remaining variables can always be assigned values such that a feasible solution avoiding all bad events is obtained. This is in stark contrast to problems like computing a maximal independent set or a $\Delta + 1$ -coloring in which any partial solution can always be completed to a solution of the whole graph. Thus, to make progress in large parts of the graph in parallel, we need to measure how bad a partial assignment is for events that do not have all their variables set. Naturally, for a bad event \mathcal{E} and partial assignment ψ this is captured by the conditional probability $Pr(\mathcal{E} \mid \psi)$. The LOCAL model works of [FG17, GHK18, Dav23] implicitly compute these values. However, in CONGEST it can be impossible to compute such marginal probabilities.

In our simulatability definition (see Definition 2.3) we do not require that conditional probabilities can be computed in the most general setting but only in the easier setting where we are given locally unique IDs from a small ID space. In the presence of little bandwidth, this helps significantly in some of the LLLs considered in this work and we believe that it will be helpful for other problems.

Next, we state the minimal assumptions that we require from an LLL instance.

Definition 2.3 (simulatable). We say an LLL (\mathcal{L}, G, ℓ) is *simulatable* in CONGEST if each of the following can be done in poly log log n rounds:

- 1. Test: Test in parallel which events of \mathcal{L} hold (without preprocessing).
- 2. Min-aggregation: Given 1-bit string in each event (variable), each variable (event) can simultaneously find the minimum of the strings for its variables (for its events).

For the following items, it is sufficient if they hold in the setting that events and variables are given $O(\log \log n)$ -bit IDs³ (that are unique within distance 4ν in G):

3. Evaluate: Given a partial assignment φ , and partial assignments ψ_1, \ldots, ψ_t , $t = O(\log n)$, in which each variable knows its values (or \perp), each event \mathcal{E} of \mathcal{L} can simultaneously for all $1 \leq i \leq t$ decide if

$$\Pr(\mathcal{E} \mid \psi_i) \le \alpha \Pr(\mathcal{E} \mid \varphi)$$

holds, where α is a parameter known by all nodes of G.

4. **Min-aggregation:** We can compute the following for $O(\log n)$ different instances in parallel: Given an $O(\log \log n)$ -bit string in each event (variable), each variable (event) can simultaneously find the minimum of the strings for its variables (for its events).

Min-aggregation to variables allows events to retract their variables. Similarly, min-aggregation to events allows events to decide if they have a retracted variable. Min-aggregation with larger messages is used to find acyclic orientations of the dependency graph, a necessary step of the LLL algorithm of [CPS17]. Due to the presence of smaller IDs, primitives (3) and (4) appear technical, but we emphasize that these are crucial to our CONGEST solution. Further, it is unlikely that any sublogarithmic-time algorithm can go along without measuring the quality of partial assignments in one way or the other; with large IDs, the respective quality of partial assignments can provably not be checked in CONGEST.

3 Technical Overview & Technical Contributions

In Section 3.1, we give a technical overview of our disjoint variable set LLL solver. In Section 3.2, we present the crucial ingredients of LLL solver for binary LLLs with low risk and explain how to use it for the degree-bounded sampling problem. In Section 3.3, we present a condensed version of our approach to the LLLs arising in the post-shattering phase of our algorithms. In Section 3.4, we sketch how to use our LLL solvers to obtain our coloring results.

 $^{^{3}}$ In general for the whole LLL instance and for non-constant distances such identifiers do not exist, but our LLL algorithms only use the primitives in settings where they do exists and are available.

3.1 Disjoint Variable Set LLLs

In this section, we present algorithms for *disjoint variable set LLLs*, where we have two disjoint sets of variables $\mathcal{V}_1, \mathcal{V}_2$ available for each event. In fact, we consider events \mathcal{E} that can be written as the conjunction of two events $\mathcal{E}_1, \mathcal{E}_2$ where $\mathsf{vbl}(\mathcal{E}_i) = \mathcal{V}_i$ and $\Pr(\mathcal{E}_i) \leq p$ holds for i = 1, 2. Note, that to avoid \mathcal{E} it is sufficient to avoid \mathcal{E}_1 or \mathcal{E}_2 . Next, we sketch our algorithm.

► We first sample all variables in \mathcal{V}_1 according to their distribution (pre-shattering phase), and then move all non-avoided events (formally when \mathcal{E}_1 is non-avoided) to the post-shattering phase. There we use the variables in \mathcal{V}_2 to avoid the respective second events \mathcal{E}_2 . \blacktriangleleft In contrast to our binary LLL solver, there are no retractions. For suitable p, the property $\Pr(\mathcal{E}_1) \leq p$ ensures that the components in the post-shattering phase are of size $N = O(\log n \operatorname{poly} d)$, and the property $\Pr(\mathcal{E}_2) \leq p$ ensures that each component in the post-shattering phase is an LLL. The main focus of our work is the case where d is at most polylogarithmic, in which case $N = \operatorname{poly} \log n$. In LOCAL, we use Theorem A.2 to solve each small component in poly $\log N = \operatorname{poly} \log \log n$ rounds. We obtain the following theorem.

Theorem 5.2. There are randomized LOCAL and CONGEST algorithms that in polylog log n rounds w.h.p. solve any <u>disjoint variable set LLL</u> of constant locality ν with dependency degree $d \leq \text{polylog } n$ and bad event upper bound p. The LOCAL algorithm requires $p < d^{-14}$ and the CONGEST algorithm requires $p < d^{-(2+c_l)-(4c+12c_{\Delta}\nu)\log\log n}$, $l \leq d^{c_l}$, $\Delta \leq \log^c n$ for constants $c_l, c_{\Delta} \geq 1$, and simulatability.

The CONGEST part of the theorem is more challenging for several reasons. We have already discussed the challenges regarding the evaluation of events and measuring progress for partial solutions. Another challenge is that shattering the dependency graph is not enough but the standard analysis only shatters the dependency graph. The issue is that, in CONGEST, one cannot independently deal with different components of the dependency graph if the mapping of the events/variables of different components to the communication network overlaps.

Thus, our solution relies on a stronger form of shattering in which we guarantee that after mapping the components of the dependency graph to the communication network, the components in the communication network remain small. This is one of the reasons why we require a stronger LLL criterion in CONGEST; note that all of our applications satisfy this stronger criterion.

The second difference between our CONGEST and LOCAL solutions lies in the post-shattering phase which we detail in Section 3.3.

3.2 Binary LLLs with low risk

In this section, we consider binary LLLs, that is, the range of the variables is {black, white}.

Outline of our binary LLL algorithm: In our algorithm, each (original) event \mathcal{E} comes with an associated event $\operatorname{assoc}(\mathcal{E})$ (usually on the same variable set) that imposes stricter conditions, i.e., it is harder to avoid, but avoiding $\operatorname{assoc}(\mathcal{E})$ implies avoiding \mathcal{E} . \blacktriangleright We first flip all variables according to their distribution and then retract all variables around failing associated events. Then, we perform a second round of retractions in which we only retract white variables around those events that were affected by a (partial) retraction in the first round of retractions. We then form the residual LLL instance on the set of unset variables and all incident events. The probability of an event is now the conditional probability given the assignment to the unretracted variables. We apply our post-shattering solver to this instance to produce the final solution. \blacktriangleleft

We introduce a new term, *risk*, which essentially upper bounds the conditional probability of a bad event to hold in the post-shattering phase under these promises. To show that it is small,

we leverage the properties that hold for our retractions. Consider the four cases that can occur for an event: a) the event was unhappy, i.e., $assoc(\mathcal{E})$ occurred on the initial assignment, so all incident variables are retracted; b) the event was *affected*: it was happy, but had incident variables retracted in the first round, so all incident white variables were retracted in second round; c) event was *impacted*: it was not unhappy and not affected, but some incident white variables were retracted in second round (by another event), and d) the event was *at peace*, with no incident variables retracted. We want to ensure that the conditional probability of the event is low in all these cases.

We say that an event pair $\mathcal{E}, \mathsf{assoc}(\mathcal{E})$ testifies risk x, if

- 1. $Pr(\mathsf{assoc}(\mathcal{E})) \leq x$, and
- 2. the marginal probability in cases a)-d) is at most x.

Condition (1) is required to ensure that the probability of becoming unhappy is small such that the process shatters the graph into small components for the post-shattering phase. As \mathcal{E} implies $\mathsf{assoc}(\mathcal{E})$, condition (1) also ensures that the marginal probability for all events in case a) is bounded above by x. Observe that in b)-d), the final assignment ψ for the post-shattering phase was one derived from the initial one φ for which the event $\mathsf{assoc}(\mathcal{E})$ did not take place. The guarantees on ψ for an event \mathcal{E} are that either no black variables were retracted from φ or all white variables were retracted from φ . We say that in this case ψ respects the initial assignment φ . Thus, condition (2) is equivalent to the following statement

 $\triangleright \Pr(\mathcal{E} \mid \psi) \leq x$ holds, for any assignment ψ that respects some $\operatorname{assoc}(\mathcal{E})$ -avoiding assignment.

In general, we show the following.

Theorem 4.4. There are randomized LOCAL and CONGEST algorithms that in polylog log n rounds w.h.p. solve any LLL of constant locality ν with dependency degree $d \leq$ polylog n and <u>risk</u> p. The LOCAL algorithm requires $p < d^{-14}$ and the CONGEST algorithm requires $p < d^{-(4+c_l)-(4c+12c\nu)\log\log n}$, $l \leq d^{c_l}$, $\Delta \leq \log^{c_{\Delta}} n$ for constants $c_l, c_{\Delta} \geq 1$ and that the LLL is simulatable.

The main difficulty with using Theorem 4.4 is to bound the risk of an LLL. As we prove in Section 7, any LLL that can be solved with the main LLL algorithm of [GHK18] has low risk. Hence, in the LOCAL model, our algorithm subsumes the one of [GHK18]. The issue is that it is difficult and technical to prove that an LLL fits the framework of [GHK18] (also see Section 7 details). The core benefit of our approach is that it is superior for any binary LLL containing monotonically increasing events, that is, events that favor more nodes to be sampled. Examples are satisfying a minimum degree bound into a set of sampled nodes, or, as in the DSS problem, a minimum sparsity in the sample. In either case, the respective bad events are easier to avoid if we add more nodes to the sample. We prove the following lemma, formally proven in Section 7 (Lemma 7.3).

No risk Lemma. The risk of a monotone increasing event \mathcal{E} is $\Pr(\mathcal{E})$ testified by $\operatorname{assoc}(\mathcal{E}) = \mathcal{E}$.

The name of the lemma stems from the fact that there is no additional risk from the conditional probabilities of the post-shattering phase. The conditional probability is identical to the probability of the event in the original LLL. Intuitively, the lemma holds, as any affected event \mathcal{E} retracts all of its white incident variables, essentially, giving it free randomness for the post-shattering phase.



Figure 1: An illustration of the cases a)-d) that can appear in the post-shattering phase of the degreebounded subgraph problem. Note that the illustration is only schematic and such a tight example with $\Delta = 6$ does not satisfy any LLL criterion. The colors represent the variable assignments after the initial sampling. The question mark indicates that the respective variable got retracted and participates in the post-shattering phase. In a), the vertex v got an extremely bad split and retracted all of its incident variables. In b), v is affected by a retraction from a type a) node, and hence retracts all of its incident white variables. In c), we see a node that was neither unhappy nor affected, but has some of its white variables retracted by some other node of type c). The node in d) is happy and does not undergo any retractions. It does not participate in the post-shattering phase. W.h.p. the bulk of the nodes are of type d).

The fact that some of its adjacent variables may remain black can only make the situation better as the event prefers black anyways and its conditional probability is upper bounded by its initial probability $Pr(\mathcal{E})$.

Example 1 (degree bounded subgraphs): Let us see our framework in action with a first simple example. We are given a Δ -regular graph and an integer k (with $k \leq \Delta/6$ and $k \gg \log \Delta$) and seek a subgraph S such that each node has at least k/3 and at most 4k neighbors in S. For each node v we have events \mathcal{E}_v^{\min} and \mathcal{E}_v^{\max} that hold if the number of neighbors of v in S is less than k/3 and more than 4k, respectively. For \mathcal{E}_v^{\max} the associated event asks to maintain a stronger of bound of 2k on the number of neighbors of v in S. For \mathcal{E}_v^{\min} the associated event is the event \mathcal{E}_v^{\min} itself.

After an initial random sampling into a set S with probability $q = k/\Delta$, each node has expected degree k into S. Nodes with degree outside the range [k/3, 2k] are unhappy, so all neighbors are retracted (both sampled and unsampled neighbors become *undetermined*). Now, nodes that had an adjacent neighbor retracted go through the second round of retraction, with all *unsampled* neighbors becoming undetermined. The post-shattering LLL is formed in terms of the undetermined nodes. Also see Figure 1 for an illustration of the four types of nodes (a)–d)).

We reason that the risk of all events is small. \mathcal{E}_v^{\min} is monotonically increasing as it favors more nodes in the sample and by the No Risk Lemma its risk equals $Pr(\mathcal{E}_v^{\min})$ which is upper bounded by $\exp(-\Theta(k))$ by a Chernoff bound. Similarly, a Chernoff bound shows that the probability of $\operatorname{assoc}(\mathcal{E}_v^{\max})$, asking for at most 2k sampled neighbors, is at most $\exp(-\Theta(k)) \ll 1/\operatorname{poly}(\Delta)$, proving (1). This implies that the process shatters and we obtain small unsolved components in the postshattering phase. To argue that the post-shattering phase indeed is a solvable LLL, what is left to prove is that \mathcal{E}_v^{\max} has risk $x = 1/\operatorname{poly}(\Delta)$; we already bounded $Pr(\operatorname{assoc}(\mathcal{E}_v^{\max}))$ for proving (1).

To prove (2), let us derive the conditional probabilities for all four types of nodes, and thereby the risk. The unhappy node has all its incident variables remain in the post-shattering instance, by Chernoff bound the conditional probability of \mathcal{E}_v^{\max} is $\exp(-\Theta(k)) \ll 1/\operatorname{poly}(\Delta)$. The affected node has at most 2k incident nodes set (only the black ones) and therefore at least $\Delta - 2k \ge 2\Delta/3$ incident variables undetermined. By Chernoff, adding more than 2k black incident variables is highly unlikely. Hence, its final number of neighbors in S will be larger than 4k, with probability



Figure 2: An example of a node v with $\Delta = 8$ neighbors and two examples of sampled subsets (red patterned nodes, black nodes). The dashed edges are non-edges, that is, all other edges e.g., the edge $\{1, 6\}$ are present in the graph. This neighborhood has 14 non-edges out of the $\binom{8}{2} = 28$ tentative edges. While v has only degree $4 = \Delta/2$ into either of the two subsets, the red patterned subset would be a sampled subset with a small sparsity, as it only contains the single non-edge $\{1, 8\}$. The black subset has larger sparsity, as it contains five non-edges $\{2, 3\}, \{3, 4\}, \{2, 4\}, \{4, 6\}, \text{ and } \{3, 6\}$. The number of non-edges in a sampled set is not a linear function of the nodes' sampling status.

 $\exp(-\Theta(k))$. The impacted node has some white variables retracted, so its degree into S may increase in the post-shattering step, but again by Chernoff, it will increase by more than 2k with probability $\exp(-\Theta(k))$. Finally, the events at peace already satisfy the requirement. Hence, the conditional probabilities of \mathcal{E}_v^{\max} are all at most $\exp(-\Theta(k)) \ll \Delta^{-32}$, bounding the event's risk.

Not having to analyze how the white-node-retraction affects the conditional probabilities of \mathcal{E}_v^{\min} , but instead relying on the No Risk Lemma is particularly helpful in our next example where the \mathcal{E}_v^{\min} is replaced with a significantly harder to deal with black-favoring event.

Example 2 (DSS): Recall that we seek a subgraph S such that for each node v, the graph $G[S \cap N(v)]$ has both low-degree and large sparsity. Let \overline{m}_v denote the number of non-edges within G[N(v)]. Also see Figure 2.

Specifically, given a sampling probability q, the expected number of non-edges in $G[S \cap N(v)]$ is $q^2 \overline{m}_v$. Again, we have a bad event \mathcal{E}_v^{\max} that holds if $G[S \cap N(v)]$ has more than $6q\Delta$ vertices, and a bad event \mathcal{E}_v^{\min} that holds if $G[S \cap N(v)]$ has fewer than $q^2 \overline{m_v}/6$ non-edges. The stricter respective events have the tighter bound of $2q\Delta$ on the degree into S, and the same lower bound on the number of non-edges. Recall that nodes sampled into S are black, those not sampled are white, while retracted nodes become *undetermined*.

After this setup, the proof for bounding the respective risks is identical to the first example. \mathcal{E}_v^{\max} and its associated event are of the same nature as in Example 1, and its risk can be bounded using identical arguments. \mathcal{E}_v^{\min} is a monotone increasing event, which again, by the No Risk Lemma has risk $\Pr(\mathcal{E}_v^{\min})$. Bounding this probability is challenging in itself as it requires a concentration bound that handles dependencies (for reasoning about the number of non-edges in the sample), but the conclusion is the same: the risk is $O(\Delta^{-32})$, as desired. Here, the No Risk Lemma shows its power, as without it, one would have to deal with dependencies and hard-to-grasp conditional probabilities of partial assignments at the same time.

Simulatability in CONGEST: Another crucial benefit of our framework is that one can easily mix and mingle events. This is pivotal to ensure that the DSS is simulatable. Given some sample S, a node v cannot even efficiently determine the number of non-edges in $G[N(v) \cap S]$, even if Δ is only polylogarithmic, as encoding the topology of the graph requires $\Omega(\Delta^2)$ IDs and a node can only receive Δ IDs of information per communication round. But, a node can easily determine its degree $d_S(v)$ in G[S], and reject the sample if its degree is too large. On the other hand, if the degree is small, also encoding the topology of the sampled subgraph in v's neighborhood can be done more efficiently and hence the DSS-LLL becomes simulatable. The No Risk Lemma is also helpful in that respect, as it shows that the associated event of a monotone increasing event is the event itself. Recall, that in the algorithm the associated event is needed to raise a red flag whenever the initial sampling goes wrong for the respective event. Prior work implicitly used involved associated events based on conditional probabilities which cannot be computed in CONGEST.

3.3 Post-shattering in CONGEST

In the post-shattering phase, we are given another LLL \mathcal{L} in a network with significantly fewer nodes, i.e., each component has at most $N = \text{poly} \log n$ nodes. Hence, the original criterion can be restated as $p < d^{-\Omega(\log \log n)} = d^{-2\log N}$, while the bandwidth remains the original $\Theta(\log n)$. In the LOCAL model, we can immediately solve this in poly $\log \log n$ rounds via Theorem A.2, but that requires large bandwidth for gathering large parts of the graph at a single node.

▶ In our CONGEST algorithm, we first compute a network decomposition of the components into $C = O(\log N)$ collections consisting of $O(\log^2 N)$ -diameter clusters [RG20, GGR21] with a distance $k = 2\nu$ between clusters in the same collection (recall, ν is the locality of the LLL). Then, we iterate through the collections in sequence. Before each iteration *i*, we have a partial assignment φ_{i-1} , formed by the assignments made in previous iterations, and formulate a new LLL \mathcal{L}_i on the unset variables of nodes in the *i*-th collection. The bad events of \mathcal{L}_i ensure that after setting these variables the conditional probability of each original event (of \mathcal{L}) increases at most by a factor d^2 . By Markov's inequality the probability that the increase is larger than d^2 if a subset of the variables of an event are sampled is at most $1/d^2$ (see Claim 6.5 for details). By induction over *i*, we maintain the invariant that after processing the *i*-th collection, we have $\Pr(\mathcal{E} \mid \varphi_i) \leq p \cdot d^{2i}$ for each bad event \mathcal{E} of \mathcal{L} . Thus, at the end $\Pr(\mathcal{E} \mid \varphi) \leq pd^{2C} < 1$, if we assume the criterion $p < d^{-2C}$. Since all variables have been fixed by $\varphi = \varphi_{C+1}$, the event \mathcal{E} is avoided under the final assignment φ .

To solve \mathcal{L}_i on each cluster (and thus each collection), we run $O(\log n)$ parallel instances of the LLL algorithm of [CPS17]. Each of them succeeds (avoids all bad events of \mathcal{L}_i) with probability $1 - 1/N \ge 1/2$. Thus, at least one of these instances succeeds w.h.p. To determine a successful assignment, we use bitwise aggregation, utilizing the small cluster diameter.

Simulatability (Definition 2.3) is the key to solving these instances in parallel in CONGEST. It ensures that each of the steps of the [CPS17] algorithm of all instances in parallel can be implemented fast enough. The full details are in Section 6. We also need to efficiently communicate between events and their variables, e.g., to resample variables. Note that many of our LLLs can only perform these steps efficiently after we compute smaller locally unique node IDs from an ID space of size poly N, which only requires $O(\log \log n)$ bits per ID.

Lastly, we want to remark that the idea of amplifying probabilities by running several instances of an algorithm in parallel has been used before, but in significantly simpler settings [HMN22, Gha19].

3.4 Coloring Sparse Graphs

Recall, that providing slack to sparse nodes by partially coloring the graph can be modeled as an LLL. Once uncolored nodes have slack, we can complete their coloring by a simple deg + 1-list coloring procedure from prior work (this brings us back to the greedy coloring regime that is well-understood), [HNT22]. Unfortunately, as discussed, the slack generation LLL is not simulatable and cannot be tackled easily in the CONGEST model. To provide slack to nodes in CONGEST, we use the DSS problem to compute two (or more) degree-bounded sparsity-preserving sets S_1 and S_2 . Having these degree-bounded sets with many non-edges in each neighborhood has several benefits.

First of all, if we only color nodes in the degree-bounded sets, the slack generation problem becomes simulatable. Secondly, we also partition the color space into two linearly-sized sets that are then used for coloring S_1 and S_2 , respectively. As every node can obtain slack from coloring nodes in either set, this effectively splits the slack generation LLL variables into two sets and aligns with our two disjoint sets LLL solver. We obtain a slack generation algorithm with runtime poly log log n.

4 Binary LLLs with low Risk

In this section, we consider binary LLLs, that is, the range of the variables is {black, white}.

For an event \mathcal{E}' , let $\mathsf{Retract}(\mathcal{E}')$ consist of all assignments ψ that are a retraction of some full assignment φ under which \mathcal{E}' is avoided. Let $\mathsf{Respect}(\mathcal{E}') \subseteq \mathsf{Retract}(\mathcal{E}')$ be the set of assignments ψ that additionally have the guarantee that either all white variables under φ in $\mathsf{vbl}(\mathcal{E}')$ are retracted, i.e., $\mathsf{vbl}(\mathcal{E}') \cap \varphi^{-1}(\mathsf{white}) \subseteq \psi^{-1}(\bot)$, or no black variables under φ in $\mathsf{vbl}(\mathcal{E}')$ are retracted, i.e., $\mathsf{vbl}(\mathcal{E}') \cap \varphi^{-1}(\mathsf{black}) \cap \psi^{-1}(\bot) = \emptyset$.

Definition 4.1 (risk). We say that an event \mathcal{E}' testifies risk x for some event $\mathcal{E} \subseteq \mathcal{E}'$ if

$$\max\left\{\Pr(\mathcal{E}'), \max_{\psi \in \mathsf{Respect}(\mathcal{E}')} \{\Pr(\mathcal{E} \mid \psi)\}\right\} \le x \ . \tag{1}$$

The *risk* of an event \mathcal{E} is the smallest risk testified by some event $\mathcal{E}' \supseteq \mathcal{E}$.

The bound on $\operatorname{Pr}(\mathcal{E}')$ will ensure shattering. The bound on $\max_{\psi \in \operatorname{Respect}(\mathcal{E}')} \{\operatorname{Pr}(\mathcal{E} \mid \psi)\}$ upper bounds the marginal probabilities of event \mathcal{E} in the post-shattering phase. The intuition for the condition $\mathcal{E}' \supseteq \mathcal{E}$ is that we want \mathcal{E} to be avoided if an event is at peace during the whole process. The next definition captures the binary LLLs that we deal with in this section.

The next definition captures the binary LLLs that we dear with in this section.

Definition 4.2 (binary LLLs with low risk). A binary LLL with risk p consists of the following:

- \mathcal{V} a set of binary independent random variables with range {black, white},
- \mathcal{B} a set of events over \mathcal{V} with risk at most p and $\Pr(\mathcal{E}) \leq p$ for all $\mathcal{E} \in \mathcal{B}$,
- For each event $\mathcal{E} \in \mathcal{B}$ an associated event $\mathsf{assoc}(\mathcal{E})$ testifying its risk.

The dependency degree d is the maximum degree of the dependency graphs induced by all events. The goal is to compute an assignment of the variables in \mathcal{V} such that all events in \mathcal{B} are avoided. We extend the definition of simulatability of a binary LLL with low risk and also require that the associated events can also be evaluated in poly log log n rounds on any assignment of the variables.

Remark 4.3. Note that the risk of an event as given in Definition 4.1 minimizes over all possible associated events and as such may not be easily computable. Hence, in Definition 4.2, we require that the respective associated events are known in a simulatable manner to the nodes of the network.

We prove the following theorem.

Theorem 4.4. There are randomized LOCAL and CONGEST algorithms that in polylog log n rounds w.h.p. solve any LLL of constant locality ν with dependency degree $d \leq$ polylog n and <u>risk</u> p. The LOCAL algorithm requires $p < d^{-14}$ and the CONGEST algorithm requires $p < d^{-(4+c_l)-(4c+12c\nu)\log\log n}$, $l \leq d^{c_l}$, $\Delta \leq \log^{c_{\Delta}} n$ for constants $c_l, c_{\Delta} \geq 1$ and that the LLL is simulatable. We next present our algorithm for Theorem 4.4 that uses the shattering technique. The CONGEST version of the algorithm requires relies on the post-shattering algorithm from Section 6. Algorithm: Consider a binary LLL as in Definition 4.2.

- Initial sampling (Step 1): Sample all variables in V according to their distribution.
 Let φ be the resulting assignment.
- Retraction I: For each $\mathcal{E} \in \mathcal{B}$ for which $\mathsf{assoc}(\mathcal{E})$ holds under φ , retract all incident variables,
- Retraction II: For each $\mathcal{E} \in \mathcal{B}$ with an unset variable, retract all incident white variables, Let ψ_{pre} be the resulting partial assignment.
- **Post-shattering:** Set up the following LLL problem consisting of all unset variables and their incident (marginal) events.
 - Variables: $\mathcal{V}_{post} = \psi_{pre}^{-1}(\perp)$, with their respective original probability distribution,
 - Bad Events: $\mathcal{B}_{post} = \{ \mathcal{E} | \psi_{pre} : \mathcal{E} \in \mathcal{B}, vbl(\mathcal{E}) \cap \mathcal{V}_{post} \neq \emptyset \}$
 - $-\ell_{\text{post}}(x) = \ell(x)$ for all $x \in \mathcal{V}_{\text{post}}$ and $\ell_{\text{post}}(\mathcal{E}') = \ell(\mathcal{E})$ for all $\mathcal{E}' \in \mathcal{B}_{\text{post}}$. To simplify the notation, we refer to ℓ_{post} as ℓ .

We compute an assignment ψ_{post} of all variables in $\mathcal{V}_{\text{post}}$ avoiding all events in $\mathcal{B}_{\text{post}}$. In LOCAL we use Theorem A.2 and in CONGEST we use our algorithm from Lemma 6.1.

• Return $\psi = \psi_{\text{pre}} \cup \psi_{\text{post}}$.

Analysis of the post-shattering phase. We first show that the LLL formulation in the postshattering phase indeed is an LLL. Afterwards we show that with high probability the dependency graph of this LLL consists of small connected components.

Lemma 4.5. \mathcal{L}_{post} is an LLL with bad event probability bound p and dependency degree d.

Proof. The dependency degree of the LLL is upper bounded by the d in Definition 4.2.

Let φ be the assignment of the variables of Step 1. Let $\mathcal{E} \in \mathcal{B}_{post}$ be an arbitrary event. If $\operatorname{assoc}(\mathcal{E})$ holds under φ , then $\operatorname{vbl}(\mathcal{E}) \subseteq \mathcal{V}_{post}$ and we obtain $\operatorname{Pr}(\mathcal{E} \mid \psi_{pre}) = \operatorname{Pr}(\mathcal{E}) \leq p$ by Definition 4.2.

If $\operatorname{assoc}(\mathcal{E})$ is avoided under φ we obtain $\Pr(\mathcal{E} \mid \psi_{\mathsf{pre}}) \leq p$ by the definition of the risk (see Definition 4.1), as $\psi_{\mathsf{pre}} \in \mathsf{Respect}(\operatorname{assoc}(\mathcal{E}))$ and $\operatorname{assoc}(\mathcal{E})$ testifies the risk p.

Note that the statement in Lemma 4.5 holds "deterministically", that is, regardless of the outcome of the random choices in Step 1 of the algorithm. Let $W = \{v \in V | \ell^{-1}(v) \cap (\mathcal{V}_{\mathsf{post}} \cup \mathcal{B}_{\mathsf{post}}) \neq \emptyset\}$ be the set of nodes that have one of their events/variables participating in the post-shattering phase. Further, let $W' = \{v \in V : \operatorname{dist}_G(v, W) \leq \nu\}$ be the nodes that are in distance at most ν from W.

We obtain the following bounds for events, variables, and nodes to be part of the post-shattering.

Lemma 4.6. The following bounds hold.

- For each event $\mathcal{E} \in \mathcal{B}$ we have $\Pr(\mathcal{E} \in \mathcal{B}_{post}) \leq p \cdot d^2$,
- For each variable $x \in \mathcal{V}$ we have $\Pr(x \in \mathcal{V}_{post}) \leq p \cdot d^2 \cdot (d+1)$,

- For each node $v \in V(G)$ we have $\Pr(v \in W) \leq p \cdot d^2 \cdot (d+1) \cdot l$,
- For each node $v \in V(G)$ we have $\Pr(v \in W') \le p \cdot d^2 \cdot (d+1) \cdot l \cdot \Delta^{\nu}$.

For distinct \mathcal{E} and \mathcal{E}' the events $\mathcal{E} \in \mathcal{B}_{post}$ and $\mathcal{E}' \in \mathcal{B}_{post}$ are independent if $\operatorname{dist}_H(\mathcal{E}, \mathcal{E}') > 1$. For a node v the event whether it is contained in W or W' only depends on the randomness at nodes v' with $\operatorname{dist}_G(v, v') \leq 5\nu$ and $\operatorname{dist}_G(v, v') \leq 6\nu$, respectively.

Proof Sketch, formal proof thereafter. Each event or variable participating in the post-shattering phase has some event \mathcal{E} in its vicinity for which $\operatorname{assoc}(\mathcal{E})$ holds after the initial sampling. $\operatorname{Pr}(\operatorname{assoc}(\mathcal{E}))$ is bounded by p due to the bounded risk. The lemma follows with several union bounds over the respective sets of events and variables in multiple hop distance neighborhoods.

Proof of Lemma 4.6. Define the following sets of events and variables:

$$E_0 = \{ \mathcal{E} \in \mathcal{B} : \mathsf{assoc}(\mathcal{E}) \text{ holds under } \varphi \}, \qquad V_0 = \bigcup_{\mathcal{E} \in X_0} \mathsf{vbl}(\mathcal{E}), \qquad (2)$$

$$E_1 = \{ \mathcal{E} \in \mathcal{B} \setminus E_0 : \mathsf{vbl}(\mathcal{E}) \cap V_0 \neq \emptyset \}, \qquad V_1 = \bigcup_{\mathcal{E} \in X_1} \mathsf{vbl}(\mathcal{E}) \setminus V_0, \qquad (3)$$

$$E_2 = \{ \mathcal{E} \in \mathcal{B} \setminus (E_0 \cup E_1) : \mathsf{vbl}(\mathcal{E}) \cap V_1 \neq \emptyset \} .$$
(4)

Observe that the set E_0 consists of all events that retract a variable in the first step of retractions and V_0 contains all the retracted variables. E_1 contains those events that are not contained in E_0 but depend on a retracted variable. The events in E_1 cause the retraction of their white variables in the second round of retractions. The set V_1 consists of the variables that are retracted in that step and E_2 contains all events that are neither in E_0 nor in E_1 but are adjacent to a retracted variable. Altogether $E_0 \cup E_1 \cup E_2$ contains all events participating in the post-shattering phase and $V_0 \cup V_1$ contains all variables participating in the post-shattering phase.

Fix an arbitrary event \mathcal{E} . By Definition 4.2, the probability that \mathcal{E} is contained in E_0 is at most p. Furthermore, \mathcal{E} cannot be contained in $E_0 \cup E_1 \cup E_2$ if none of the events that are within distance at most 2 in the dependency graph $H_{\mathcal{L}}$ —note that this is the dependency graph of the original LLL and not the LLL in the post-shattering phase—are contained in E_0 . There are at most d^2 such events, and with a union bound over we obtain $\Pr(\mathcal{E} \in \mathcal{B}_{post}) = \Pr(\mathcal{E} \in E_0 \cup E_1 \cup E_2) \leq d^2 p$.

Fix an arbitrary variable $x \in \mathcal{V}$. In order for $x \in \mathcal{V}_{post}$ to hold, one of its $d_{\mathcal{V}}$ adjacent events needs to participate in \mathcal{B}_{post} . We obtain $\Pr(x \in \mathcal{V}_{post}) \leq p \cdot d^2 \cdot d_{\mathcal{V}} \leq d^2(d+1)$ with a union bound over these events.

Fix an arbitrary node $v \in V$. For $v \in W$ to hold, one of its l(v) events/variables needs to be contained in $\mathcal{B}_{post} \cup \mathcal{V}_{post}$. We obtain $\Pr(v \in W) \leq p \cdot l(v) \cdot d^2 \cdot d_{\mathcal{V}} \leq p \cdot l \cdot d^2 \cdot d_{\mathcal{V}}$ with a union bound over all these events/variables. Similarly, $v \in W'$ only holds if there is a node $u \in N^{\nu}(v)$ with $u \in W$. We obtain $\Pr(v \in W') \leq p \cdot l \cdot d^2 \cdot d_{\mathcal{V}} \cdot \Delta^{\nu}$ with a union bound over the Δ^{ν} nodes in $N^{\nu}(v)$.

Note that the event $(\mathcal{E} \in \mathcal{B}_{post})$ only depends on the random choices of variables of events in distance at most 2 in H. Thus, $(\mathcal{E} \in \mathcal{B}_{post})$ and $(\mathcal{E}' \in \mathcal{B}_{post})$ for events $\mathcal{E}, \mathcal{E}'$ depend on distinct sets of variables if dist_H $(\mathcal{E}, \mathcal{E}') > 4$.

For a node $v \in V$ the event $(v \in W)$ holds if one of the events/variables in $\ell^{-1}(v)$ participate in $\mathcal{V}_{\mathsf{post}} \cup \mathcal{B}_{\mathsf{post}}$. For a variable $x \in \ell^{-1}(v)$, the longest dependency chain is that $x \in \mathsf{vbl}(\mathcal{E}_1)$, where \mathcal{E}_1 cause a retraction of x in the second round of retractions, and \mathcal{E}_1 cause the retraction, as some other event \mathcal{E}_2 retracted a variable $y \in \mathsf{vbl}(\mathcal{E}_1)$ because \mathcal{E}_2 was not avoided under the initial assignment φ of its variables $\mathsf{vbl}(\mathcal{E}_2)$. Let $z \in \mathsf{vbl}(\mathcal{E}_2)$. Then, the event $(v \in W)$ may depend on the randomness

of z, that is, the largest distance is at most $\operatorname{dist}(v, \ell(z)) \leq 4\nu$. For an event \mathcal{E} a similar reasoning upper bounds the dependence to 5ν . For $v \in W'$, this distance increase by an additive ν by the definition of $W' = N^{\nu}(W)$.

To obtain the bounds in the lemma we use that $d \ge d_{\mathcal{V}} - 1$ as all events incident to a variable are dependent.

We obtain that w.h.p. each connected component in the dependency graph of the LLL in the post-shattering graph contains few events, that is, the connected components of $H[\mathcal{B}_{post}]$ are small. This is sufficient for the post-shattering in the LOCAL model as small locality ν implies that we can simulate any algorithm on these small components efficiently and in parallel in G.

In the CONGEST model, we require stronger guarantees. Observe, that in an LLL instance with load l a vertex can simulate up to l events/variables and it may be that these are not dependent on each other. So, even though the components in the dependency graph of the LLL in the post-shattering phase may be small, it may be that their projection via ℓ to the communication network may result in huge connected components of G. For an efficient post-shattering phase, we require that also the projections to the communication network remain shattered, as we prove in Lemma 4.8.

Observe that two events in the dependency graph $H_{\mathcal{L}_{post}}$ of \mathcal{L}_{post} are connected by an edge if they share a variable in \mathcal{V}_{post} . So, $H_{\mathcal{L}_{post}} \subseteq H[\mathcal{B}_{post}]$ where $H_{\mathcal{L}_{post}}$ does not contain an edge $\{\mathcal{E}, \mathcal{E}\}$ of $H[\mathcal{B}_{post}]$ if all variables in $vbl(\mathcal{E}) \cap vbl(\mathcal{E}')$ already have a value in ψ_{pre} .

Lemma 4.7 (shattering). If $p \leq d^{-22}$ holds, then with high probability each connected component of $H_{\mathcal{L}_{post}} \subseteq H[\mathcal{B}_{post}]$ contains at most $O(d^8 \cdot \log n)$ nodes.

If $p < (d^2 \cdot d_{\mathcal{V}} \cdot l \cdot \Delta^{\nu})^{-1} \cdot \Delta^{-(4+12\nu)}$ holds, then with high probability the graph G[W'] consists of connected components of size $O(\log n \cdot \Delta^{6\nu})$.

Proof. By Lemma 4.6 and the shattering Lemma A.3 (with $c_3 = 2, c_2 = 4, c_1 = 21$), we obtain that with high probability in n the connected components of $H[\mathcal{B}_{post}]$ are of size at most $O(\log n \cdot d^8)$ if $p \leq d^{-22}$ holds. This high probability guarantee depends on the randomness of Step 1.

By Lemma 4.6 and the shattering Lemma A.3 (with $c_3 = 2, c_2 = 6\nu, c_1 = 4 + 24\nu$), we obtain that with high probability in *n* the connected components of G[W'] are of size at most $O(\log n \cdot \Delta^{12\nu})$ if $\Pr(v \in W') = p(d^2 \cdot (d+1) \cdot l \cdot \Delta^{\nu}) \leq \Delta^{-c_1}$ holds, only using randomness of Step 1.

Lemma 4.8. The projection of each connected component of $H[\mathcal{B}_{\mathcal{L}_{post}}]$ via ℓ , including all incident variables in $\mathcal{V}_{\mathcal{L}}$ of the events in $\mathcal{B}_{\mathcal{L}_{post}}$, is contained in a single connected component of G[W'].

Proof. Assume for contradiction that there are two nodes w_1, w_2 in different connected components of G[W'] that simulate dependent events $\mathcal{E}_1, \mathcal{E}_2$ of \mathcal{L}_{post} . By the definition of \mathcal{L}_{post} we obtain that $w_1, w_2 \in W$. Let $x \in \mathsf{vbl}(\mathcal{E}_1) \cap \mathsf{vbl}(\mathcal{E}_2)$. By the locality of ℓ , we obtain $d(w_i, \ell(x)) \leq \nu$ for i = 1, 2. But this would imply that $\ell(x)$ is in the same connected component of G[W'] as w_i for both i = 1 and i = 2, which is a contradiction. The same holds for all variables of these events.

Theorem 4.4 follows by plugging all lemmas in this section together, using that the LLL is simulatable for the algorithm's CONGEST implementation and verifying that the bound on the error probability p in Theorem 4.4 is sufficient to actually apply Lemma 4.7.

Proof of Theorem 4.4. Step 1 and the retractions can be implemented in $O(\nu)$ rounds in the LOCAL model. In the CONGEST model, Step 1 and the first round of retractions can be implemented in polylog log n rounds if (\mathcal{L}, G, ℓ) is simulatable. More detailed with the first primitive of Definition 2.3, node $\ell(\mathcal{E})$ can test whether $\operatorname{assoc}(\mathcal{E})$ holds (note that we extended the definition

of simulatability to associated events, see Definition 4.2) and via the second primitive respective nodes can send a *retract* command to the respective variables. the first and second primitive in). The second round of retractions is implemented as follows. First, each event determines whether it has a retracted variable by the aggregation primitive. Each retracted variable sends out a bit 0, each variable with a value $\neq \bot$ sends out the bit 1, and these are combined with the minimum operator. Thus, for each event $\mathcal{E} \in \mathcal{B}$ the node $\ell(\mathcal{E})$ can determine whether it has a retracted variable after the first round of retractions. If this is the case, the event sends the bit 0 to all its variables indicating that white variables should retract their value. Note that these commands can be combined (formally aggregated via the min operator) if a variable gets that retraction command from multiple of its events. With similar usage of these primitives for each event $\mathcal{E} \in \mathcal{B}$ the node $\ell(\mathcal{E})$ can determine whether the event \mathcal{E} should participate in the post-shattering phase, i.e., whether there is some $x \in vbl(\mathcal{E})$ with $\varphi_{pre}(x) = \bot$.

At the end of the algorithm, all events in \mathcal{B} are avoided under φ for the following reasons. Let $\mathcal{E} \in \mathcal{B}$ and let $S = \mathsf{vbl}(\mathcal{E}) \cap \mathcal{V}_{\mathsf{post}}$ the variables of the event whose assignment is computed in the post-shattering phase. If $S = \emptyset$, then $\mathsf{assoc}(\mathcal{E})$ is avoided under ψ_{pre} and we have $\psi(x) = \psi_{\mathsf{pre}}(x)$ for all $x \in \mathsf{vbl}(\mathcal{E})$. Then \mathcal{E} is avoided under φ as $\mathcal{E} \subseteq \mathsf{assoc}(\mathcal{E})$ holds. If $S \neq \emptyset$, the assignment of the variables in S in ψ_{post} together with ψ_{pre} avoids the event \mathcal{E} by the definition of the corresponding event in $\mathcal{B}_{\mathsf{post}}$.

Lemma 4.5 shows that \mathcal{L}_{post} indeed is an LLL with dependency degree d and upper bound p on the bad event probability.

The application of Theorem A.2 and Lemma 6.1 for solving \mathcal{L}_{post} in the post-shattering phase is almost identical to the one in the proof of Theorem 5.2. The respective lemmas need to be replaced with the lemmas of this section and Equation (6) needs to be replaced with

$$p < d^{-(4+c_l)-(4c+12c\nu)\log\log n} \le (d^2 \cdot (d+1) \cdot l)^{-1} \Delta^{-(4+12\nu)} , \tag{5}$$

which holds due to the bound on p from Theorem 4.4 and justifies that we can use Lemma 4.7. \Box

5 Disjoint Variable Set LLLs

Definition 5.1 (Disjoint variable set LLLs). Consider a set \mathcal{V} of random variables that is the union of two disjoint sets of variables \mathcal{V}_i , i = 1, 2, and some set of events \mathcal{E} defined over \mathcal{V} . This is a *disjoint variable set LLL* with bad event probability bounded by p if each event $\mathcal{E} = \mathcal{E}_1 \cap \mathcal{E}_2$ is the conjunction of two events $\mathcal{E}_1, \mathcal{E}_2$ where $\mathsf{vbl}(\mathcal{E}_i) = \mathcal{V}_i$ and $\Pr(\mathcal{E}_i) \leq p$ holds for i = 1, 2.

Note that for an event $\mathcal{E} = \mathcal{E}_1 \cap \mathcal{E}_2$ it is sufficient to avoid \mathcal{E}_1 or \mathcal{E}_2 to avoid the event \mathcal{E} . In the case of a disjoint variable set LLL in the CONGEST model, we extend the definition of simulatability and require that the event \mathcal{E}_1 can be tested in polyloglog n rounds on any partial assignment φ with $\perp \notin \varphi(\mathsf{vbl}(\mathcal{E}_1))$. The rest of this section is devoted to proving the following theorem.

Theorem 5.2. There are randomized LOCAL and CONGEST algorithms that in polylog log n rounds w.h.p. solve any <u>disjoint variable set LLL</u> of constant locality ν with dependency degree $d \leq \text{polylog } n$ and bad event upper bound p. The LOCAL algorithm requires $p < d^{-14}$ and the CONGEST algorithm requires $p < d^{-(2+c_l)-(4c+12c_{\Delta}\nu)\log\log n}$, $l \leq d^{c_l}$, $\Delta \leq \log^c n$ for constants $c_l, c_{\Delta} \geq 1$, and simulatability.

Our algorithm first samples all variables in \mathcal{V}_1 . Events not avoided by the assignment to their variables in \mathcal{V}_1 join the post-shattering phase together with their variables in \mathcal{V}_2 . The benefit is that connected components in the post-shattering are of polylogarithmic size which allows for faster

algorithms, both in the LOCAL model (Theorem A.2) and in the CONGEST model under certain stronger LLL criteria (Section 6).

Algorithm: Consider an LLL $\mathcal{L} = (\mathcal{V}, \mathcal{B}, p, d)$ as in Definition 5.1.

• Initial sampling (Step 1): Sample all variables in \mathcal{V}_1 according to their distribution.

Let ψ_{pre} be the resulting partial assignment.

- **Post-shattering:** Set up the following LLL instance $\mathcal{L}_{post} = (\mathcal{V}_{post}, \mathcal{B}_{post}, \ell)$ consisting of the following variables and (marginal) events.
 - Bad Events: $\mathcal{B}_{post} = \{ (\mathcal{E} | \psi_{pre}) : \mathcal{E} \in \mathcal{B}, \Pr(\mathcal{E} | \psi_{pre}) > 0 \}$
 - Variables: $\mathcal{V}_{post} = \mathcal{V}_2 \cap \bigcup_{\mathcal{E} \in \mathcal{B}_{post}} \mathsf{vbl}(\mathcal{E})$, with their respective original probability distribution. The variables in $\mathcal{V}_2 \setminus \mathcal{V}_{post}$ are not part of the postshattering phase; their values are set arbitrarily.
 - $-\ell_{\text{post}}(x) = \ell(x)$ for all $x \in \mathcal{V}_{\text{post}}$ and $\ell_{\text{post}}(\mathcal{E}|\psi_{\text{pre}}) = \ell(\mathcal{E})$ for all $(\mathcal{E}|\psi_{\text{pre}}) \in \mathcal{B}_{\text{post}}$. To simplify the notation, we refer to ℓ_{post} as ℓ .

We compute an assignment ψ_{post} of the variables in $\mathcal{V}_{\text{post}}$ that avoids all events in $\mathcal{B}_{\text{post}}$. In LOCAL this is done via Theorem A.2 and in CONGEST we use the algorithm from Lemma 6.1.

• Return $\psi = \psi_{\text{pre}} \cup \psi_{\text{post}}$.

Analysis of the post-shattering phase. We first show that the LLL formulation in the post-shattering phase indeed is an LLL. Afterwards, we show that with high probability the dependency graph of this LLL consists of small connected components. In the CONGEST model, we actually require a stronger statement, that is, we require that the projection of the post-shattering LLLs to the communication graph only consists of small components. To simplify the notation in the proofs, we identify the events $\mathcal{E}' = (\mathcal{E}|\psi_{pre})$ in \mathcal{B}_{post} with their corresponding event $\mathcal{E} \in \mathcal{B}$.

Lemma 5.3. \mathcal{L}_{post} is an LLL with bad event probability bound p and dependency degree d.

Proof. The dependency degree of \mathcal{L} is upper bounded by the d in Definition 5.1, and thus so is the dependency degree of \mathcal{L}_{post} . By Definition 5.1, we obtain $\Pr(\mathcal{E}') = \Pr(\mathcal{E} \mid \varphi_{pre}) \leq p$ for any event $\mathcal{E}' = (\mathcal{E} \mid \varphi_{pre})$ in \mathcal{B}_{post} .

Note that the statement in Lemma 5.3 holds "deterministically", that is, regardless of the outcome of the random choices in Step 1 of the algorithm.

Let $W = \{v \in V | \ell^{-1}(v) \cap (\mathcal{V}_{post} \cup \mathcal{B}_{post}) \neq \emptyset\}$ be the set of nodes that have one of their events/variables participating in the post-shattering phase. Further, let $W' = \{v \in V : \operatorname{dist}_G(v, W) \leq \nu\}$ be the set of nodes within distance ν from W.

We obtain the following bounds for events, variables, and nodes to be part of the post-shattering.

Lemma 5.4. The following bounds hold.

- For each event $\mathcal{E} \in \mathcal{B}$ we have $\Pr(\mathcal{E} \in \mathcal{B}_{post}) \leq p$,
- For each variable $x \in \mathcal{V}$ we have $\Pr(x \in \mathcal{V}_{post}) \leq p \cdot (d+1)$,
- For each node $v \in V(G)$ we have $\Pr(v \in W) \leq p \cdot (d+1) \cdot l$,

• For each node $v \in V(G)$ we have $Pr(v \in W') \leq p \cdot (d+1) \cdot l \cdot \Delta^{\nu}$.

For distinct \mathcal{E} and \mathcal{E}' the events $\mathcal{E} \in \mathcal{B}_{post}$ and $\mathcal{E}' \in \mathcal{B}_{post}$ are independent if $\operatorname{dist}_H(\mathcal{E}, \mathcal{E}') > 1$. For a node v the event whether it is contained in W or W' only depends on the randomness at nodes v' with $\operatorname{dist}_G(v, v') \leq 2\nu$ and $\operatorname{dist}_G(v, v') \leq 3\nu$, respectively.

Proof. Fix an arbitrary event $\mathcal{E} \in \mathcal{B}$. By Definition 5.1, the probability that \mathcal{E} is contained in \mathcal{B}_{post} is at most p. Fix an arbitrary variable $x \in \mathcal{V}$. In order for $x \in \mathcal{V}_{post}$ to hold, one of its $d_{\mathcal{V}}$ adjacent events needs to participate in \mathcal{B}_{post} . We obtain $\Pr(x \in \mathcal{V}_{post}) \leq p \cdot d_{\mathcal{V}}$ with a union bound over these events.

Fix an arbitrary node $v \in V$. For $v \in W$ to hold, one of its l(v) events/variables needs to be contained in $\mathcal{B}_{post} \cup \mathcal{V}_{post}$. We obtain $\Pr(v \in W) \leq p \cdot l(v) \cdot d_{\mathcal{V}} \leq p \cdot l \cdot d_{\mathcal{V}}$ with a union bound over all these events/variables. Similarly, $v \in W'$ only holds if there is a node $u \in N^{\nu}(v)$ with $u \in W$. We obtain $\Pr(v \in W') \leq p \cdot l \cdot d_{\mathcal{V}} \cdot \Delta^{\nu}$ with a union bound over the Δ^{ν} nodes in $N^{\nu}(v)$.

The event $(\mathcal{E} \in \mathcal{B}_{post})$ only depends on the random choices of the variables in $vbl(\mathcal{E}) \cap \mathcal{V}_1$. Thus, $(\mathcal{E} \in \mathcal{B}_{post})$ and $(\mathcal{E}' \in \mathcal{B}_{post})$ for events $\mathcal{E}, \mathcal{E}'$ depend on distinct sets of variables if $dist_H(\mathcal{E}, \mathcal{E}') > 1$.

For a node $v \in V$ the event $(v \in W)$ holds if one of the events/variables in $\ell^{-1}(v)$ participate in $\mathcal{V}_{\mathsf{post}} \cup \mathcal{B}_{\mathsf{post}}$. For a variable $x \in \ell^{-1}(v)$, this only depends on the outcome of all variables in $\mathsf{vbl}(\mathcal{E})$ for each event \mathcal{E} with $x \in \mathsf{vbl}(\mathcal{E})$. These variables are necessarily simulated by a node in distance at most 2ν . For an event $\mathcal{E} \in \ell^{-1}(v)$, whether the event $\mathcal{E} \in \mathcal{B}_{\mathsf{post}}$ holds only depends on the variables in $\mathsf{vbl}(\mathcal{E}) \cap \mathcal{V}_1$, which are contained in $N^{\nu}(v)$. For the last claim, this distance increases by an additive ν by the definition of $W' = N^{\nu}(W)$.

To obtain the bounds in the lemma we use that $d \ge d_{\mathcal{V}} - 1$ as all events incident to a variable are dependent.

For intuition regarding the next lemma, we refer to the text before Lemma 4.7. Observe that two events in the dependency graph $H_{\mathcal{L}_{post}}$ of \mathcal{L}_{post} are connected by an edge if they share a variable in \mathcal{V}_{post} . So, $H_{\mathcal{L}_{post}} \subseteq H[\mathcal{B}_{post}]$ holds.

Lemma 5.5 (shattering). If $p \leq d^{-14}$ holds, then with high probability each connected component of $H[\mathcal{B}_{post}]$ contains at most $O(d^4 \cdot \log n)$ nodes.

If $p < ((d+1) \cdot l)^{-1} \Delta^{-(4+12\nu)}$ holds, then with high probability the graph G[W'] consists of connected components of size $O(\log n \cdot \Delta^{6\nu})$.

Proof. By Lemma 5.4 and the shattering Lemma A.3 (with $c_3 = 2, c_2 = 1, c_1 = 9$), we obtain that with high probability in n the connected components of $H[\mathcal{B}_{post}]$ are of size at most $O(\log n \cdot d^2)$ if $p \leq d^{-14}$ holds. This with high probability guarantee depends on the randomness of Step 1.

By Lemma 5.4 and the shattering Lemma A.3 (with $c_3 = 2, c_2 = 3\nu, c_1 = 4 + 12\nu$), we obtain that with high probability in *n* the connected components of G[W'] are of size at most $O(\log n\Delta^{6\nu})$ if $\Pr(v \in W') \leq p((d+1) \cdot l \cdot \Delta^{\nu}) \leq \Delta^{-c_1}$ holds. This high probability guarantee depends on the randomness of Step 1.

The proof of the following lemma is identical to the proof of Lemma 4.8.

Lemma 5.6. The projection of each connected component of $H[\mathcal{B}_{\mathcal{L}_{post}}]$ via ℓ , including all incident variables in $\mathcal{V}_{\mathcal{L}}$ of the events in $\mathcal{B}_{\mathcal{L}_{post}}$, is contained in a single connected component of G[W'].

Proof of Theorem 5.2. First note that Step 1 can be executed in polylog log n rounds in LOCAL if the event/variable assignment (\mathcal{L}, G, ℓ) has constant locality. In the CONGEST model, it can be done in polylog log n rounds if the LLL is simulatable. Note, that each event $\mathcal{E} = \mathcal{E}_1 \cap \mathcal{E}_2$ can also test in poly log log n rounds whether \mathcal{E}_1 is avoided after Step 1 as for disjoint variable set LLLs this is required from the simulatability definition (see the line after Definition 5.1).

At the end of the algorithm each event $\mathcal{E} = \mathcal{E}_1 \cap \mathcal{E}_2 \in \mathcal{B}$ is avoided under φ for as either \mathcal{E}_1 is avoided under ψ_{pre} which agrees with φ , or \mathcal{E}_2 is avoided under ψ_{post} which also agrees with φ .

Lemma 5.3 shows that \mathcal{L}_{post} indeed is an LLL with dependency degree d and upper bound p on the bad event probability.

In the LOCAL model we solve $\mathcal{L}_{\text{post}}$ as follows. Lemma 5.5 shows that the connected components of $H[\mathcal{B}_{\text{post}}]$ are of size $O(d^4 \log n)$ if $p < d^{-14}$. As the dependency graph of $H_{\mathcal{L}_{\text{post}}}$ is a subgraph of $H[\mathcal{B}_{\text{post}}]$ its connected components are of the same size. Now, we can independently apply Theorem A.2 to solve each of these instances in parallel in polylog $N = \text{poly} \log \log n$, where each round of communication in $H_{\mathcal{L}_{\text{post}}}$ can be simulated in $O(\nu)$ rounds in the communication network G.

In the CONGEST model, we solve \mathcal{L}_{post} as follows. Formally, set up a new LLL \mathcal{L}'_{post} in which we append \mathcal{L}_{post} by the variables $\mathcal{V}_{post,passive} = \bigcup_{\mathcal{E} \in \mathcal{B}_{post}} \mathsf{vbl}(\mathcal{E}) \setminus \mathcal{V}_{post}$. Also, we set $\ell(x) = \ell(x)$ for all $x \in \mathcal{V}_{post,passive}$. Note that all these variables already have an assignment in ψ_{pre} . Due to $\Delta \leq \log^c n, l \leq d^{c_l}, d_{\mathcal{V}} \leq d$ and the condition on p in Theorem 5.2, we obtain

$$p < d^{-(2+c_l) - (4c+12c\nu)\log\log n} \le ((d+1) \cdot l)^{-1} \Delta^{-(4+12\nu)} .$$
(6)

In other words, the conditions of Lemma 5.5 are met for sufficiently large n. Thus, the lemma shows that the connected components of G[W'] are of size $N = O(\log n\Delta^{6\nu}) = O(\log^{6c\nu+1} n)$ and the same holds for each connected component of $H[\mathcal{B}_{post}]$. By Lemma 5.6, we also have that the projection via ℓ of each connected component of $H[\mathcal{B}_{post}]$ is contained in a connected component of G[W'] (this includes the assignment of $\mathcal{V}_{post,passive}$ to nodes in G). Hence, we can apply Lemma 6.1 in parallel on all components of \mathcal{L}'_{post} with the partial assignment ψ_{pre} restricted to variables incident to events in \mathcal{B}_{post} . This solves \mathcal{L}'_{post} in poly log N = poly log log n rounds and be correct with probability at least $1-2^{\text{bandwidth}} = 1-n^{-2}$, if bandwidth = $2\log n$. In the application of Lemma 6.1, we set $\lambda = 10 \log \log n$, which satisfies the condition on the LLL criterion in Lemma 6.1 as $p < d^{-(2+c_l)-(4c+12c\nu)\log\log n} \le p^{-\lambda}$. As $\lambda = \Omega(\log N)$, the algorithm runs in poly log log n rounds by Corollary 6.2. This also solves \mathcal{L}_{post} .

6 Efficient Post-shattering in CONGEST

In this section, we devise CONGEST algorithms for the LLL subinstances to be solved after shattering certain LLLs.

Known solutions for LLL in the CONGEST model cannot make use of the small component size unless d and Δ are at most poly log log n [MU21, HMN22]. Thus, we develop a novel algorithm to efficiently solve LLLs on small components in poly log log n rounds. Recall that, intuitively, an LLL instance is simulatable if one can (a) evaluate the status of each event in poly log log n rounds (note the variables of an event might be simulated by a node that is a few hops from the node that simulates the event), and (b) nodes can determine (in poly log log n rounds) the influence on their simulated bad events by partial assignments of variables. There are additional conditions to the definition of simulatability taking into account some allowed pre-processing and specifying the bandwidth that is allowed for the respective steps.

The goal of this section is to prove the following lemma that we need in the post-shattering phases of our algorithms.

Lemma 6.1. Let $\lambda > 0$ be a (possibly non-constant) parameter. There is a CONGEST(bandwidth) algorithm for LLL instances $\mathcal{L} = (\mathcal{V}, \mathcal{B}, \ell)$ with dependency degree is d and for which $\Pr(\mathcal{E} \mid \psi) < d^{-\lambda}$

holds for the marginal error probability of all events $\mathcal{E} \in \mathcal{B}$ where ψ can be any partial assignment of the variables in \mathcal{V} .

The algorithm requires that the locality ν of the LLL instances is constant and that each connected component of $G[N^{\nu}(\ell(\mathcal{V} \cup \mathcal{B}))]$ is of size at most N. It works with an ID space that is exponential in N. It requires that the LLL instance is simulatable and runs in polylog log $n \cdot \log(N \cdot l) + N^{2/\lambda} \cdot \operatorname{poly} \log N$ rounds where l is the load of the LLL. The error probability is upper bounded by $2^{-\text{bandwidth}}$.

Corollary 6.2. If $\lambda = \Omega(\log N)$, $x \leq \operatorname{poly} \log \log n$, $l \leq \operatorname{poly} \log \log n$, and bandwidth $= \Omega(\log n)$, the instances as in Lemma 6.1 can be solved in poly $\log \log n$ rounds with high probability in n.

6.1 Network Decomposition

A weak distance-k (C, β) -network decomposition with congestion κ is a partition of the vertex set of a graph into clusters C_1, \ldots, C_p of (weak) diameter $\leq \beta$, together with a color from [C] assigned to each cluster such that clusters with the same color are further than k hops apart. Additionally, each cluster has a communication backbone, a Steiner tree of radius $\leq \beta$, and each edge of G is used in at most κ backbones. For additional information on such decompositions, we refer the reader to [MU21, GGR21]. For the sake of our proofs, we only require that such decompositions can be computed efficiently (Theorem 6.3) and that one can efficiently aggregate information in all clusters of the same color in parallel in time that is essentially proportional to the diameter β (see Lemma A.4 for the precise statement).

Theorem 6.3 ([MU21]). For any (possibly non constant) $k \ge 1$ and any $\lambda \le \log N$ there is a deterministic CONGEST algorithm that, given a graph G with at most N nodes and unique IDs from an exponential ID space, computes a weak $(\lambda, k \cdot N^{1/\lambda} \log^3 n)$ -network decomposition of G^k with congestion $\kappa = O(\log N \cdot \min\{k, N^{1/\lambda} \cdot \log^2 N\})$ in $O(k \cdot N^{2/\lambda} \cdot \lambda \cdot \log^6 N \cdot \min\{k, N^{1/\lambda} \log^2 N\})$ rounds.

While we state our results in terms of general λ , the most natural invocation of Theorem 6.3 is $\lambda = \log N$ and constant k, in which case it computes a weak $(\log N, \log^3 N)$ -network decomposition of G^k with congestion $O(\log N)$ in $O(\log^7 N)$ rounds.

There are algorithms to compute similar network decompositions that are more efficient than the one in Theorem 6.3, e.g., in [GGH⁺23, MPU23]. However, as stated in their results, they require $\Omega(\log N \cdot \log \log N) = \omega(\log N)$ colors, and as the number of colors factors into the LLL criterion of the instances we can solve and for the sake of simplicity, we refrain of using their algorithms in a black box manner.

6.2 The LLL algorithm of Chung, Pettie, Su

In the LOCAL model there is a simple $O(\log n)$ -round randomized distributed algorithm for LLLs with $epd^2 < 1$ and constant locality [CPS17].

It is important for our paper that Algorithm 1 can be implemented in the CONGEST model if natural primitives are available. These primitives are the evaluation of events and the computation of local ID-minima (in the dependency graph) of a subset of the events for an arbitrary ID assignment (that may even be adversarial). The algorithm is used as a subroutine in our post-shattering solution.

Theorem 6.4 ([CPS17]). Suppose Algorithm 1 is run for $O(\log_{1/epd^2} |\mathcal{B}|)$ iterations on an LLL $\mathcal{L} = (\mathcal{V}, \mathcal{B})$ satisfying $epd^2 < 1$. Then, with probability at least $1 - 1/|\mathcal{B}|$, it computes an assignment

Initialize a random assignment of the variables Let \mathcal{F} be the set of bad events under the initial assignment **While** $\mathcal{F} \neq \emptyset$ Let $I = \{A \in \mathcal{F} : ID(A) = \min\{ID(B) | B \in N_{\mathcal{H}_{\mathcal{L}}}(A)\}\}$ Resample $\mathsf{vbl}(I) = \bigcup_{A \in I} \mathsf{vbl}(A)$ Let \mathcal{F} be the set of bad events under the current assignment

that avoids all bad events. The algorithm requires events to be equipped with identifiers that are unique within their connected component of the dependency graph.

6.3 Efficient Post-shattering in CONGEST (details)

To devise an efficient CONGEST post-shattering algorithm, we decompose each small component into small clusters via the network decomposition algorithm from Theorem 6.3. Then, the objective is to iterate through the collections of the decomposition and when processing a cluster we want to fix all variables in that cluster. When doing so we need to ensure that the influence on (neighboring) clusters processed in later stages of the algorithm is not (too) negative. To efficiently measure and limit this effect to a d^2 -factor increase in the marginal probability of each remaining event, we set up a new LLL for each cluster that ensures just that condition. An application of Markov's inequality shows that the probability of the marginal probability to increase more than a factor d^2 if a subset of the variables of an event is sampled is at most $1/d^2$ (see Claim 6.5). Implementing all required primitives efficiently needs the simulatability of the LLL.

In Sections 4 and 5, we want to use Lemma 6.1 to exploit that the post-shattering phase consists of several small connected components.

Proof of Lemma 6.1. First, we use Theorem 6.3 to compute a network decomposition of the communication network G where the distance between two clusters of the same color is strictly larger than 2ν (recall, that $\nu = O(1)$ is the locality of event/variable assignment ℓ). We instantiate the theorem such that we obtain λ collections, weak diameter $O(N^{1/\lambda} \log^3 N)$ and congestion $O(\log N)$. The algorithm runs in $O(\lambda \cdot N^{2/\lambda} \log^6 N)$ rounds as ν is constant. This is obtained by running the algorithm on all connected components of $G[N^{\nu}(\ell(\mathcal{V} \cup \mathcal{B}))]$ in parallel.

To compute an assignment of \mathcal{V} that avoids all events in \mathcal{B} , we start with an initial partial assignment φ_0 with all variables unassigned and then iterate through the λ collections of the network decomposition. When processing the nodes V_i in collection *i*, we permanently assign the variables in $\mathcal{V}_i = \mathcal{V} \cap \ell^{-1}(V_i)$. Let φ_i be the partial assignment after processing the *i*-th collection. The invariant that we maintain for each event $\mathcal{E} \in \mathcal{B}$ is that after processing the *i*-th collection we have $\Pr(\mathcal{E} \mid \varphi_i) \leq p \cdot d^{2i}$. Initially, the invariant holds for $\varphi_0 = \psi$ (ψ from the lemma statement) by the upper bound *p* on bad event probabilities of events in \mathcal{B} . Let $\varphi = \varphi_{\lambda}$ be the final assignment in which each variable of \mathcal{V} has a value $\neq \bot$. We obtain that $\Pr(\mathcal{E} \mid \varphi) \leq pd^{2\lambda} < 1$, since $p < d^{-2\lambda}$. Since all variables have been fixed by φ , this means that the event \mathcal{E} is avoided under φ .

Next, we detail the process for a single collection, that is, how to find the partial assignment φ_i , given φ_{i-1} . Note that throughout the algorithm all partial assignments are known in a distributed manner, that is, for a variable $x \in \mathcal{V}$, the node $\ell(x)$ knows the value of x (or \perp) in the current partial assignment. Also observe that for each variable $x \in \mathcal{V}$ node $\ell(x)$ knows whether $x \in \mathcal{V}_i$ holds. Let \mathcal{B}_i be the set of events \mathcal{E} with $\mathsf{vbl}(\mathcal{E}) \cap \mathcal{V}_i \neq \emptyset$. By using the aggregation primitive (here,

we only require a broadcast) from the variables in \mathcal{V}_i to the events in \mathcal{B}_i nodes learn whether their events are contained in \mathcal{B}_i . We define the following LLL \mathcal{L}_i for collection *i*.

- $\mathcal{V}_{\mathcal{L}_i}$ set of variables: \mathcal{V}_i with their original distribution.
- $\mathcal{B}_{\mathcal{L}_i}$ set of bad events: For each $\mathcal{E} \in \mathcal{B}_i$ there is an event \mathcal{E}' that holds on an assignment ψ of \mathcal{V}_i if

$$\Pr(\mathcal{E} \mid \psi \cup \varphi_{i-1}) \ge d^2 \Pr(\mathcal{E} \mid \varphi_{i-1}),$$

• $\ell_{\mathcal{L}_i}(x) = \ell(x)$ for all $x \in \mathcal{V}_{\mathcal{L}_i}$ and $\ell_{\mathcal{L}_i}(\mathcal{E}') = \ell(\mathcal{E})$ for all $\mathcal{E}' \in \mathcal{B}_{\mathcal{L}_i}$.

Claim 6.5. \mathcal{L}_i is an LLL with error probability at most $1/d^2$ and dependency degree d.

Proof. Consider an event $\mathcal{E}' \in \mathcal{B}_{\mathcal{L}_i}$ and let $\mathcal{E} \in \mathcal{B}_{\mathcal{L}}$ be the corresponding event of \mathcal{L} . For an assignment ψ of $\mathcal{V}_{\mathcal{L}_i}$, let $p_{\psi} = \Pr(\mathcal{E} \mid \psi \cup \varphi_{i-1})$. Note that formally p_{ψ} is a random variable over the randomness of the variables in $\mathcal{V}_{\mathcal{L}_i}$. For each assignment of these variables p_{ψ} is a value in [0, 1]. Hence, we obtain $E_{\mathcal{V}_{\mathcal{L}_i}}[p_{\psi}] = \Pr(\mathcal{E} \mid \psi \cup \varphi_{i-1}) \eqqcolon \mu$, where the subscript indicates that the randomness of the expectation is only for the variables in $\mathcal{V}_{\mathcal{L}_i}$. By Markov inequality, we have $\Pr(\mathcal{E}') = \Pr(p_{\psi} \ge d^2\mu) \le 1/d^2$.

We use the following claim (sometimes implicitly) throughout the proof.

Claim 6.6. Each connected component of the dependency graph of $H_{\mathcal{L}_i}$ is contained in a connected component of $G[N^{\nu}(\ell(\mathcal{V} \cup \mathcal{B}))]$.

Now, we run k = bandwidth parallel instances of the LLL algorithm of [CPS17] with the LLL \mathcal{L}_i . Each instance runs for $O(\log(N \cdot l))$ iterations (note that each connected component of $G^{\nu}[\ell(\mathcal{V} \cup \mathcal{B})]$ contains at most $N \cdot l$ events); the precise discussion of the total runtime of executing these instances and the remaining parts of the algorithm is deferred to the end of the proof. As a result we obtain k assignments ψ_1, \ldots, ψ_k of \mathcal{V}_i . For $j \in [k]$, we say that an assignment ψ_j is *correct* for an event $\mathcal{E}' \in \mathcal{B}_{\mathcal{L}_i}$ if \mathcal{E}' is avoided under φ_j .

To define φ_i from the assignments ψ_1, \ldots, ψ_k , let us define the following notation for each cluster \mathcal{C} with color i in the network decomposition. Let $\mathcal{V}_{\mathcal{C}} = \mathcal{V}_i \cap \ell^{-1}(\mathcal{C})$ and $\mathcal{B}_{\mathcal{C}} = \{\mathcal{E} \in \mathcal{B}_i : \mathsf{vbl}(\mathcal{E}) \cap \mathcal{V}_{\mathcal{C}} \neq \emptyset\}$. For two distinct clusters \mathcal{C} and \mathcal{C}' of the *i*-th collection we obtain that $\mathcal{V}_{\mathcal{C}} \cap \mathcal{V}_{\mathcal{C}'} = \emptyset$ and $\mathcal{B}_{\mathcal{C}} \cap \mathcal{B}_{\mathcal{C}'} = \emptyset$ as the cluster separation is strictly larger than 2ν . By the properties of Algorithm 1 (and using Claim 6.6), each ψ_j is correct for all events in $\mathcal{B}_{\mathcal{C}}$ with probability $\geq (1 - 1/(N \cdot l)) \geq 1/2$. Thus, with probability at least $1 - 1/2^k$ there is an $j_{\mathcal{C}}^* \in [k]$ such that $\psi_{j_{\mathcal{C}}^*}$ is correct for all events of $\mathcal{B}_{\mathcal{C}}$. Each node holding an event of \mathcal{L}_i determines which assignments are correct for its event. For each cluster \mathcal{C} in parallel the nodes in $\ell_{\mathcal{L}_i}(\mathcal{B}_{\mathcal{C}})$ agree on an index $j_{\mathcal{C}}^*$ such that assignment $\psi_{j_{\mathcal{C}}^*}$ is correct for all events in $\mathcal{B}_{\mathcal{C}}$; let $\tilde{\psi}_{j_{\mathcal{C}}^*}$ denote the restriction of this assignment to $\mathcal{V}_{\mathcal{C}}$. Different clusters may decide on different indices. Lastly, we set $\varphi_i = \varphi_{i-1} \cup \bigcup_{\mathcal{C} has color i} \tilde{\psi}_{j_{\mathcal{C}}^*}$. The partial assignment ψ_i is well-defined as each variable of \mathcal{V}_i appears in exactly one cluster.

Claim 6.7. The invariant $\Pr(\mathcal{E} \mid \varphi_i) \leq p \cdot d^{2i}$ holds.

Proof. For each event $\mathcal{E} \notin \mathcal{B}_i$, the claim follows as none of the variables in $\mathsf{vbl}(\mathcal{E})$ change their value when processing the *i*-th collection, that is, we obtain $\Pr(\mathcal{E} \mid \varphi_i) = \Pr(\mathcal{E} \mid \varphi_{i-1}) \leq p \cdot d^{2(i-1)} \leq p \cdot d^{2i}$.

For each event $\mathcal{E} \in \mathcal{B}_i$, there is a cluster \mathcal{C} with color *i* for which $\mathcal{E} \in \mathcal{B}_{\mathcal{C}}$. The partial assignment $\tilde{\psi}_{j^*_{\mathcal{C}}}$ avoids all events in $\mathcal{B}_{\mathcal{C}}$. As $\Pr(\mathcal{E} \mid \varphi_{i-1}) \leq p \cdot d^{2(i-1)}$ holds, the definition of the avoided events in $\mathcal{B}_{\mathcal{C}} \subseteq \mathcal{B}_i$ implies $\Pr(\mathcal{E} \mid \psi \cup \varphi_{i-1}) \geq d^2 \Pr(\mathcal{E} \mid \varphi_{i-1}) \leq p d^{2i}$.

Before we run the k parallel instances of Algorithm 1, we compute a locally unique ID $\iota(v) \in \{1, \ldots, N\}$ for each node $v \in \ell_{\mathcal{L}_i}(\mathcal{B}_i)$, for each cluster \mathcal{C} of color *i*. Then, for each event $\mathcal{E} \in \mathcal{B}_i$ assign a locally unique ID $\iota(\mathcal{E}) = (\iota(\ell_{\mathcal{L}_i}(\mathcal{E}), j) \in \{1, \ldots, N\}^2$ where *j* is the index of \mathcal{E} among the events simulated by node $\iota(\ell_{\mathcal{L}_i}(\mathcal{E}))$ (according to an arbitrary ordering of these events). These IDs are used to compute the local minima of non-avoided events in the simulation of Algorithm 1. As the distance between clusters of the same color is strictly larger than 2ν , these IDs are unique within the connected components of the dependency graph of $H_{\mathcal{L}_i}$, which tailors them sufficiently for the simulation of Algorithm 1. Assigning these locally unique IDs can be done in $O(N^{1/\lambda} \operatorname{poly} \log N)$ rounds by using the Steiner tree of the clusters.

We next, bound the runtime of the whole process and show that all steps can be executed. To improve the readability let $x = poly \log \log n$ be the number of rounds from Definition 2.3. Recall, that nodes know whether their simulated variables and events are contained in \mathcal{V}_i and \mathcal{B}_i , respectively. As the LLL \mathcal{L} is simulatable, we only need to send $k \cdot x$ bits to run one round of each of the k instances of [CPS17]. In more detail, as the new IDs that we computed are represented with bit strings of length $O(\log N)$, the pre-processing phase of Definition 2.3 requires $x \cdot \mathsf{IDbitLength} = x \cdot \mathsf{poly} \log N$ rounds of CONGEST(bandwidth). Then Algorithm 1 applied to \mathcal{L}_i can be implemented with the operations in the second part of Definition 2.3. More detailed, in x rounds, for each event \mathcal{E} , the corresponding nodes can determine whether $\Pr(\mathcal{E} \mid \psi \cup \varphi_{i-1}) \geq d^2 \Pr(\mathcal{E} \mid \varphi_{i-1})$ holds or not, where ψ is an assignment of the variables in \mathcal{V}_i . Also, we can compute a set of local minimum IDs I (in the dependency graph $H_{\mathcal{L}_i}$) of events \mathcal{E} for which this is not the case by using the broadcast and aggregation primitives. Let Z be the set of non-avoided events in some iteration. Each event $\mathcal{E} \in \mathbb{Z}$ broadcasts its locally unique ID $\iota(\mathcal{E})$ to all variables in $\mathsf{vbl}(\mathcal{E}) \cap \mathcal{V}_i$. For a variable $x \in \mathcal{V}_i$ let $c(x) = \min_{\mathcal{E} \in \mathbb{Z}} \iota(\mathcal{E})$. Then the variables use the aggregation primitive to send the smallest ID that they have received to the events that contain them, that is, each event ${\cal E}$ receives $c(\mathcal{E}) = \min_{x \in \mathsf{vbl}(\mathcal{E}) \cap \mathcal{V}_i} c(x)$. The set $I = \{\mathcal{E} : c(\mathcal{E}) = \iota(\mathcal{E})\}$ is consists of locally minimum IDs of $H_{\mathcal{L}_i}[Z]$, as required. Lastly, events in I use the broadcasting function to inform their respective variables such that they can re-sample themselves for the next iteration of Algorithm 1.

Thus, simulating one round of all k instances of Algorithm 1 in parallel requires $O(k \cdot x/\text{bandwidth}) = x$ rounds as bandwidth = k. Hence, running all instances for $O(\log N)$ rounds takes only $x \cdot \log(N \cdot l)$ rounds. The simulatability also implies that nodes in $\ell_{\mathcal{L}_i}(\mathcal{B}_{\mathcal{L}_i})$ can check which events hold in which of the k assignments ψ_1, \ldots, ψ_k in x rounds.

Agreeing on the index $j_{\mathcal{C}}^*$ in cluster \mathcal{C} can be done efficiently as follows: Each node $v \in \ell_{\mathcal{L}_i}(\mathcal{B}_{\mathcal{C}})$ holds a bit string of length k = bandwidth in which the *j*-th bit equals 1 if and only if all its events, i.e., the events in $\mathcal{B}_i \cap \ell_{\mathcal{L}_i}^{-1}(v)$, are avoided in ψ_j . All nodes in $\ell_{\mathcal{L}_i}(\mathcal{B}_{\mathcal{C}})$ agree on the index $j_{\mathcal{C}}^*$ in time linear in the cluster's weak diameter (ignoring congestion between different clusters for now) by computing a bitwise-AND of the bitstrings.

When simulating the k instances of Algorithm 1, there is no congestion between clusters that are processed simultaneously as their distance in the graph is strictly greater than ν . Agreeing on the assignment of the variables within one cluster is done on the Steiner tree of the cluster, where each edge is contained in at most one tree of clusters of the same color. Since also the clusters' weak diameter is of size poly log N and as we have $O(\log N)$ colors classes the whole process runs in $x \cdot \log(N \cdot l) + \operatorname{poly} \log N$ rounds, and the claim follows as $x = \operatorname{poly} \log \log n$.

7 Applications and Bounding Risks

Section 7.1 serves as a warm-up. We sketch how to use our disjoint variable set LLL in order to solve the slack generation problem. As this assumes that we have already solved the DSS problem,

the formal statement and proof appear are deferred to Section 8. In Section 7.2, we present several techniques to bound the risk events and show that our framework solve all LLLs that can be solved with the main LLL algorithm of [GHK18]. In Section 7.3 we bound the risk of several types of binary LLLs and show that all LLLs that can be solved with the main LLL algorithm of [GHK18] can be solved with out framework (in the LOCAL model).

7.1 Example of Disjoint Variable Set LLL: Slack Generation

Let us illustrate the usefulness of disjoint variable set LLLs by having a closer look at one example that is important for our coloring results. Recall that the goal is to color some of the nodes of the graph such that each node v receives some *slack*, that is, (at least) two of its neighbors $w, w' \in N(v)$ are colored with the same color. A simple randomized algorithm to generate slack for such nodes is to activate each node with a constant probability and then activated nodes select a random candidate color. Then, the candidate colors are exchanged with their neighbors and a node gets permanently colored with its candidate color if no neighbor tried the same color.

Algorithm 2 SLACKGENERATION

Input: $S \subseteq V$

- 1: Each node in $v \in S$ is active w.p. 1/20
- 2: Each active node v samples a color r_v u.a.r. from $[\chi]$.
- 3: v keeps the color r_v if no neighbor tried the same color.

In Section 8, we prove the following lemma that bounds the probability that this process provides nodes with slack. Variants of this result have appeared numerous times [MR13, EPS15, HKMT21]

Lemma 7.1 (Slack generation, Lemma 8.6 simplified). Let Δ_s, χ be positive integers with $\chi \geq \Delta_s/10$. Let $S \subset V$ be a subset of nodes. Consider a node $v \in V$ with at least \overline{m} non-edges in $G[N(v) \cap S]$. Suppose that all nodes in S, as well as v, have at most Δ_s neighbors in S. After running SlackGeneration on S with color palette $[\chi]$, the slack of v is increased by $\Omega(\overline{m}/\chi)$, with probability at least $1 - \exp(-\Omega(\overline{m}/\chi))$. This holds independent of random choices at distance more than 2.

Lemma 7.1 immediately gives rise to an LLL for $\chi = \Delta$. Introduce a bad event \mathcal{E}_v for each node v of a graph that holds if v does not obtain slack. Then, for sparse Δ -regular graphs, that is, graphs in which any node has $\Omega(\Delta^2)$ non-edges in its neighborhood the probability that \mathcal{E}_v holds is upper bounded by $p = \exp(-\Omega(\Delta))$ and it only shares randomness with $d = \Delta^4$ events of other nodes.

Consider a version of the slack generation problem where we are given two sets $S_1, S_2 \subseteq V$, such that each node v has many non-edges in the graph induced by $N(v) \cap S_1$ and also in the graph induced by $N(v) \cap S_2$. Observe that we can compute such sets by solving the DSS problem.

Given these sets, we introduce the event $\mathcal{E}_v = \mathcal{E}_{v,1} \cap \mathcal{E}_{v,2}$ for each node v where for i = 1, 2the event $\mathcal{E}_{v,i}$ is avoided if v receives slack from the coloring of the nodes in S_i . So, translating our algorithm for such LLLs into the language of this slack generation problem, we first execute SlackGeneration for all nodes in S_1 . Then, each node v that did not get slack yet, together with its neighbors in S_2 goes to the post-shattering phase. Here, we exploit that connected components are small and solve the slack generation problem for these nodes faster, e.g., by using Theorem A.2 in the LOCAL model or using Lemma 6.1 in the CONGEST model.

7.2 Techniques to Bound Risk

The objective of this section is to illustrate techniques for bounding risk. to show that all LLLs that can be solved with the main LLL algorithm of [GHK18] can be solved with our framework. This section does not reason simulatability for an efficient CONGEST implementation, which is done elsewhere whenever these events are needed.

Often we are just interested in the variables whose value is **black** and we sometimes abuse language and speak of these as *sampled* variables/nodes; hence we also speak of a *sampling LLL*. We illustrate in this section a number of natural sampling problems that have LLL with low risk and can therefore be handled with our method. As we can modify the domain and the sampling, and we can combine criteria, this induces a space of solvable problems.

A property that appears frequently in sampling LLLs is that its bad events are monotone in the sense that they either profit from having more nodes set to black or white respectively. Examples are the events in the degree-bounded subgraph sampling and in the DSS problem discussed in the introduction. Formally, monotone events are captured by the following definition.

Definition 7.2 (monotone events). An event \mathcal{E} defined over a set of independent binary random variables is color-*favoring* for color \in {black, white} if $\Pr(\mathcal{E} \mid \psi) \leq \Pr(\mathcal{E} \mid \varphi)$ holds for any ψ, φ with $\psi(x) \leq \varphi(x)$ for each $x \in \mathsf{vbl}(\mathcal{E})$ (where color $< \bot < \overline{\mathsf{color}}$).

We also call an event *monotone increasing* (*monotone decreasing*) if it is black-favoring (white-favoring). The following lemma is one of the core advantages of our binary LLL solver, compared to using prior LLL algorithms (even when used in the LOCAL model).

Lemma 7.3 (No Risk Lemma). The risk of a monotone increasing event \mathcal{E} is $Pr(\mathcal{E})$ testified by $assoc(\mathcal{E}) = \mathcal{E}$.

Proof. Let \mathcal{E} be a monotone-increasing event. Then \mathcal{E} itself testifies that the risk of \mathcal{E} is at most p. First, we trivially have $\Pr(\mathcal{E}) \leq p$. We need to show that $\max_{\psi \in \mathsf{Respect}(\mathcal{E})} \{\Pr(\mathcal{E} \mid \psi)\} \leq p$. Let $\psi \in \mathsf{Respect}(\mathcal{E})$ be any promise retraction and let φ be a full assignment corresponding to ψ . By definition of $\mathsf{Respect}(\mathcal{E})$, either all white variables are retracted ($\varphi(x) \in \{\mathsf{black}, \bot\}$), or no black variables are retracted. In the first case, $\psi(x) \leq \bot$ for all $x \in \mathsf{vbl}(\mathcal{E})$, hence $\Pr(\mathcal{E} \mid \psi) \leq \Pr(\mathcal{E}) \leq p$. If no black variables are retracted, we have $\psi(x) \leq \varphi(x)$ for all $x \in \mathsf{vbl}(\mathcal{E})$. Hence, $\Pr(\mathcal{E} \mid \psi) \leq \Pr(\mathcal{E} \mid \varphi) = 0$, where the last equality follows from the fact that φ does not satisfy \mathcal{E} by definition of $\mathsf{Respect}(\mathcal{E})$.

The issue is that LLLs that only consist of monotone increasing events are trivial (one can simply set all variables to **black** to solve them) and the asymmetry in the aforementioned approach (we only undo white variables in the second step of retractions) prevents us from dealing with monotone decreasing events in the same manner simultaneously. Next, we present a general method for bounding the risk, mostly originating from prior work (and not used in our applications).

Bounding the risk of general events. For completeness and to compare with prior work, we first explain how the risk of an event can be bounded by the analysis of LLL process given in prior work. The next few definitions and lemmas are reformulated in the language of this paper but appear in a similar manner in [GHK18] and slightly differently already in [CPS17]. Afterward, we discuss why these are helpful in general but unsuitable in the context of our efficient CONGEST algorithms.

Consider some event \mathcal{E} defined over some random variables $\mathsf{vbl}(\mathcal{E})$. Let q be a parameter. A partial assignment φ of $\mathsf{vbl}(\mathcal{E})$ is q-dangerous if there exists some retraction ψ of φ such that

$$\Pr(\mathcal{E} \mid \psi) > q \tag{7}$$

holds. Let \mathcal{E}_{danger}^q be the event that holds on all assignments that are q-dangerous. The following lemma follows immediately from the definition of risk and \mathcal{E}_{danger} .

Lemma 7.4. The risk of any event \mathcal{E} is upper bounded by $\max\{q, \Pr(\mathcal{E}_{danger}^q)\}$ testified by \mathcal{E}_{danger}^q .

As a result of Lemma 7.4 (applied with a suitable $q = 1/\text{poly }\Delta$), any LLL that can be solved with the main LLL algorithm of [GHK18, Section 6] can also solved with our LLL solver in the LOCAL model. Our algorithm is advantageous whenever one cannot easily bound $\Pr(\mathcal{E}_{danger}^q)$, but can instead use Lemma 7.3 to bound the risk.

In practice, it is highly non-trivial to derive upper bounds on $\Pr(\mathcal{E}_{danger}^q)$. As an additional tool Ghaffari, Kuhn, and Harris introduce another technical term, the *fragility* $f(\mathcal{E})$ of an event [GHK18, Definition 6.3].

Definition 7.5 (Fragility). Let \mathcal{E} be an event on variables $\mathsf{vbl}(\mathcal{E}) = \{x_1, \ldots, x_k\}$ and let φ_1, φ_2 be two partial assignments with actual values for $\mathsf{vbl}(\mathcal{E})$ and \bot for other variables. For any vector $a \in \{0,1\}^k$, define a new partial assignment ψ_a by $\psi_a(x_i) = \varphi_{a_i}(x_i)$ for $1 \le i \le k$ and $\psi_a(x_i) = \bot$ for $x_i \notin \mathsf{vbl}(\mathcal{E})$. Let \mathcal{E}_B be the event

$$\mathcal{E}_B = \bigvee_{a \in \{0,1\}^k} \mathcal{E} \text{ occurs on assignment } \psi_a$$

The fragility of \mathcal{E} , denoted $f(\mathcal{E})$, is the probability of \mathcal{E}_B when φ_1 and φ_2 are drawn independently, according to the distribution of the variables in $vbl(\mathcal{E})$.

In [GHK18, Proposition 6.4] they also show that if an event \mathcal{E} has fragility $f(\mathcal{E})$, then the probability that \mathcal{E}_{danger}^q holds is upper bounded by $f(\mathcal{E})/q$.

Lemma 7.6. The risk of an event \mathcal{E} is at most $\max(f(\mathcal{E})/q, q)$.

While these are powerful tools they come with two issues for our purposes: (1) the fragility and more general $\Pr(\mathcal{E}_{danger}^q)$ can be quite hard to analyze for involved LLL processes with multiple dependencies such as DSS or the slack generation LLL, and (2) the associated event $\operatorname{assoc}(\mathcal{E}) = \mathcal{E}_{danger}^q$ cannot, in general, be evaluated on an assignment in the CONGEST model, as the respective conditional probabilities of Equation (7) are unlikely to be computable without full information about all variables, the local graph structure in a graph problem, etc., none of which are readily available in the CONGEST model.

The prime monotone decreasing events that appear in the context of our CONGEST applications are events that control the maximum degree into some subgraph. In contrast to the involved lemmas above, it is easy to find a simple associated event. Alternatively, one could use a result from [GHK18, Theorem 6.8] to bound the fragility and hence via Lemma 7.6 the risk.

Lemma 7.7. Consider a random variable X that is a sum of independent binary random variables. For some threshold parameter x > 0, let \mathcal{E}_x be the event that X > x holds. Then, the risk of \mathcal{E}_x is at most $\Pr(\mathcal{E}_{x/2})$ testified by $\mathcal{E}_{x/2}$.

Proof. By definition, the risk testified by $\mathcal{E}_{x/2}$ is given by

$$\max\left\{\Pr(\mathcal{E}_{x/2}), \max_{\psi \in \mathsf{Respect}(\mathcal{E}_{x/2})} \{\Pr(\mathcal{E} \mid \psi)\}\right\}.$$

To prove the lemma, take an arbitrary assignment $\psi \in \text{Respect}(\mathcal{E}_{x/2})$. As ψ is a retraction of an assignment on which $\mathcal{E}_{x/2}$ was avoided at most x/2 of the variables in ψ are black. Let ψ' be the assignment obtained from ψ by setting all black variables to white. We obtain.

$$\Pr(\mathcal{E}_x \mid \psi) \le \Pr(\mathcal{E}_{x/2} \mid \psi') \le \Pr(\mathcal{E}_{x/2})$$

which proves the claim.

Note that monotone decreasing events, as in Lemma 7.7, do not rely on only white variables being retracted in the second round of retractions (in fact, they prefer the white variables to stay).

The following lemma is useful to obtain upper bounds on the risk of combined events.

Lemma 7.8. Let \mathcal{E}_1 and \mathcal{E}_2 be events defined over the same set of independent random variables, and suppose they have risk p_1 and p_2 , respectively. Then the risk of $\mathcal{E}_1 \cup \mathcal{E}_2$ is at most $p_1 + p_2$.

Proof. Let $\mathcal{E}'_1 = \operatorname{assoc}(\mathcal{E}_1)$, $\mathcal{E}'_2 = \operatorname{assoc}(\mathcal{E}_2)$ be events testifying the risk of \mathcal{E}_1 and \mathcal{E}_2 . Let $\mathcal{E}'_{1,2} = \mathcal{E}'_1 \cup \mathcal{E}'_2$. We shall show that the event $\mathcal{E}'_{1,2}$ testifies a risk of at most $p_1 + p_2$ for $\mathcal{E}_1 \cup \mathcal{E}_2$. Firstly, $\Pr[\mathcal{E}'_{1,2}] \leq \Pr[\mathcal{E}'_1] + \Pr[\mathcal{E}'_2] \leq p_1 + p_2$, by the union bound and the definition of risk. We need to bound $\max_{\psi \in \mathsf{Respect}(\mathcal{E}'_{1,2})} \Pr(\mathcal{E}_1 \cup \mathcal{E}_2 \mid \psi)$. Consider any $\psi \in \mathsf{Respect}(\mathcal{E}'_{1,2})$. Let $\hat{\varphi}$ be a full assignment that respects ψ and avoids $\mathcal{E}'_{1,2}$, which exists by the definition of retraction. Since $\hat{\varphi}$ avoids $\mathcal{E}'_{1,2} = \mathcal{E}'_1 \cup \mathcal{E}'_2$, it in particular avoids \mathcal{E}'_1 and \mathcal{E}'_2 . Hence, $\psi \in \mathsf{Retract}(\mathcal{E}'_1)$ and $\psi \in \mathsf{Retract}(\mathcal{E}'_2)$. Also, since ψ satisfies the promises, $\psi \in \mathsf{Respect}(\mathcal{E}'_1) \cap \mathsf{Respect}(\mathcal{E}'_2)$. Using this, we get $\Pr(\mathcal{E}_1 \cup \mathcal{E}_2 \mid \psi) \leq \Pr(\mathcal{E}_1 \mid \psi) + \Pr(\mathcal{E}_2 \mid \psi) \leq p_1 + p_2$, by the assumption on the risk of \mathcal{E}_1 and \mathcal{E}_2 . Hence, $\mathcal{E}'_{1,2}$ testifies the risk of at most $p_1 + p_2$ for the event $\mathcal{E}_1 \cup \mathcal{E}_2$.

7.3 Example LLLs with Low Risk

We illustrate here how we bound the risk of several types of natural binary LLLs. In each of these problems, we assume that each node is sampled independently with probability q.

Lemma 7.9. Consider the following LLLs. In all of them, each node is sampled independently with probability q, which induces a subgraph S. Let $S_v = N(v) \cap S$ be the sampled neighbors of v. Let $\ell, \overline{\ell}$ be such that the induced neighborhood graph G[N(v)] contains $\ell \cdot d(v)$ edges and thus $\overline{\ell}d(v) = (d(v) - \ell)d(v)/2$ non-edges. We bound the risk of the following events.

- 1. \mathcal{E}_v : $|S_v| \ge 3qd(v)$, i.e. v has more than 3qd(v) sampled neighbors in S.
- 2. $\mathcal{E}_v: |S_v| \leq qd(v)/2$, i.e. v has fewer than qd(v)/2 sampled neighbors in S.
- 3. \mathcal{E}_v : $G[S_v]$ has fewer than $q^2 \overline{\ell} d(v)/2$ non-edges.
- 4. \mathcal{E}_v : $G[S_v]$ has fewer than $q^2\ell d(v)/2$ edges.

Events (1) and (2) have risk at most $\exp(-\Omega(qd(v)))$. Event (3) has risk at most $\exp(-q\overline{\ell}/5)$ and (4) has risk at most $\exp(-q\ell/5)$.

- *Proof.* 1. Monotone decreasing event. We apply Lemma 7.7 to the variable $X_v = |S_v|$, with associated event \mathcal{E}'_v being that $|S_v| \ge 3qd(v)/2$. The risk is at most $\Pr[\mathcal{E}'_v]$, which by Chernoff is $\exp(-\Omega(qd(v)))$.
 - 2. Monotone increasing event for which we apply Lemma 7.3. The risk is at most $\Pr[\mathcal{E}_v]$, which by Chernoff is $\exp(-\Omega(qd(v)))$.
 - 3. Monotone increasing event. Represent the number of non-edges in $G[S_v]$ by the random variable $X_v = |\{\{u, w\} \in S_v \times S_v : \{u, v\} \notin E\}|$. The expected value of X_v is $q^2 \overline{\ell} d(v)$. By Lemma 7.3, the risk is at most $\Pr[\mathcal{E}_v]$, which by Lemma 8.3 is at most $\exp(-q\overline{\ell}/5)$.

4. Identical to part 3, switching the role of edges and non-edges.

The subsetting problems show that our method need not depend on the maximum degree, Δ , but rather in terms of the size of the domain of the sampling.

These properties can be easily generalized to restricted forms of sampling.

Observation 7.10. These properties hold also when we sample from a subset $T \subseteq V$. The bounds are then in terms of $d_T(v) = |T \cap N(v)|$.

We now use these to bound the risk of various subsampling problems. The lemma does not take simulatability of the (associated) events into account yet.

Lemma 7.11. The following problems can be formulated as LLLs with bounded risk:

- 1. Vertex subset splitting: Split a given subset $T \subseteq V$ of nodes into two parts V_1 , V_2 , such that each node v in V has between $d_T(v)/2$ and $3d_T(v)$ neighbors. Risk: $\exp(-\Omega(d_T(v)))$.
- 2. Sparsity splitting: We are given that each node has at least $\ell d(v)$ non-edges within its neighborhood and wish to partition the vertices into two sets S_1 , S_2 , such that each node has at least $\ell d(v)/16$ non-edges within $N(v) \cap S_1$ and within $N(v) \cap S_2$. Risk: $\exp(-\Omega(\ell))$.
- 3. Sparsity-preserving sampling: We are given that each node v has at least $\ell d(v)$ non-edges within its neighborhood) and a parameter d'. We seek a subset $S \subset V$ such that each node has: a) at most d' neighbors in S and b) at least $\ell' d'$ edges within $G[S_v]$, where $\ell' = \Omega(\ell)$ is maximized. Risk: $\exp(-\Omega(\ell))$.
- 4. Density splitting: Given that each node has at least $\ell d(v)$ edges within its neighborhood, partition the vertices into two sets S_1 , S_2 , such that each node has at least $\ell d(v)/16$ non-edges within $N(v) \cap S_1$ and within $N(v) \cap S_2$. Risk: $\exp(-\Omega(\ell))$.
- 5. Splitting a set with a large matching: Given that each node has a matching of size ℓ in its neighborhood, partition the vertices into two sets S_1 , S_2 , such that each node has a matching of size $\ell/64$ within $G[N(v) \cap S_1]$ and within $G[N(v) \cap S_2]$.
- *Proof.* 1. Combine Lemma 7.9 parts (1) and (2), using Lemma 7.8 and Observation 7.10.
 - 2. We use sampling with fair coins for each node. So, in the context of a node v, let X_1 , X_2 be the two parts of N[v] formed by the sampling. The expected sparsity within either set X_i is $\ell/4$. The bad event \mathcal{E} is when the sparsity within either part is less than $\ell/16$ and the associated event \mathcal{E}'_v is the stricter event that the sparsity within either part is less than $\ell/6$. The probability of \mathcal{E}'_v is $\exp(-\Omega(\ell))$, by Lemma 8.3. If \mathcal{E}'_v occurs, then all incident variables are retracted, in which case the marginal probability is at most that of \mathcal{E} itself, or $\exp(-\Omega(\ell))$. So, assume \mathcal{E}'_v did not occur. Now allow that an arbitrary subset T of N(v) gets retracted (without using the property of the second round of retractions). Let $X'_1 = X_1 \setminus T$ and $X'_2 = X_2 \setminus T$. Since \mathcal{E}'_v did not occur, it holds for either i that the set $X'_i \cup T$ has sparsity at least $\ell/6$. Now each node in T is flipped again with a fair coin, producing sets T_1 and T_2 (that are r.v.'s). Let $S_i = X'_i \cup T_i$, for i = 1, 2. The expected size of S_i is at least $\ell/24$ (since only a subset of T gets thrown out of $X'_i \cup T$ in the formation of S_i , each node with probability 1/2). Thus, the probability that S_i has sparsity less than $\ell/32$ is $\exp(-\Omega(\ell))$, by Lemma 8.3. Since this holds for every possible retraction, the risk is bounded above by $\exp(-\Omega(\ell))$.
 - 3. Follows by Lemma 7.9 part (1) and (3), combined via Lemma 7.8.
 - 4. Identical to case 2, switching the roles of edges and non-edges.
 - 5. Fix a particular matching M_v within G[N(v)] of size ℓ . We use again sampling with fair coins for each node, resulting in an initial partition of N(v) into vertex sets X_1 and X_2 . Let Y_i

be the number of edges of M_v with both endpoints in X_i , for i = 1, 2. Both endpoints of an edge in M_v are in Y_i with probability 1/4, so $\mathbb{E}[Y_i] = \mu = |M_v|/4 = \ell/4$. The bad events are $\mathcal{E} = \mathcal{E}^1 \cup \mathcal{E}^2$ with \mathcal{E}^i denoting that $Y_i < \mu/8$. Consider the stricter associated events \mathcal{E}' that either set satisfies $Y_i < \mu/2$. By Chernoff, $\Pr[\mathcal{E}'] \leq \Pr[\mathcal{E}] = exp(-\Omega(|M_v|))$. Our argument now follows part (2) closely. Consider the case when a node was happy with the assignment X_1, X_2 , but afterward, a (possibly empty) subset T of incident variables was retracted. To bound the risk, we want to bound the conditional probability of \mathcal{E} given the assignment fixed on $N(v) \setminus T$. The subgraph induced by $X_i \cup T$ had a matching of size $\mu/2$. When flipping the subset T to produce sets X'_1, X'_2 , in expectation at least a $\mu/8$ -sized matching remains in X'_i . Thus, by Chernoff, the probability X'_i contains no matching of size at least $\mu/16$ is $\exp(-\Omega(\mu))$. Hence, the risk is $\exp(-\Omega(\ell)$.

The proofs of the splitting problems (2), (4), and (5) in Lemma 7.11 do not explicitly use the monotonicity of some of the events. Instead, they rely on a hereditary property of the events.

8 Applications II: Coloring Sparse Graphs and Slack Generation

8.1 Degree+1 List Coloring (d1LC)

In the deg+1-list coloring (d1LC) problem, each node of a graph receives as input a list of available colors whose size exceeds its degree. The goal is to compute a proper vertex coloring in which each node outputs a color from its list. In the centralized setting, the problem can be solved with a simple greedy algorithm and it also admits efficient distributed algorithms.

Lemma 8.1 (List coloring [HKNT22, HNT22]). There is a randomized CONGEST algorithm to (deg + 1)-list-color (d1LC) any graph in $O(\log^5 \log n)$ rounds, w.h.p. This reduces to $O(\log^3 \log n)$ rounds when the degrees and the size of the color space is $poly(\log n)$.

In this section, we present our algorithms for degree-bounded sparsity-preserving sampling, slack generation, and our results on coloring triangle-free and sparse graphs with $\ll \Delta$ colors.

8.2 Sparsity-preserving Degree Reduction

Local Sparsity. The *(local) sparsity* ζ_v of a vertex v is defined as $\zeta_v = \frac{1}{\Delta} \left(\begin{pmatrix} \Delta \\ 2 \end{pmatrix} - m(N(v)) \right)$, where for a set X of vertices, m(X) denotes the number of edges in the subgraph G[X]. Roughly speaking, ζ_v is proportional to the number of missing edges \overline{m}_v in the graph induced by v's neighborhood. For a node with degree Δ , we have exactly $\overline{m}_v = \Delta \zeta_v$ non-edges in G[N(v)]. In general, the number of non-edges is $\binom{d(v)}{2} - m(N(v))$, where $m(N(v)) \leq \binom{\Delta}{2} - \zeta_v \Delta$ for nodes with sparsity at least ζ_v .

We require the following theorem to analyze how sparsity is preserved when randomly sampling nodes into a set.

Theorem 8.2 (Janson's inequality). Let $S \subseteq V$ be a random subset formed by sampling each $v \in V$ independently with probability p. Let \mathcal{A} be any collection of subsets of V. For each $A \in \mathcal{A}$, let $I_A := \mathbb{1}(A \subseteq S)$ be an indicator variable for the event that A is contained in S. Let $f := \sum_{A \in \mathcal{A}} I_A$ and $\mu := \mathbb{E}[f]$. Define

$$K := \frac{1}{2} \sum_{A,B \in \mathcal{A}, A \cap B \neq \emptyset} \mathbb{E}[I_A I_B]$$
(8)

Then, for any $0 \leq t \leq \mathbb{E}[f]$,

$$\Pr[f \le \mathbb{E}[f] - t] \le \exp\left(-\frac{t^2}{2\mu + K}\right)$$

The expected number of non-edges that is preserved when sampling nodes into a set S with probability p is a p^2 -fraction. The following lemma shows that the probability of deviating from this expectation is small.

Lemma 8.3 (Non-edge hitting Lemma). Let G be a graph on the vertex set X with \overline{m} non-edges. Sample each node of X with probability p into a set S and let f be the random variable describing the number of non-edges in G[S]. Then we have $\Pr(f \leq p^2 \overline{m}/2) \leq \exp(-p\overline{m}/5|X|)$.

Proof. Let \overline{E} be the set of non-edges in G. For each non-edge $e \in \overline{E}$, define an indicator variable $I_e = \mathbb{1}(e \subseteq S)$ for the event that the non-edge e is preserved in G[S]. We have $\mathbb{E}[f] = \sum_{e \in \overline{E}} \mathbb{E}[I_e] = \overline{mp^2}$. We can bound (8):

$$K = \frac{1}{2} \sum_{e,e' \in \overline{E}, e \cap e' \neq \emptyset} \mathbb{E}[I_e I_{e'}] \le \frac{1}{2} \overline{m}(2|X| - 2)p^3 \le \overline{m}|X|p^3$$

Using Theorem 8.2 with $t = \mathbb{E}[f]/2$, we can compute $\Pr[f \le p^2 \overline{m}/2] = \Pr[f \le \mathbb{E}[f] - t]$, where $\Pr[f \le \mathbb{E}[f] - t] \le \exp\left(-\frac{t^2}{2\mathbb{E}[f]+K}\right) \le \exp\left(-\frac{p^4 \overline{m}^2/4}{2p^2 \overline{m}+p^3 \overline{m}|X|}\right) = \exp\left(-\frac{p^2 \overline{m}}{8+4p|X|}\right) \le \exp\left(-\frac{p \overline{m}}{5|X|}\right)$ assuming $p|X| \ge 8$.

We use the following lemma to compute a small-degree subgraph that preserves a large number of non-edges:

Lemma 8.4 (Degree-Bounded Sparsity-Preserving Sampling). Assume that $\log^2 \log n \leq \Delta \leq O(\log n)$. Let $X, Y \subseteq V$ such that for all $v \in X$, the number of non-edges in $G[N(v) \cap Y]$ is at least $\alpha \Delta^2$ for some $0 < \alpha \leq 1/2$ with $1/\alpha = O(\operatorname{poly} \log \log n)$. Let $\mu = (600/\alpha) \log \Delta \cdot \log \log n$. There is a randomized poly $\log \log n$ -time CONGEST algorithm, that w.h.p. finds a set $S \subseteq Y$ s.t. each $v \in X$ has at most $\Delta_s = 4\mu$ neighbors in S, and at least $\overline{m}_{thres} = \alpha \mu^2/2 = \Omega((1/\alpha) \log^2 \Delta \cdot \log^2 \log n)$ non-edges in subgraph induced by $N(v) \cap S$.

Proof. We define a sparsity-preserving degree reduction LLL and solve it with the algorithm of Theorem 4.4 for binary LLLs with low risk. For each $w \in Y$, define a variable indicating that w is sampled to S, which happens with probability $p := \mu/\Delta$. Sampled nodes are called black and non-sampled nodes white. For each $v \in X$, define two unwanted events \mathcal{E}_d and \mathcal{E}_{ζ} :

- Let \mathcal{E}_d be the event that v has more than 4μ sampled neighbors in S. The expected number of neighbors in S is $d(v) \cdot \mu/\Delta \leq \mu$. Hence, $\Pr(\mathcal{E}_d) \leq \exp(-2\mu/3)$ by Chernoff. Additionally, define an associated event $\operatorname{assoc}(\mathcal{E}_d)$ as the event that at most 2μ neighbors are sampled. We have $\Pr(\operatorname{assoc}(\mathcal{E}_d)) \leq \exp(-\mu/3)$. This bounds the risk of \mathcal{E}_d to be at most $\Pr(\operatorname{assoc}(\mathcal{E}_d))$ by Lemma 7.7
- Let \mathcal{E}_{ζ} be the event that the number of non-edges in $G[N(v) \cap S]$ is less than $\overline{m}_{thres} = \alpha \mu^2/2$. Let f be a random variable for the number of non-edges in the graph induced by $X = N(v) \cap S$. Apply Lemma 8.3, with $|X| \leq \Delta$ and $\overline{\mu} \geq \alpha \Delta^2$. We have $\mathbb{E}[f] \geq p^2 \overline{m} \geq \alpha \mu^2$. This gives $\Pr(\mathcal{E}_{\zeta}) = \Pr(f \leq \mathbb{E}[f]/2) \leq \exp\left(-\frac{p\overline{m}}{5|X|}\right) \leq \exp\left(-\frac{\alpha\mu}{5}\right)$. \mathcal{E}_{ζ} is a monotone increasing event. Hence, its risk is at most $p_{\zeta} = \Pr(\mathcal{E}_{\zeta})$ by Lemma 7.3, where the associated event $\operatorname{assoc}(\mathcal{E}_{\zeta})$ is \mathcal{E}_{ζ} itself.

For each $v \in X$, a bad event \mathcal{E} is defined as the union $\mathcal{E}_d \cup \mathcal{E}_{\zeta}$. This event depends on the sampling status of the neighbors of v. Hence, the dependency degree of the sparsity-preserving sampling LLL is $d = \Delta^2$. By Lemma 7.8, the associated event of \mathcal{E} is the union, $\operatorname{assoc}(\mathcal{E}) = \operatorname{assoc}(\mathcal{E}_d) \cup \operatorname{assoc}(\mathcal{E}_{\zeta})$. The risk of \mathcal{E} is at most $p_{\zeta} + p_d = \Pr(\mathcal{E}_{\zeta}) + 2\Pr(\mathcal{E}_d) \leq e^{-\mu/3} + 2e^{-\alpha\mu/5} \leq d^{100 \cdot \log \log n}$.

The locality of the LLL is 2, since dependent events are within two hops of each other in G. We show that the LLL is simulatable. We can show that the required primitives in Definition 2.3 can be executed in $O(\mu) = \text{poly} \log \log n$ rounds:

- 1. Test: Sampled nodes inform their neighbors. If an event node $v \in X$ has more than 2μ sampled neighbors, the event $\operatorname{assoc}(\mathcal{E})$ occurs. Otherwise, v pipelines a list of IDs of its sampled neighbors $L = N(v) \cap S$ to each of its (sampled) neighbors in $O(\mu)$ rounds. Each neighbor $w \in N(v) \cap S$ receiving the list report which nodes $x \in L$ are not in N(w). This allows v to learn the list of non-edges in $G[N(v) \cap S]$.
- 2. 1-bit Min-Aggregation: Trivial to do in one round, as variables are adjacent to events in G.

For the following, we can assume that nodes have access to $\mathsf{IDbitLen} = \Theta(\operatorname{poly} \log \log n)$ -bit IDs.

- 3 Evaluate: We start with some pre-processing for each $v \in X$ to learn the edges in G[N(v)] (all parallel instances share a common pre-processing phase). All nodes $v \in V$ forward the list of IDs in N(v) to each neighbor. The IDs take at most $O(\mathsf{IDbitLen} \cdot \Delta) = O(\log n \cdot \mathsf{poly} \log \log n$ bits, which can be sent in poly $\log \log n$ rounds. Given partial assignments φ and ψ , any event node $v \in X$ can compute the required conditional probabilities locally, using knowledge of the structure of G[N(v)].
- 4 Min-aggregation: This is trivial as variables are adjacent to events in G. Sending the $O(\log n)$ different $O(\log \log n)$ -bit strings takes $O(\log \log n)$ rounds.

8.3 Slack Generation for Sparse Nodes

The slack of a node (potentially in a subgraph) is defined as the difference between the size of its palette and the number of uncolored neighbors (in the subgraph).

Definition 8.5 (Slack). Let v be a node with color palette $\Psi(v)$ in a subgraph H of G. The slack of v in H is the difference $|\Psi(v) - d|$, where d is the number of uncolored neighbors of v in H.

Slack generation is based on trying a random color for a subset of nodes. Sample a set of nodes and a random color for each of the sampled nodes. Nodes keep the random color if none of their neighbors chose the same color. See Algorithm 2 for a pseudocode.

Consider a node v with lots of sparsity in its neighborhood, i.e., many non-edges in G[N(v)]. Let $w, w' \in N(v)$ be two neighbors of v such that w, w' are not adjacent. If w and w' keep the same color, v gets one unit of slack, as two of its neighbors get colored while the color palette is only reduced by one color. It can be shown that the obtained slack is roughly proportional to the sparsity of v, see for example [EPS15, HKMT21]. We prove a similar result in terms of the number of non-edges, with a custom-sized color palette:

Lemma 8.6 (Slack generation with custom color space). Let Δ_s, χ be positive integers with $\chi \geq c'\Delta_s$ for some constant c' > 0. Let $S \subseteq V$ be a subset of nodes. Consider a node $v \in V$ with at least \overline{m} non-edges in $G[N(v) \cap S]$. Suppose that all nodes in S, as well as v, have at most Δ_s neighbors in

Algorithm 3 TRYCOLOR (vertex v, color c_v)

- 1: Send c_v to N(v), receive the set $T = \{c_u : u \in N(v)\}$.
- 2: if $c_v \notin T$ then permanently color v with c_v .
- 3: Send/receive permanent colors, and remove the received ones from $\Psi(v)$.

S. After running SlackGeneration on S with color palette $[\chi]$, the slack of v is increased by at least $e^{-3/c'}\overline{m}/(500\chi) = \Omega(\overline{m}/\chi)$, with probability at least $1 - \exp(-\Omega(\overline{m}/\chi))$. This holds independently of the random choices at a distance greater than 2.

Proof. Let $N_S(v) = N(v) \cap S$ for short. Let $X \subseteq \binom{N_S(v)}{2}$ be the set of non-edges, where $|X| = \overline{m}$. Each node in S is activated with probability p = 1/20. Activated node w runs TRYCOLOR, where w selects a color u.a.r. from $[\chi]$.

Let c_w be the random color chosen, and let x_w be the possible permanent color, or $x_w = \bot$ in case of a conflict.

Let Z be the number of colors $c \in [\chi]$ s.t. there exists a non-edge $\{u, w\} \in X$ with $c_u = c_w = c$, and for all such non-edges, $x_u = x_w = c$, i.e. all the nodes retain the color (see below for a formal definition with quantifiers). Say that a non-edge $\{u, w\} \in X$ is successful if $x_u = x_w \neq \bot$, and no node in $N_S(v) \setminus \{u, w\}$ picks the same color. Let $Y_{\{u, w\}}$ be an indicator function for the event that $\{u, w\}$ is successful. We have $\mathbb{E}[Z] \geq \sum_{\{u, w\} \in X} \mathbb{E}[Y_{\{u, w\}}]$, since each non-edge with $Y_{\{u, w\}} = 1$ counts towards Z. The probability of a non-edge being successful is at least

$$\Pr(Y_{\{u,w\}}) \ge p^2 \left(\frac{\chi - 1}{\chi}\right)^{2\Delta_s - 2} \left(\frac{1}{\chi}\right) \left(\frac{\chi - 1}{\chi}\right)^{\Delta_s - 2} \ge \frac{p^2}{\chi} \left(\frac{\chi - 1}{\chi}\right)^{3\Delta_s} \ge \frac{e^{-3/c'} p^2}{\chi}$$

where u and v are activated with probability p^2 and select the same color w.p. $1/\chi$; with probability $(\frac{\chi-1}{\chi})^{2\Delta_s-2}$ none of the neighbors of u nor v choose that particular color, and $(\frac{\chi-1}{\chi})^{\Delta_s-2}$ is the probability that no other neighbors of v choose that color. In the last inequality, we used that $\chi \geq c'\Delta_s$. This gives $\mathbb{E}[Z] \geq \sum_{\{u,w\} \in X} \mathbb{E} Y_{\{u,w\}} \geq \overline{m}e^{-3/c'}p^2/\chi$.

Next, we show that Z is concentrated around its mean. Let T be the number of colors that are randomly chosen in TRYCOLOR by both nodes of at least one non-edge in X. Let D be the number of colors that are chosen in TRYCOLOR by both nodes of at least one non-edge in X, but are not retained by at least one of them. Formally,

$$T := \# \text{ colors } c \text{ s.t. } \exists \{u, w\} \in X : c_u = c_w = c$$
$$D := \# \text{ colors } c \text{ s.t. } (\exists \{u, w\} \in X : c_u = c_w = c) \land (\exists \{u, w\} \in X : (c_u = c_w = c) \land (x_u = \bot \lor x_w = \bot))$$
$$Z := \# \text{ colors } c \text{ s.t. } (\exists \{u, w\} \in X : c_u = c_w = c) \land (\forall \{u, w\} \in X : (c_u = c_w = c) \implies (x_u = x_w = c))$$

We have Z = T - D, since D counts the colors where the implication in the definition of Z fails.

We upper bound $\mathbb{E}[T]$ (which implies the same bound for D, as $D \leq T$). For a fixed color c, the probability that both u, w pick c is at most $1/\chi^2$. By union bound, the probability that c is picked by at least one non-edge is at most \overline{m}/χ^2 . There are χ colors, so $\mathbb{E}[T] \leq \chi \cdot \overline{m}/\chi^2 = \overline{m}/\chi$.

The functions T and D are r-certifiable with r = 2 and r = 3, respectively. See the appendix and Lemma A.1 for the definition of an r-certifiable function. T and D are both 2-Lipschitz: whether a node is activated, and which color it picks affects the outcome by at most c = 2. We apply Lemma A.1 with $b = \mathbb{E}[Z]/10 - 60c\sqrt{r \cdot \mathbb{E}[T]}$:

$$\begin{aligned} \Pr[|T - \mathbb{E}[T]| &\geq b + 60c\sqrt{r \cdot \mathbb{E}[T]}] = \Pr[|T - \mathbb{E}[T]| \geq \mathbb{E}[Z]/10] \\ &\leq 4\exp\left(-\frac{\left(\mathbb{E}[Z]/10 - 60c\sqrt{r \cdot \mathbb{E}[T]}\right)^2}{8c^2 r \mathbb{E}[T]}\right) \\ &\leq \exp\left(-\Theta(1)\left(\frac{\mathbb{E}[Z]^2}{\mathbb{E}[T]} - \frac{\mathbb{E}[Z]}{\sqrt{\mathbb{E}[T]}} + O(1)\right)\right) \\ &\leq \exp\left(-\Omega(\overline{m}/\chi)\right) \end{aligned}$$

In the last inequality, we used that $\mathbb{E}[Z] \geq \overline{m}e^{-3/c'}p^2/\chi$ and $\mathbb{E}[T] \leq \overline{m}/\chi$. The same concentration bound applies for D, meaning that $\Pr[|D - \mathbb{E}[D]| \ge \mathbb{E}[Z]/10] \le \exp(-\Omega(\overline{m}/\chi))$. By union bound, neither of the events $|T - \mathbb{E}[T]| \geq \mathbb{E}[Z]/10$ and $|D - \mathbb{E}[D]| \geq \mathbb{E}[Z]/10$ occur with probability at least $1 - 2 \exp\left(-\Omega(\overline{m}/\chi)\right)$. Hence, $Z = T - D \geq \mathbb{E}[T] - \mathbb{E}[Z]/10 - (\mathbb{E}[D] + \mathbb{E}[Z]/10) = (4/5) \cdot \mathbb{E}[Z] \geq 1$ $e^{-3/c'}\overline{m}/(500\chi)$, with probability at least $1 - \exp(-\Omega(\overline{m}/\chi))$.

In the next lemma, we use our disjoint variable set LLL to produce large amounts of slack for nodes. The lemma assumes that we are already given two sets S_1 and S_2 that induce sufficiently many non-edges in the neighborhood of every relevant node.

Lemma 8.7. Let $\Delta_s = O(\operatorname{poly} \log \log n)$. Let \overline{m} and $\chi = O(\Delta)$ be positive integers such that $\overline{m}/\chi = \Omega(\log \Delta \cdot \log \log n)$ and $\chi \geq c'\Delta_s$ for some constant c'. Let $W \subseteq V$ and let $S_1, S_2 \subset V$ be disjoint sets such that for i = 1, 2,

- $\forall v \in (W \cup S_1 \cup S_2) : d_{S_s}(v) < \Delta_s$,
- $\forall v \in W$: the number of non-edges in $N(v) \cap S_i$ is at least \overline{m}

There is a randomized CONGEST algorithm that w.h.p. colors a subset of $S_1 \cup S_2$ using a palette of size 2χ such that every node in W has at least $e^{-3/c'}\overline{m}/(500\chi) = \Omega(\overline{m}/\chi)$ same-colored neighbors. Every node in W, S_1, S_2 has at most $2\Delta_s$ of its neighbors colored.

Proof. We define a disjoint variable LLL for slack generation and solve it with Theorem 5.2. The random process is defined by the SLACKGENERATION algorithm. We define two variables for each $v \in S_1 \cup S_2$. Let a_v be a variable indicating that v is activated, which happens with probability 1/20. Let c_v be a variable for a color chosen uniformly at random from the node's palette, $1, \ldots, \chi$ and $\chi + 1, \ldots, 2\chi$ for nodes in S_1 and S_2 , respectively. For notational purposes, c_v is defined for all nodes, regardless of the value of a_v . The variables a_v, c_v are also independent of the variables of other nodes. For each $v \in W$, define the bad event \mathcal{E}_v that the slack of v does not get increased by at least $s_{thres} := e^{-3/c'} \overline{m}/(500\chi)$. The probability of \mathcal{E}_v is at most $\exp(-\Omega(\overline{m}/\chi))$ by Lemma 8.6. We show that the slack generation LLL is simulatable.

- 1. Test: Each variable node $v \in S_1$ runs SLACKGENERATION. Activated nodes that retained their color inform their neighbors. An event node $w \in W$ receives the retained colors in its neighborhood and counts how much slack was generated.
- 2. 1-bit min-aggregation: Each event broadcasts its bit, traveling for 2 hops. Broadcasts of multiple events can be combined (since any variable node $v \in S$ is a variable for all events in its 2-hop neighborhood). The same works in the other direction.

By the definition of simulatability, we can assume that each event and variable has access to a unique $O(\log \log n)$ -bit identifier id for the following:

- 3. Evaluate: We show the primitive for a single partial assignment using poly log log n bandwidth – the primitive for $O(\log n)$ parallel instances follows from this. Let $S = S_1 \cup S_2$. Each event node $w \in W$ can learn the list of neighbors in S for each of its immediate neighbors $v \in N(w) \cap S$ in $O(\Delta_s) = \text{poly log log } n$ rounds (this will even be the same for all parallel instances). Given a partial assignment ψ , where each variable node $v \in S$ knows its values (or \perp) for a_v and c_v , w learns the values in $N^2(w) \cap S$: each immediate neighbor $v \in N(w)$ pipelines $L = \{(\psi(x), \mathrm{id}(x)) : x \in N(v) \cap S\}$ to w. Communicating L takes $O(\Delta_s \cdot \chi) =$ poly log log n bits. Knowing the structure of the distance-2 neighborhood in S, as well as the partial assignment for nodes in $N^2(w) \cap S$, allows w to locally compute the conditional probability of obtaining enough slack, i.e. its event not occurring.
- 4 Min-aggregation: The principle is the same as for 1-bit aggregation. Sending the $O(\log n)$ different $O(\log \log n)$ -bit strings takes $O(\log \log n)$ rounds.

8.4 Coloring Sparse Graphs

In this section, we prove the following theorem.

Theorem 8.8 (Coloring sparse graphs). Let G be a graph with maximum degree $\Delta = \Omega(\log^2 \log n)$ and sparsity $\zeta \geq \epsilon^2 \Delta$. Let $x = \log \Delta \cdot \log \log n/6$. There is a randomized polyloglog n-time CONGEST algorithm computing a $(\Delta - x)$ -coloring of G with high probability.

Proof. Let $\Delta' = \Delta - x$. Let $D = \{v \in V : d(v) \geq \Delta'\}$. As a node $v \in D$ with $d(v) \geq \Delta'$ initially has more neighbors than available colors, we need to increase its slack by at least $d(v) + 1 - \Delta' \leq \Delta + 1 - \Delta' = x + 1$ to get a $(\deg + 1)$ -list coloring instance. The sparsity of any node vis $\zeta = \frac{1}{\Delta} \left({\Delta \choose 2} - m(v) \right) \geq \epsilon^2 \Delta$, which implies that the number of edges m(v) in G[N(v)] is at most ${\Delta \choose 2} - \Delta^2 \epsilon^2$. For a node $v \in D$, the number of non-edges in G[N(v)] is at least ${\Delta \choose 2} - m(v) \geq {\Delta \choose 2} - {\Delta \choose 2} + \epsilon^2 \Delta^2 \geq \frac{1}{2} \epsilon^2 \Delta^2 =: \overline{m}$ where the last inequality assumes $x \leq \epsilon^2 \Delta/2$, which holds for large enough n.

Assume that $\Delta = \Omega(\log^2 \log n)$ and $\Delta \leq 100 \log n$. Start by computing two disjoint sets $S_1, S_2 \subseteq V$, that preserve the sparsity for nodes in D, while having a bounded degree. Apply Lemma 8.4 for X = D and Y = V, where the number of non-edges is at least $\overline{m} = \epsilon^2 \Delta^2/2$. The result is a set $S_1 \subseteq Y$ s.t. all nodes have at most $(4800/\epsilon^2) \log \Delta \cdot \log \log n$ neighbors in S_1 and each $v \in D$ has at least $(10^5/\epsilon^2) \log^2 \Delta \cdot \log^2 \log n$ non-edges in $G[N(v) \cap S_1]$. The algorithm works with high probability and runs in polylog $\log n$ rounds. The number of non-edges in the remaining graph $G[N(v) \cap (V \setminus S_1)]$ for $v \in D$ is at least $\epsilon^2 \Delta^2/2 - \Delta \cdot \Delta_s \geq \epsilon^2 \Delta^2/4$ when n is large enough. Compute another sparsity preserving set $S_2 \subset V \setminus S_1$ with the same approach. All nodes have at most $\Delta_s := (9600/\epsilon^2) \log \Delta \cdot \log \log n$ neighbors in S_2 and each $v \in D$ has at least $\overline{m}_s := (10^5/\epsilon^2) \log^2 \Delta \cdot \log^2 \log n$ non-edges in $G[N(v) \cap S_2]$.

We generate slack for nodes in D by running Lemma 8.7 with the sets S_1, S_2 . We use a color palette of size $2\Delta_s$ (divided equally for the two sets). This colors a subset of $S_1 \cup S_2$ (possibly including nodes in D), such that the slack for each $v \in D$ is increased by at least $\overline{m}_s/(1000e^3 \cdot \Delta_s) \ge$ $(1/5) \log \Delta \cdot \log \log n$. This is at least the required slack, x + 1. At this point, we can color the remaining uncolored nodes in D and $V \setminus D$ (at the same time) with the (deg+1)-list coloring algorithm of Lemma 8.1. Lastly, suppose that $\Delta \geq 100 \log n$. Run SLACKGENERATION on G, using all Δ' colors. It was previously shown that each $v \in D$ has at least $\overline{m} = \frac{1}{2}\epsilon^2\Delta^2$ non-edges in its neighborhood. By Lemma 8.6, each $v \in D$ has its slack increased by at least $\frac{1}{1000e^6}\frac{\epsilon^2\Delta^2}{\Delta - x} \geq \frac{1}{10^6}\epsilon^2\Delta$, with probability at least $1 - \exp(-\Omega(\overline{m}/\Delta')) \geq 1 - \exp(-\Omega(\Delta)) \geq 1 - n^{-c}$ for some constant c. The obtained slack $\frac{1}{10^6}\epsilon^2\Delta$ is at least the required $\log \Delta \cdot \log \log n + 1$ when n is large enough.

8.5 Coloring Triangle-free Graphs

Triangle-free graphs are maximally sparse in the sense that neighbors of a node are never connected. This allows us to generate slack linear in Δ . We prove the following theorem.

Theorem 8.9 (Coloring Triangle-free Graphs). Let G be a triangle-free graph with maximum degree Δ . There is a randomized polylog log n-time CONGEST algorithm to $\gamma\Delta$ -color G with high probability, for a constant $\gamma = 1 - 10^{-7}$.

Proof. Let $\Delta' = \gamma \Delta$. Let $D = \{v \in V : d(v) \ge \Delta'\}$. We need to generate slack for all nodes in D. As a node $v \in D$ with $d(v) \ge \Delta'$ initially has more neighbors than available colors, we need to increase its slack by at least $d(v) + 1 - \Delta' \le \Delta + 1 - \Delta'$ to get a $(\deg + 1)$ -list coloring instance.

We consider three different ranges of Δ : (1) $\Delta = O(\log^2 \log n)$, (2) $\Delta = \Omega(\log^2 \log n)$ and $\Delta \leq 100 \log n$, and (3) $\Delta \geq 100 \log n$. We start by giving a proof for (2). In the other cases, the result follows from previous work, or all events hold with high probability.

Assume that $\Delta = \Omega(\log^2 \log n)$ and $\Delta \leq 100 \log n$. Split all vertices into $k = 2\epsilon^4 \Delta/(\ln \Delta \log^2 \log n)$ classes with discrepancy $\epsilon \Delta/k$ using the algorithm of [HMN22, Theorem 23], for some small constant ϵ . The algorithm runs in poly log log n rounds, with high probability. It produces a partition V_1, \ldots, V_k of V s.t. for all $v \in V$ and all $1 \leq i \leq k : d_{V_i}(v) = d(v)/k \pm \epsilon \Delta/k$. In particular, for all $v \in D$, the number of neighbors in V_i is at most $\Delta_s := \frac{\Delta}{k} + \frac{\epsilon \Delta}{k} = O(\ln \Delta \cdot \log^2 \log n)$ and at least $\delta_s := \frac{\gamma \Delta}{k} - \frac{\epsilon \Delta}{k} = \Omega(\ln \Delta \cdot \log^2 \log n)$. As there are no triangles, for each $v \in D$ the number of non-edges in $G[N(v) \cap V_i]$ is at least $\overline{m} := {\delta_s \choose 2} = \frac{1}{2} (\frac{\gamma \Delta}{k} - \frac{\epsilon \Delta}{k}) (\frac{\gamma \Delta}{k} - \frac{\epsilon \Delta}{k} - 1) \geq \frac{\Delta^2}{16k^2}$ assuming $\gamma - \epsilon \geq 1/2$.

We generate slack on the classes of the partition in parallel. For $1 \leq i \leq k$, each class V_i is assigned a subset of $\chi = \lfloor \frac{\Delta'}{k} \rfloor$ colors, $[(i-1)\lfloor \frac{\Delta'}{k} \rfloor, i \cdot \lfloor \frac{\Delta'}{k} \rfloor]$. Form k/2 instances, each using a pair of vertex classes: for $1 \leq j \leq k/2$, the nodes in V_{2j-1}, V_{2j} are used together. We apply Lemma 8.7 on each instance in parallel. The size of the color palette satisfies $\chi \geq c\Delta_s$ for c = 1/2. Each instance succeeds with high probability, and we take a union bound over all k/2 instances. To avoid congestion, we assign each instance a subset of the edges for communication, such that any edge is used for at most two instances. Note that a node in V_i only needs to communicate with other nodes in V_i within 2 hops. Let $\{v, w\} \in E$ be any edge, where $v \in V_x$ and $w \in V_y$ for some $1 \leq x \leq y \leq k$. The edge $\{v, w\}$ is used for communication in the instances with V_x and V_y , for a total of at most 2 instances.

By Lemma 8.7, in each instance, the amount of slack generated is at least $\frac{e^{-3/c\overline{m}}}{500\chi} \ge \frac{\Delta^2/16k^2}{500e^6\cdot\gamma\Delta/k} \ge \frac{\Delta}{4\cdot10^6\cdot k}$. Over all the k/2 instances, the amount of slack generated for each $v \in D$ is at least $\Delta \cdot 10^{-7}$. Hence, all nodes in D get slack by at least the required amount, $d(v)+1-\Delta' \le \Delta+1-\gamma\Delta \approx 10^{-7}\cdot\Delta$. At this point, we can color the remaining uncolored nodes in D and $V \setminus D$ (at the same time) with the $(\deg + 1)$ -list coloring algorithm of Lemma 8.1.

It remains to handle cases (1) and (3). When $\Delta \geq 100 \log n$, slack is generated with high probability. Run SLACKGENERATION on G, using all $\chi = \Delta'$ colors. Each $v \in D$ has at least $\overline{m} = (\Delta')^2/2$ non-edges in its neighborhood. By Lemma 8.6, each $v \in D$ has its slack increased by at least $\frac{\overline{m}}{500e^3\chi} \geq \frac{1}{21000}\Delta' \geq \Delta \cdot 10^{-7}$, with probability at least $1 - \exp(-\Omega(\overline{m}/\chi)) \geq 1 - \exp(-\Omega(\gamma\Delta)) =$

 $1-1/n^c$ for some constant c. Lastly, when $\Delta = O(\log^2 \log n)$, this same process of slack generation is an exponential LLL with a small dependency degree, $d = \text{poly} \log \log n$. This can be solved with high probability using the algorithm of [MU21], running in poly $\log \log n$ rounds.

References

- [Bar15] L. Barenboim. Deterministic (Δ + 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.
- [BBH⁺21] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. J. ACM, 68(5):39:1–39:30, 2021.
 - [BE10] Leonid Barenboim and Michael Elkin. Sublogarithmic distributed mis algorithm for sparse graphs using nash-williams decomposition. *Distributed Computing*, 22(5):363– 379, 2010.
 - [BE13] Leonid Barenboim and Michael Elkin. Distributed Graph Coloring: Fundamentals and Recent Developments. Morgan & Claypool Publishers, 2013.
 - [Bec91] József Beck. An Algorithmic Approach to the Lovász Local Lemma. Random Structures & Algorithms, 2(4):343–365, 1991.
 - [BEG18] Leonid Barenboim, Michael Elkin, and U. Goldenberg. Locally-iterative distributed (delta + 1)-coloring below szegedy-vishwanathan barrier, and applications to selfstabilization and to restricted-bandwidth models. In the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), 2018.
- [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.
- [BFH⁺16a] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A Lower Bound for the Distributed Lovász Local Lemma. In the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), 2016.
- [BFH⁺16b] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed Lovász local lemma. In Proc. 48th ACM Symposium on Theory of Computing (STOC 2016), pages 479–488. ACM, 2016.
 - [BGR20] Sebastian Brandt, Christoph Grunau, and Václav Rozhon. Generalizing the sharp threshold phenomenon for the distributed complexity of the Lovász local lemma. In PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020, pages 329–338, 2020.
 - [BKM20] Philipp Bamberger, Fabian Kuhn, and Yannic Maus. Efficient deterministic distributed coloring with small bandwidth. In PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020, pages 243–252, 2020.

- [BMU19] Sebastian Brandt, Yannic Maus, and Jara Uitto. A sharp threshold phenomenon for the distributed complexity of the Lovász local lemma. In Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019, pages 389–398, 2019.
 - [Bra19] Sebastian Brandt. An automatic speedup theorem for distributed problems. In Peter Robinson and Faith Ellen, editors, Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019, pages 379–388. ACM, 2019.
- [CHL⁺20] Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. Distributed edge coloring and a special case of the constructive Lovász local lemma. ACM Trans. Algorithms, 2020.
 - [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.
 - [CP19] Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the LOCAL model. SIAM J. Comput., 48(1):33–69, 2019.
 - [CPS17] Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed algorithms for the lovász local lemma and graph coloring. *Distributed Comput.*, 30(4):261–280, 2017.
 - [Dav23] Peter Davies. Improved distributed algorithms for the lovász local lemma and edge coloring. In Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 4273–4295. SIAM, 2023.
 - [EL74] Paul Erdös and László Lovász. Problems and Results on 3-chromatic Hypergraphs and some Related Questions. Colloquia Mathematica Societatis János Bolyai, pages 609–627, 1974.
 - [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.
 - [FG17] Manuela Fischer and Mohsen Ghaffari. Sublogarithmic Distributed Algorithms for Lovász Local Lemma, and the Complexity Hierarchy. In the Proceedings of the 31st International Symposium on Distributed Computing (DISC), pages 18:1–18:16, 2017.
- [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.
- [FHM23] Manuela Fischer, Magnús M. Halldórsson, and Yannic Maus. Fast distributed Brooks' theorem. In the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 2567–2588, 2023.
 - [FK23] Marc Fuchs and Fabian Kuhn. List defective colorings: Distributed algorithms and applications. In Rotem Oshman, editor, 37th International Symposium on Distributed Computing, DISC 2023, October 10-12, 2023, L'Aquila, Italy, volume 281 of LIPIcs, pages 22:1–22:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

- [GG23] Mohsen Ghaffari and Christoph Grunau. Faster deterministic distributed MIS and approximate matching. In Barna Saha and Rocco A. Servedio, editors, Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023, pages 1777–1790. ACM, 2023.
- [GGH⁺23] Mohsen Ghaffari, Christoph Grunau, Bernhard Haeupler, Saeed Ilchi, and Václav Rozhoň. Improved distributed network decomposition, hitting sets, and spanners, via derandomization. In the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 2532–2566, 2023.
 - [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.
 - [Gha19] Mohsen Ghaffari. Distributed maximal independent set using small messages. In Proc. 30th Symp. on Discrete Algorithms (SODA), pages 805–820, 2019.
 - [GHK18] Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018, pages 662–673, 2018.
- [GHKM18] Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, and Yannic Maus. Improved distributed delta-coloring. In Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018, pages 427–436, 2018.
 - [GK21] Mohsen Ghaffari and Fabian Kuhn. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. In the Proceedings of the Symposium on Foundations of Computer Science (FOCS), pages 1009–1020, 2021.
 - [GKM17] Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 784–797. ACM, 2017.
 - [GS17] Mohsen Ghaffari and Hsin-Hao Su. Distributed degree splitting, edge coloring, and orientations. In the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 2505–2523, 2017.
- [HKMT21] Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Tigran Tonoyan. Efficient randomized distributed coloring in CONGEST. In the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 1180–1193, 2021. Full version at CoRR abs/2105.04700.
- [HKNT22] Magnús M. Halldórsson, Fabian Kuhn, Alexandre Nolin, and Tigran Tonoyan. Nearoptimal distributed degree+1 coloring. In Stefano Leonardi and Anupam Gupta, editors, STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022, pages 450–463. ACM, 2022.
- [HMN22] Magnús M. Halldórsson, Yannic Maus, and Alexandre Nolin. Fast distributed vertex splitting with applications. In Christian Scheideler, editor, 36th International Symposium on Distributed Computing, DISC 2022, October 25-27, 2022, Augusta, Georgia, USA, volume 246 of LIPIcs, pages 26:1–26:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

- [HN21] Magnús M. Halldórsson and Alexandre Nolin. Superfast coloring in CONGEST via efficient color sampling. In Tomasz Jurdzinski and Stefan Schmid, editors, Structural Information and Communication Complexity - 28th International Colloquium, SIROCCO 2021, Wrocław, Poland, June 28 - July 1, 2021, Proceedings, volume 12810 of Lecture Notes in Computer Science, pages 68–83. Springer, 2021.
- [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.
- [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.
- [Joh99] Öjvind Johansson. Simple distributed $\Delta + 1$ -coloring of graphs. Inf. Process. Lett., 70(5):229–232, 1999.
- [Lin92] Nati Linial. Locality in distributed graph algorithms. SIAM Journal on Computing, 21(1):193–201, 1992.
- [Mau21] Yannic Maus. Distributed graph coloring made easy. In Kunal Agrawal and Yossi Azar, editors, SPAA '21: 33rd ACM Symposium on Parallelism in Algorithms and Architectures, Virtual Event, USA, 6-8 July, 2021, pages 362–372. ACM, 2021.
- [MPU23] Yannic Maus, Saku Peltonen, and Jara Uitto. Distributed symmetry breaking on power graphs via sparsification. In *Proceedings of the 2023 ACM Symposium on Principles* of Distributed Computing, PODC '23, page 157–167, New York, NY, USA, 2023. Association for Computing Machinery.
 - [MR13] Michael Molloy and Bruce Reed. *Graph colouring and the probabilistic method*, volume 23. Springer Science & Business Media, 2013.
 - [MT10] Robin A. Moser and Gábor Tardos. A Constructive Proof of the General Lovász Local Lemma. J. ACM, pages 11:1–11:15, 2010.
 - [MT20] Yannic Maus and Tigran Tonoyan. Local conflict coloring revisited: Linial for lists. In Hagit Attiya, editor, 34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference, volume 179 of LIPIcs, pages 16:1–16:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
 - [MU21] Yannic Maus and Jara Uitto. Efficient CONGEST algorithms for the Lovász local lemma. In Seth Gilbert, editor, the Proceedings of the International Symposium on Distributed Computing (DISC), volume 209 of LIPIcs, pages 31:1–31:19. Schloss Dagstuhl
 Leibniz-Zentrum für Informatik, 2021.
 - [NS95] Moni Naor and Larry Stockmeyer. What can be computed locally? SIAM J. on Comp., 24(6):1259–1277, 1995.
 - [Pel00] David Peleg. Distributed Computing: A Locality-Sensitive Approach. SIAM, 2000.
 - [RG20] Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 350–363, 2020.

A Supplementary Results

Concentration Bounds

We use the following Talagrand's inequality. A function $f(x_1, \ldots, x_n)$ is called *c-Lipschitz* iff the value of any single x_i affects f by at most c. Additionally, f is *r*-certifiable if for every $x = (x_1, \ldots, x_n)$, (1) there exists a set of indices $J(x) \subseteq [n]$ such that $|J(x)| \leq r \cdot f(x)$, and (2) if x' agrees with x on the coordinates in J(x), then $f(x') \geq f(x)$.

Lemma A.1 (Talagrand's Inequality II [MR13]). Let X_1, \ldots, X_n be independent random variables and $f(X_1, \ldots, X_n)$ be a c-Lipschitz, r-certifiable function. For any $b \ge 1$:

$$\Pr(|f - \mathbb{E}[f]| > b + 60c\sqrt{r \mathbb{E}[f]}) \le 4 \exp\left(-\frac{b^2}{8c^2 r \mathbb{E}[f]}\right)$$

Deterministic LLL in LOCAL

LLLs can be solved efficiently with deterministic algorithms in the LOCAL model.

Theorem A.2 (Deterministic LLL in LOCAL, [RG20]). There is a deterministic LOCAL algorithm for the constructive Lovász local lemma under criterion $epd(1 + \varepsilon) < 1$, for a constant $\varepsilon > 0$ that runs in $O(\log^* s) + poly \log n$ rounds if the communication network has at most n nodes and node IDs are from a space of size s and the event/variable assignment has locality poly log n.

Theorem A.2 is proven by using the powerful general derandomization framework of [RG20, GHK18, GKM17] for the algorithm of Moser-Tardos [MT10].

Shattering Lemma

Lemma A.3 (The Shattering Lemma, [FG17, BEPS16]). Let G = (V, E) be a graph with maximum degree Δ . Consider a process that generates a random subset $B \subseteq V$ such that $P[v \in B] \leq \Delta^{-c_1}$, for some constant $c_1 \geq 1$, and such that the random variables $1(v \in B)$ depend only on the randomness of nodes within at most c_2 hops from v, for all $v \in V$, for some constant $c_2 \geq 1$. Then, for any constant $c_3 \geq 1$, satisfying $c_1 > c_3 + 4c_2 + 2$, we have that any connected component in G[B] has size at most $O(\log_{\Delta} n\Delta^{2c_2})$ with probability at least $1 - n^{-c_3}$.

Communication on Top of Weak Network Decompositions

Lemma A.4 ([GGR21, MU21]). Let G be a communication graph on n vertices. Suppose that each vertex of G is part of some cluster C such that each such cluster has a rooted Steiner tree $T_{\mathcal{C}}$ of diameter at most β and each node of G is contained in at most κ such trees. Then, in $O(\max\{1, \kappa/b\} \cdot (\beta + \kappa))$ rounds of the CONGEST model with b-bit messages, we can perform the following operations for all clusters in parallel on all clusters:

- 1. Broadcast: The root of $T_{\mathcal{C}}$ sends a b-bit message to all nodes in \mathcal{C} ;
- 2. Convergecast: We have O(1) special nodes $u \in C$, where each special node starts with a separate b-bit message. At the end, the root of T_C knows all messages;
- 3. Minimum: Each node $u \in C$ starts with a non negative b-bit number x_u . At the end, the root of T_C knows the value of $\min_{u \in C} x_u$;
- 4. Summation: Each node $u \in C$ starts with a non negative b-bit number x_u . At the end, the root of $T_{\mathcal{C}}$ knows the value of $\left(\sum_{u \in C} x_u\right) \mod 2^{O(b)}$.