Efficient CONGEST Algorithms for the Lovász Local Lemma

Yannic Maus
Technion, Haifa, Israel

Jara Uitto
Aalto University, Espoo, Finland

Abstract

We present a poly log log n time randomized CONGEST algorithm for a natural class of Lovász Local Lemma (LLL) instances on constant degree graphs. This implies, among other things, that there are no LCL problems with randomized complexity between $\Omega(\log n)$ and poly log log n. Furthermore, we provide extensions to the network decomposition algorithms given in the recent breakthrough by Rozhoň and Ghaffari [STOC2020] and the follow up by Ghaffari, Grunau, and Rozhoň [SODA2021]. In particular, we show how to obtain a large distance separated weak network decomposition with a negligible dependency on the range of unique identifiers.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases distributed graph algorithms, CONGEST, Lovász Local Lemma, locally checkable labelings

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.31

Related Version *Full Version*: https://arxiv.org/abs/2108.02638

Funding This project was partially supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 755839 (Yannic Maus).

1 Introduction

Our main contribution is a poly log log n round randomized distributed CONGEST algorithm to solve a natural class of Lovász Local Lemma (LLL) instances on constant degree graphs. Among several other applications, e.g., various defective graph coloring variants, this implies that there is no LCL problem (locally checkable labeling problem) with randomized complexity strictly between poly log log n and $\Omega(\log n)$, which together with the results of [3] implies that the *known world* of complexity classes in the sublogarithmic regime in the CONGEST and LOCAL model are almost identical. As a side effect of our techniques we extend the understanding of the computation of network decompositions in the CONGEST model. We now explain our contributions on LLL and LCLs in more detail; in the second half of the introduction we explain our techniques, contributions on network decomposition and how they relate to results in the LOCAL model.

We work in the CONGEST model of distributed computing, where a network of computational devices is abstracted as an n-node graph, where each node corresponds to a computational unit. In each synchronous round, the nodes can send messages of size $b = O(\log n)$ to their neighbors. In the end of the computation, each node is responsible of outputting its own part of the output/solution, e.g., its color in a graph coloring problem. The LOCAL model is otherwise the same, except that there is no bound on the message size b.

Background on LLL and LCLs in LOCAL

An *instance* of the *Lovász Local Lemma problem* is formed by a set of variables and a set of bad event $\mathcal{E}_1, \ldots, \mathcal{E}_n$ that depend on the variables. The famous Lovász Local Lemma [23] states that if the probability of each event is upper bounded by p, each event only shares variables with at most d other events and $epd < 1$ holds, then there exists an assignment to the variables that avoids all bad events. An example of a problem that can be solved via LLL is the *sinkless orientation* problem on graphs with minimum degree 4. In the sinkless orientation problem the objective is to orient the edges of a graph such that every vertex has at least one outgoing edge. This can be modeled by an LLL as follows: Orient each edge randomly – each edge represents a variable – and introduce a bad event for each vertex that holds if and only if all edges are oriented towards it. One obtains $p = 2^{-d}$ and $d \geq 4$ such that $epd < 1$ holds. The Lovász Local Lemma has had a huge success in theory of computation. One highlight is the beautiful and simple parallel algorithm by Moser and Tardos [39]. In the distributed version of the problem each variable and each event is simulated by one node of the communication network such that the variables of each event $\mathcal{E}$ are simulated by the node simulating $\mathcal{E}$ or one of its neighbors. The algorithm by Moser and Tardos immediately yields a randomized $O(\log^2 n)$ LLL algorithm in the LOCAL model. Often one has stronger guarantees on the relation of p and d, e.g., a polynomial criterion $epd^2 < 1$ (or even larger exponents) instead of only $epd < 1$, and can obtain simpler and faster algorithms. E.g., in the same model, Chung, Pettie, and Su obtained a randomized algorithm that runs in time $O(\log_{epd^2} n)$ whenever the *criterion* $epd^2 < 1$ holds [20].

LCLs through LLL: The main recent interest in distributed LLL are constant degree graphs motivated by the study of LCLs. LCL problems are defined on constant degree graphs and in an *LCL problem* each node is given an input from a constant sized set of labels and must output a label from a constant sized set of labels. The problem is characterized by a set of feasible constant-radius neighborhoods (for a formal definition see Section 4). Many classic problems are LCL problems, e.g., the problem of finding a vertex coloring with Δ colors in a graph with maximum degree Δ. The systematic study of LCLs in the LOCAL model initiated by Naor & Stockmeyer [40] has picked up speed over the last years leading to an almost complete classification of the complexity landscape of LCL problems in the LOCAL model [18, 19, 4, 2, 43]. One of the most important results in this line of research is by Chang and Pettie [19] who showed that any $o(\log n)$-round randomized LOCAL algorithm for an LCL problem P implies the existence of a T_{LLL}-round algorithm for P where T_{LLL} is the runtime of an LLL algorithm with an (arbitrary) polynomial criterion, e.g., for $p(ed)^{100}$. Among other implications, the breakthrough result by Rozhoň and Ghaffari provided an $(\operatorname{poly}\log\log n)$-round LLL algorithm, in the LOCAL model, for the case that the dependency degree d is bounded. This implies a gap in the randomized complexity landscape for LCLs. There is no LCL problem with a complexity strictly between $\operatorname{poly}\log\log n$ and $\Omega(\log n)$ in the LOCAL model.

1.1 Our Results on the Distributed Lovász Local Lemma and LCLs

Motivated by the progress in LOCAL using the *LCLs through LLL* application, we aim to design CONGEST algorithms for the LLL problem. The first observation is that one wants to limit the range of the variables, as any reasonable algorithm should be able to send the value of a variable in a CONGEST message, and one also wants to limit the number of variables at a node such that nodes can learn assignments of all variables associated with their bad event(s) efficiently. We call an LLL instance with dependency degree d *range bounded* if each

event only depends on poly d variables and each variable uses values from a range of size poly d. At first sight, this seems like a a very restrictive setting, but in fact most instances of LLL satisfy these requirements. Our main result is contained in the following theorem and meets the poly log log n round state of the art in the LOCAL model [43].

▶ **Theorem 1.** *There is a randomized* CONGEST *algorithm that with high probability solves any range bounded Lovász Local Lemma instance that has at most n bad events, dependency degree $d = O(1)$ and satisfies the LLL criterion $p(ed)^8 < 1$ where p is an upper bound on the probability that a bad event occurs, in* poly log log n *rounds.*

On the negative side, it is know that a double logarithmic dependency cannot be avoided. There is an $\Omega(\log \log n)$-round lower bound for solving LLL instances with randomized algorithms in the LOCAL model [11], that holds even for constant degree graphs, and it naturally applies to the CONGEST model as well. It stems from a lower bound on the aforementioned sinkless orientation problem.

Implications for the Theory of LCLs

As the reduction from [19], that reduces sublogarithmic time randomized algorithms to LLL algorithms, immediately works in CONGEST, our fast LLL algorithm implies a gap in the complexity landscape of LCLs. A more precise definition of LCLs and a proof for the following corollary will be presented in the end of Section 4.

▶ **Corollary 2.** *There is no LCL problem with randomized complexity strictly between* poly log log n *and* $\Omega(\log n)$ *in the* CONGEST *model.*

In fact, Corollary 2 together with the results of [3] implies that the *known world* of complexity classes in the sublogarithmic regime in the CONGEST and LOCAL model are almost identical. In fact, a difference can only appear in the extremely small complexity regime between $\Omega(\log \log^* n)$ and $O(\log^* n)$ and in the important regime of complexities that lie between $\Omega(\log \log n)$ and poly log log n where complexity classes are not understood in the LOCAL model. An immediate implication of Corollary 2 is a poly log log n-round randomized CONGEST algorithm for Δ-coloring when Δ is constant, a result that was previously only known in the LOCAL model [29]. Chang and Pettie [19] conjecture that the runtime of LLL in the LOCAL model is $O(\log \log n)$ on general bounded degree graphs which would further simplify the complexity landscape for LCLs. On trees this complexity can be obtained [17] in the LOCAL model and for the specific LLL instances that arise in the study of LCLs on trees an $O(\log \log n)$-round algorithm has also been found in the CONGEST model [3].

1.2 Technical Overview and Background on our Methods

Our core technical contribution to obtain Theorem 1 is a bandwidth efficient derandomization of the LLL algorithm by Chung, Pettie, and Su [20] that we combine with the shattering framework of Fischer and Ghaffari [24]. To explain these ingredients and how we slightly advance our understanding of network decompositions in the CONGEST model on the way, we begin with explaining the background on distributed derandomization and network decompositions and the relation to results in the LOCAL model.

Background on Network Decompositions and Distributed Derandomization

Network decompositions are a powerful tool with a range of applications in the area of distributed graph algorithms and were introduced by Awerbuch, Luby, Goldberg, and Plotkin [1]. A (C, D)-*network decomposition* (ND) is a partition of the vertices of a graph

into C collections of clusters (or color classes of clusters) such that each cluster has diameter[1] at most D. Further, it is required that the clusters in the same collection are *independent*, i.e., are not connected by an edge. This is extremely helpful in the LOCAL model, e.g., to compute a $(\Delta+1)$-coloring of the network graph one can iterate through the C collections, and within each collection process all clusters in parallel (due to their independence). Due the unbounded message size dealing with a single cluster is trivial. A cluster leader can learn all information about the cluster in time that is proportional to the cluster diameter D, solve the problem locally and disseminate the solution to the vertices of the cluster. Thus the runtime scales as $O(C \cdot D)$. Hence, the objective has been to compute such decompositions as fast as possible and with C and D as small as possible, optimally, all values should be polylogarithmic in n. Awerbuch et al. gave a deterministic LOCAL algorithm with round complexity and C and D equal to $2^{O(\sqrt{\log n \log\log n})} \gg \operatorname{poly}\log n$. Panconesi and Srinivasan improved both parameters and the runtime to $2^{O(\sqrt{\log n})}$ [41]. Linial and Saks showed that the optimal trade-off between diameter and the number of colors is $C = D = O(\log n)$ and they provided an $O(\log^2 n)$-round randomized algorithm to compute such decompositions [38] . These algorithms remained the state of the art for almost three decades even though the need for an efficient deterministic algorithm for the problem has been highlighted in many papers, e.g., [7, 32, 27, 19, 31].

A few years ago, Ghaffari, Kuhn, and Maus [33] and Ghaffari, Harris, and Kuhn [28] highlighted the importance of network decompositions by showing that an efficient deterministic LOCAL algorithm for network decompositions with $C = D = \operatorname{poly}\log n$ would immediately show that an efficient randomized LOCAL algorithm for any efficiently verifiable graph problem would yield an efficient deterministic algorithm. Here, all occurrences of *efficient* mean polylogarithmic in the number of nodes of the network. Then, in the aforementioned breakthrough Rozhoň and Ghaffari [43] devised such an efficient deterministic algorithm for network decompositions, yielding efficient deterministic algorithms for many problems. The result also had an immediate impact on randomized algorithms. Many randomized algorithms in the area use the *shattering technique* that at least goes back to Beck [10] who used the technique to solve LLL instances in centralized settings. It has been introduced to distributed computing by Barenboim, Elkin, Pettie and Schneider in [9]. The shattering technique usually implements the following schematic: First, the nodes use a randomized process and the guarantee is that with high probability *almost* all nodes find a satisfactory output. The remainder of the graph is *shattered* into *small components*, that is, after this so called *pre-shattering* phase the unsolved parts of the graph induce small – think of $N = \operatorname{poly}\log n$ size – connected components. In the *post-shattering phase* one wishes to use an efficient deterministic algorithm, e.g., a deterministic algorithm with complexity $T(n) = \operatorname{poly}\log n$ applied to each small component results in a complexity of $T(N) = T(\operatorname{poly}\log\log n)$. Combining the shattering technique, e.g., [9, 24, 25], the network decomposition algorithm from [43] and the derandomization from [32, 27] the randomized complexities for many graph problems in the LOCAL model are $\operatorname{poly}\log\log n$.

[1] For the sake of this exposition it is enough to assume that a cluster $\mathcal{C}$ has diameter D, that is, any two vertices in the cluster are connected with a path within the cluster of length at most D. Actually, often these short paths are allowed to leave the cluster which may cause congestion when one uses these paths for communication in two clusters in parallel. The details of the standard way to model this congestion and its impact are discussed in Section 3.

The Challenges in the CONGEST Model

The holy grail would be to obtain a similar derandomization result in the CONGEST model. But, even though the network decomposition algorithm from [43] immediately works in CONGEST, we are probably far from obtaining such a result. Even for simple problems like computing a maximal independent set or a $(\Delta+1)$-coloring one has to work much harder to even get $\operatorname{poly}\log n$ round deterministic algorithms in the CONGEST model, even if a $(\log n, \log n)$-network decomposition is given for free [15, 5]. For LLL obtaining such a bandwidth efficient algorithm seems much harder. Even in the LOCAL model one either has to go through the aforementioned derandomization result or one can solve an LLL instance with criterion $p(ed)^\lambda$, e.g., think of $\lambda = 10$, if the network decomposition only has λ color classes. This immediately implies [38] that the cluster diameter is $\Omega(n^{1/\lambda})$ and algorithms that rely on aggregating all information about a cluster in a cluster leader must use at least $\Omega(n^{1/\lambda})$ rounds. In the LOCAL model there are black box reductions [32, 43, 8] to compute such decompositions. We provide an analysis of the algorithm by Rozhoň and Ghaffari (and of another algorithm by Ghaffari, Grunau and Rozhoň [26]) where we carefully study the trade-off between number of colors, the diameter of the clusters, and the runtime of the algorithm to obtain such decompositions in CONGEST.

▶ **Theorem 8** (informal). *For $k \geq 1$ and $1 \leq \lambda < \log n$ there is an $O(k \cdot n^{1/\lambda} \operatorname{poly}\log n)$-round deterministic* CONGEST *algorithm to compute a $(\lambda, (k \cdot n^{1/\lambda} \operatorname{poly}\log n)$-network decomposition such that any two clusters with the same color have distance strictly more than k. The algorithm works with a mild dependence on the ID space*[2].

For $k = 1$, decompositions with few colors similar to Theorem 8 could already be obtained (using randomization) through the early works by Linial and Saks, who also showed that their trade-off of the number of colors and the cluster diameter in Theorem 8 is nearly optimal [38]. In fact, Theorem 8 does not just provide an improved analysis of the algorithm of [43] but it also comes with a mild dependence on the identifier space and an arbitrary parameter to increase the distance between clusters. Both ingredients are also present in our second result on network decompositions (Theorem 6) where they are crucial for the proof of Theorem 1.

Our Solution: Range bounded LLLs in CONGEST. We use the shattering framework for LLLs of [24] whose pre-shattering phase, as we show, works in the CONGEST model for range bounded LLLs. As a result we obtain small remaining components of size $N \ll n$, in fact, we obtain $N = O(\log n)$. Furthermore, one can show that the remaining problem that we need to solve on the small components is also an LLL problem. We solve these small instances via a bandwidth efficient derandomization of the LLL algorithm by Chung, Pettie, Su [20]. Note that we cannot solve the small components without derandomizing their algorithm, as running their randomized algorithm on each component for $T(N) = O(\log N) = O(\log\log n)$ rounds would imply an error probability of $1/N$ which is exponentially larger than the desired high probability guarantee of $1/n$. Thus, we desire to find *good random bits* for all nodes with which we can execute the algorithm from [20] without any error. The goal is to use a network decomposition algorithm to partition the small components into $O(\log N) = O(\log\log n)$ collections of independent clusters. Then, we iterate through the collections and want to obtain good random bits for the vertices inside each cluster. Since the randomized runtime of [20] on a small component would be $T(N) = O(\log N) = O(\log\log n)$, we observe that the

[2] To be precise, the dependency on an ID space $\mathcal{S}$ space is a $\log^* |\mathcal{S}|$ factor.

random bits of a node v cannot influence the correctness at a node u if u and v are much further than $T(N)$ hops apart. Thus, similar to Theorem 8 we devise the following theorem to compute network decompositions with large distances between the clusters. In fact, if we apply the theorem with $k > 2 \cdot T(N)$, we obtain independent clusters in each color class of the decomposition.

▶ **Theorem 6** (informal). *For $k \geq 1$ there is an $O(k \cdot \text{poly} \log n)$-round deterministic* CONGEST *algorithm to compute a $(\log n, \text{poly} \log n)$-network decomposition such that any two clusters with the same color have distance strictly more than k. The algorithm works with a mild dependence on the ID space.*

While [43] provided a modification of their algorithm that provides a larger distance between clusters its runtime and cluster diameter depend polylogarithmically not only on the number of nodes in the network but also on the ID space. In [26] Ghaffari, Grunau and Rozhoň have reduced the ID space in the special case in which the cluster distance k equals one. However, our bandwidth efficient LLL algorithm requires $k \gg 1$ and identifier independence at the same time. Thus, one can either say we add the ID space independence to [43] or we extend the results of [26] to $k > 1$.

We already explained why we require that we obtain a network decomposition with a large cluster distance to derandomize an algorithm. The mild dependence on the identifier space in Theorem 6 is also essential as the small components with N nodes on which we want to use the network decomposition algorithm, actually live in the original communication network G. Thus they are equipped with identifiers that are polynomial in n, that is, exponential in N. The fact that the small components live in the large graph makes our life harder when computing a network decomposition but it helps us when designing efficient CONGEST algorithms. The bandwidth when executing an algorithm on the small components is still $O(\log n)$ bits per edge per round while the components are of size $N \ll n$. Ignoring details for the sake of this exposition, we use the increased bandwidth to gather enough information about a cluster in a single cluster leader such that it can select good random bits for all nodes of the cluster and in parallel with all other clusters of the same color class due to the large distance between clusters.

We emphasize that there have been several other approaches to derandomize algorithms in the CONGEST model and discussing all of them here would be well beyond the scope of this work. However, we still believe that our derandomization technique in the post-shattering phase might be of independent interest as it applies to a more general class of algorithms than just the aforementioned LLL algorithm from [20].

Other Implications. As the runtime of Theorem 6 is a $\log n$ factor faster than the result in [43], it also improves the complexity of deterministic distance-2 coloring with $(1+\varepsilon)\Delta^2$ colors in the CONGEST model to $O(\log^7 n)$ rounds when plugged into the framework of [35]. Similar improvements carry over to the approximation of minimum dominating sets [21] and spanner computations [30]. Due to the identifier independence and possibility to increase the distance between clusters Theorem 6 yields an improved randomized complexity of distance-2 coloring. Using Theorem 6 in the shattering based algorithm from [36] improves the runtime for $\Delta^2 + 1$ colors from $2^{O(\sqrt{\log \log n})}$ to $O(\log^7 \log n)$ rounds.

Roadmap

In Section 1.3, we provide pointers for further related work, mainly on network decompositions. In Section 2 we define the models and introduce notation. In Section 3 we formally state the result on network decompositions and indicate the main changes to prior work. Due to space limitations, the formal proofs of these results appear in the full version of the paper. The main part of the paper, Section 4, deals with proving Theorem 1.

1.3 Further Related Work

We already mentioned that [26] provides an efficient deterministic CONGEST algorithm with a mild dependence on the ID space. More precisely, they provided a $(O(\log n), \log^2 n))$-network decomposition in $O(\log^5 n + \log^4 n \cdot \log^* b)$ rounds, where $\log^* b$ is the dependency on the identifier length b [26]. One drawback that we have ignored until now – and that also applies to all of our results – is that these network decompositions only have so called *weak diameter*, that is, the diameter of a cluster is only guaranteed to be small if it is seen as a subset of the communication network G, that is, the distance between two vertices is measured in G and not only in the subgraph induced by a cluster. In contrast, in so called *strong network decompositions* the diameter in the subgraph of G that is induced by each cluster has to be small. Very recently, at the cost of increasing the polylogarithmic factors in the runtime the results of [26] were extended to obtain strong $(O(\log n), \log^2 n))$-network decompositions [16]. Earlier works by Elkin and Neiman provided an $O(\log^2 n)$ randomized algorithm for computing strong $(O(\log n), O(\log n))$-decompositions [22].

Most previous works in the CONGEST model do not ensure large distances between clusters. Besides the result in [43] that we have already discussed there are two notable exceptions. Ghaffari and Kuhn matched the complexity of the LOCAL model algorithm by Awerbuch et al. by giving a deterministic $k \cdot 2^{O(\sqrt{\log n \log\log n})}$ round algorithm [30]. Later, this was improved to $k \cdot 2^{O(\sqrt{\log n})}$ rounds by Ghaffari and Portmann [34]. In both cases, the decomposition parameters were identical to the runtimes. None of these results is the state of the art anymore, except for the fact that they compute strong network decompositions with large distances between the clusters.

Barenboim, Elkin, and Gavoille provide various algorithms with different trade-offs between the number of colors and the cluster diameter, most notably a randomized algorithm to compute a strong network decomposition with diameter $O(1)$ and $O(n^\varepsilon)$ colors [8]. A similar result with $O(n^{1/2+\varepsilon})$ colors was obtained by Barenboim in [6].

Brandt, Maus, and Uitto and Brandt, Grunau, and Rozhoň have shown that LLL instances with an exponential LLL criterion, that is, $p2^d < 1$ holds, can be solved in $O(\log^* n)$ rounds on bounded degree graphs [14, 12]. Furthermore, it is known that $\Omega(\log^* n)$ rounds cannot be beaten under any LLL criterion that is a function of the dependency degree d [20]. For an exponential criterion that satisfies $p2^d \geq 1$, it follows from the works of Brandt et al. and Chang, Kopelowitz, and Pettie [18] that there is a lower bound of $\Omega(\log n)$ rounds for deterministic LLL. Hence [14, 12] and [18] show that there is a sharp transition of the distributed complexity of LLL at $2p^d = 1$. Before the result in [43] improved the runtime to $\operatorname{poly}\log\log n$, Ghaffari, Harris, and Kuhn gave the state of the art randomized LLL algorithm in the LOCAL model. On constant degree graphs its runtime was described by a tower function and lies between $\operatorname{poly}\log\log n$ and $2^{O(\sqrt{\log\log n})}$ [27]. We are not aware of any works that explicitly studied LLL in the CONGEST model before our paper.

2 Models, LCLs & Notation

Given a graph $G = (V, E)$ the hop distance in G between two vertices $u, v \in V$ is denoted by $dist_G(u, v)$. For a vertex $v \in V$ and a subset $S \subseteq V$ we define $dist_G(v, S) = \min_{u \in S}\{dist(v, u)\} \in \mathbb{N} \cup \{\infty\}$. For two subsets $S, T \subseteq V$ we define $dist_G(S, T) = \min_{v \in T} dist_G(v, S)$. For an integer n we denote $[n] = \{0, \ldots, n-1\}$.

The LOCAL and CONGEST Model of distributed computing [37, 42]. In both models the graph is abstracted as an n-node network $G = (V, E)$ with maximum degree at most Δ. Communication happens in synchronous rounds. Per round, each node can send one

message to each of its neighbors. At the end, each node has to know its own part of the output, e.g., its own color. In the LOCAL model there is no bound on the message size and in the CONGEST model messages can contain at most $O(\log n)$ bits. Usually, in both models nodes are equipped with $O(\log n)$ bit IDs (polynomial ID space) and initially, nodes know their own ID or their own color in an input coloring but are unaware of the IDs of their neighbors. Randomized algorithms do not use IDs (but they can create them from space $[n^c]$ for a constant c and with a $1/n^c$ additive increase in the error probability), have a fixed runtime and need to be correct with probability strictly more than $1 - 1/n$ (Monte Carlo algorithm). Actually, algorithms are defined such that a parameter n is provided to them, where n represents an upper bound on the number of nodes of the graph and the algorithm's guarantees have to hold on any graph with at most n nodes. For technical reasons in our gap results, we assume that the number of random bits used by each node is bounded by some finite function $h(n)$. We note that the function $h(n)$ can grow arbitrarily fast and that this assumption is made in previous works as well [18, 3].

▶ **LCL definition** ([40]). An *LCL problem* Π is a tuple $(\Sigma_{\text{in}}, \Sigma_{\text{out}}, F, r)$ satisfying the following.

- Both Σ_{in} and Σ_{out} are constant-size sets of labels,
- the parameter r is an arbitrary constant, called the *checkability radius* of Π,
- F is a finite set of pairs $(H = (V^H, E^H), v)$, where:
  - H is a graph, v is a node of H, and the radius of v in H is at most r;
  - Every pair $(v, e) \in V^H \times E^H$ is labeled with a label in Σ_{in} and a label in Σ_{out}.

In an instance of an LCL problem $\Pi = (\Sigma_{\text{in}}, \Sigma_{\text{out}}, F, r)$ on a graph $G = (V, E)$ each node receives an *input label* from Σ_{in}. An algorithm solves Π if, for each node $v \in V$, the radius-r hop neighborhood of v together with the input labels and computed outputs for the nodes lies in the set of *feasible labelings* F.

3 Graph Decompositions

In this section we state our results on network decompositions and provide the respective definitions. At the end of the section we provide a short overview of the techniques that we use to prove the results. The formal proofs appear in the full version of the paper.

3.1 Cluster Collections and Network Decompositions with Congestion

Classically, one defines a *(weak)* (c, d)*-network decomposition* as a coloring of the vertices of a graph G with c colors such that the connected components of each color class have weak diameter at most d. The *weak diameter* of a subset $\mathcal{C} \subseteq V$ is the maximum distance in G that any two vertices of $\mathcal{C}$ have.

We use the following definition that augments one color class of a network decomposition with a communication backbone for each cluster. A *cluster* $\mathcal{C}$ of a graph is a subset of nodes. A *Steiner tree* is a rooted tree with nodes labeled as *terminal* and *nonterminal*.

▶ **Definition 3** (cluster collection). *A collection of clusters of a graph $G = (V, E)$ consists of subsets of vertices $\mathcal{C}_1, \ldots, \mathcal{C}_p \subseteq V$ and has Steiner radius β and unique b-bit cluster identifiers if it comes with associated Steiner subtrees $T_1, \ldots, T_p$ of G such that clusters are disjoint, i.e., $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for $i \neq j$ and for each $i \in 1, 2, \ldots, p$ we have*

1. *cluster $\mathcal{C}_i$ has a unique b-bit identifier $\text{id}_{\mathcal{C}_i}$,*
2. *the terminal nodes of Steiner tree T_i of cluster $\mathcal{C}_i$ are formed by $\mathcal{C}_i$ (T_i might contain nodes $\notin \mathcal{C}_i$ as non terminal nodes),*
3. *the edges of Steiner Tree T_i are oriented towards a* cluster leader *$\ell_{\mathcal{C}_i}$,*
4. *Steiner tree T_i has diameter at most β (Steiner tree diameter).*

The collection has congestion κ *if each edge in* E *is contained in at most* κ *Steiner trees. When we compute a cluster collection in the* CONGEST *model, we require that each node of a cluster knows the cluster identifier. Furthermore, we require that for each edge of* T_i *that is incident to some* $v \in V$*, node* v *knows* $id_{\mathcal{C}_i}$ *and additionally* v *knows the direction of the edge towards the root* $r_{\mathcal{C}_i}$*.*

We now define our notion of a network decomposition with a communication backbone.

▶ **Definition 4** (network decomposition with congestion)**.** *Let* $k \geq 1$*. A* (C, β)-network decomposition *with cluster distance* k *and* congestion κ *of a graph* $G = (V, E)$ *is a partition of* V *into* C *cluster collections with Steiner radius* β *and congestion* κ *such that any two clusters* $\mathcal{C} \neq \mathcal{C}'$ *in the same cluster collection have distance strictly more than* k*, i.e.,* $dist_G(\mathcal{C}, \mathcal{C}') > k$*.*

The C *cluster collections are also called the* C color classes *of the decomposition.*

Instead of a network decomposition with cluster distance k, we often just speak of a network decomposition of G^k. In all cases the Steiner trees are given by vertices and edges in G and not by edges in G^k, which would corresponds to paths in G. The default value for k is 1 which we use whenever we do not specify its value. in this case the definition corresponds with the classic network decomposition.

3.2 Network Decomposition and Ball Carving Algorithms

At the core of our network decomposition algorithm is the following *ball carving* result to form one color class of the decomposition, i.e., to form one cluster collection. The name ball carving stems from the original existential proof for network decompositions [1].

▶ **Lemma 5** (ball carving, $k \geq 1$)**.** *Let* $k \geq 1$ *be an integer and* $x \geq 1$ *(both potentially functions of* n*). Then there is a deterministic distributed* CONGEST *(bandwidth* b *and* b*-bit identifiers) algorithm that, given a graph* $G = (V, E)$ *with at most* n *nodes and a subset* $S \subseteq V$*, computes a cluster collection* $\mathcal{C}_1, \ldots, \mathcal{C}_p \subseteq S$ *with*

- $|\mathcal{C}_1 \cup \ldots \cup \mathcal{C}_p| \geq (1 - 1/x) \cdot |S|$*,*
- *Steiner radius* $\beta = O(k \cdot x \cdot \log^3 n)$ *and congestion* $\kappa = O(\log n \cdot \min\{k, x \cdot \log^2 n\})$*,*
- *the pairwise distance in* G *between* $\mathcal{C}_i$ *and* $\mathcal{C}_j$ *for* $1 \leq i \neq j \leq p$ *is strictly more than* k*.*

The runtime is $O(k \cdot x \cdot \log^4 n \cdot \log^* b) + O(k \cdot x^2 \cdot \log^6 n \cdot \min\{k, x \cdot \log^2 n\})$ *rounds.*

By using the aggregation tools for overlapping Steiner trees of [26] Lemma 5 is by a $\log n$ factor faster than the corresponding result in [43]. The ball carving result immediately imply the following network decomposition results.

▶ **Theorem 6** ($O(\log n)$ colors)**.** *For any (potentially non constant)* $k \geq 1$*, there is a deterministic* CONGEST *algorithm with bandwidth* b *that, given a graph* G *with at most* n *nodes and unique* b*-bit IDs from an exponential ID space, computes a (weak)* $(O(\log n), O(k \cdot \log^3 n))$*-network decomposition with cluster distance* k *and with congestion* $O(\log^2 n \cdot \min\{k, \log^2 n\})$ *in* $O(k \cdot \log^7 n \cdot \min\{k, \log^2 n\})$ *rounds.*

We also analyze a more involved ball carving algorithm of [26] for different parameters in order to obtain decompositions with fewer colors. Then Lemmas 5 and 7 imply the following result for network decompositions with few colors.

▶ **Lemma 7** (faster ball carving, $k = 1$)**.** *Let* $x \geq 1$ *(potentially a function of* n*). Then there is a deterministic distributed* CONGEST *(bandwidth* b *and* b*-bit identifiers) algorithm that, given a graph* $G = (V, E)$ *with at most* n *nodes and a subset* $S \subseteq V$*, computes a cluster collection* $\mathcal{C}_1, \ldots, \mathcal{C}_p \subseteq S$ *with*

- $|\mathcal{C}_1 \cup \ldots \cup \mathcal{C}_p| \geq (1 - 1/x) \cdot |S|$,
- *Steiner radius* $\beta = O(x \cdot \log^2 n)$ *and congestion* $\kappa = O(\log n)$,
- *the pairwise distance in* G *between* $\mathcal{C}_i$ *and* $\mathcal{C}_j$ *for* $1 \leq i \neq j \leq p$ *is strictly more than* 1.

The runtime of the algorithm is $O(x^2 \log^4 n)$ *rounds.*

▶ **Theorem 8** (few colors). *For any* $\lambda \leq \log n$ *there is a deterministic* CONGEST *algorithm with bandwidth* b *that, given a graph* G *with at most* n *nodes and unique* b*-bit IDs from an exponential ID space, computes a weak* $(\lambda, n^{1/\lambda} \log^2 n)$*-network decomposition of* G *in* $O(\lambda \cdot n^{2/\lambda} \cdot \log^4 n)$ *rounds with congestion* $\kappa = O(\log n)$.

For any (possibly non constant) $k \geq 1$ *and any* $\lambda \leq \log n$ *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 *in* $O(k \cdot n^{2/\lambda} \cdot \log^6 n \cdot \min\{k, x \log^2 n\})$ *rounds.*

Theorems 6 and 8 also work with more general ID spaces, as long as the IDs fit into a CONGEST message. To simplify the statements and because exponential IDs are (currently) the main application we do not explicitly state their mild dependence on the size of the ID space $\mathcal{S}$ in theorems. It can be quantified by $O(x \cdot \text{poly} \log n \cdot \log^* |\mathcal{S}|)$ where the polylogarithmic terms are strictly dominated by the ones in the theorems.

Summary of Our Ball Carving Contributions. Both, the ball carving results in [43] and [26] begin with each vertex of the to be partitioned set S (see Lemmas 5 and 7) being its own cluster inheriting its node ID as the cluster ID. Then they implement a distributed *ball growing* approach in which vertices leave their cluster, join other clusters, or become dead, i.e., are disregarded; even whole clusters can dissolve, e.g., if all their vertices join a different cluster. The goal is to ensure that at the end no two alive clusters are neighboring, that is, in distance k, that the fraction of vertices of S that are declared dead is small and that the diameter of the clusters does not become too large. In fact, as vertices can leave clusters one only obtains guarantees on the weak diameter of clusters. To this end the algorithm of [43] iterates through the bits of the cluster IDs and in phase i, vertices change their clusters or become dead such that at the end of the i-th phase no two neighboring clusters have the same i-th bit in their cluster ID. If cluster IDs have b bits, then after the b-th phase each remaining cluster is not connected to any other cluster. Thus, the runtime crucially depends on the size of the cluster IDs. The main change in [26] to remove this ID space dependence is to replace the bits (0 or 1) used in phase i with a coloring of the clusters with two colors, red and blue. A new coloring is computed in every phase and the crucial property of the coloring that ensures progress towards the separation of the clusters is that each connected component of clusters has roughly the same number of red and blue clusters. Thus, intuitively, at the end of a phase red and blue clusters are separated and the sizes of all connected components decrease.

In our algorithms we follow the same high level approach but extend the techniques (for identifier "independence") to work with a larger separation between the clusters, and also for the computation of decompositions with fewer colors. A crucial difficulty is that we cannot quickly disregard all dead vertices and reason that each connected components of clusters in G can be treated independently. It might even be that two clusters are connected by a path with $\leq k$ hops and the path might contain dead vertices. Instead, we consider connected components in G^k and always keep all vertices of G in mind. As two clusters that are adjacent in such a component might not be adjacent in the original graph we ensure that no congestion appears *in between* the clusters. Careful algorithm design and reasoning is needed when computing a balanced coloring of clusters in such components.

To obtain the network decompositions with fewer colors, e.g., with $\lambda = 10$ colors, we analyze our algorithm for a suitable choice of parameters and show that all congestion parameters are not affected by the choice of λ.

4 Distributed Range Bounded LLL and its Implications

The objective of this section is to prove the following theorem.

▶ **Theorem 1.** *There is a randomized* CONGEST *algorithm that with high probability solves any range bounded Lovász Local Lemma instance that has at most n bad events, dependency degree $d = O(1)$ and satisfies the LLL criterion $p(ed)^8 < 1$ where p is an upper bound on the probability that a bad event occurs, in* $\operatorname{poly}\log\log n$ *rounds.*

Distributed Lovász Local Lemma (LLL). In a distributed Lovász Local Lemma instance we are given a set of independent random variables $\mathcal{V}$ and a family $\mathcal{X}$ of (bad) events $\mathcal{E}_1, \ldots, \mathcal{E}_n$ on these variables. Each bad event $\mathcal{E}_i \in \mathcal{X}$ depends on some subset $vbl(\mathcal{E}_i) \subseteq \mathcal{V}$ of the variables. Define the dependency graph $H_{\mathcal{X}} = (\mathcal{X}, \{(\mathcal{E}, \mathcal{E}') \mid vbl(\mathcal{E}) \cap vbl(\mathcal{E}') \neq \emptyset\})$ that connects any two events which share at least one variable. Let Δ_H be the maximum degree in this graph, i.e., each event $\mathcal{E} \in \mathcal{X}$ shares variables with at most $d = \Delta_H$ other events $\mathcal{E}' \in \mathcal{X}$. The Lovász Local Lemma [23] states that $Pr(\bigcap_{\mathcal{E}\in\mathcal{X}} \mathcal{E}) > 0$ if $epd < 1$. In the *distributed LLL problem* each bad event and each variable is assigned to a vertex of the communication network such that all variables that influence a bad event $\mathcal{E}$ that is assigned to a vertex v are either assigned to v or a neighbor of v. The objective is to compute an assignment for the variables such that no bad event occurs. At the end of the computation each vertex v has to know the values of all variables that influence one of its bad events (by definition these variables are assigned to v or one of its neighbors). In all results on distributed LLL in the LOCAL model that are stated below the communication network is identical to the dependency graph H. Most result in the distributed setting require stronger LLL criteria.

Algorithm 1 The algorithm from [20] iteratively resamples all variables of local ID minima of violated events. IDs can be assigned in an adversarial manner. The runtime is with high probability $O(\log_{epd^2} n)$ rounds under LLL criterion $epd^2 < 1$.

Initialize a random assignment to the variables
Let $\mathcal{F}$ be the set of bad events under the current variable assignment
while $\mathcal{F} \neq \emptyset$ **do**
  Let $I = \left\{A \in \mathcal{F} \mid ID(A) = \min\{ID(B) | B \in N^+_{\mathcal{F}}(A)\}\right\}$
  Resample $vbl(I) = \cup_{A\in I} vbl(A)$
end

Despite its simplicity – the algorithm simply iteratively resamples local ID minima of *violated events*, i.e., bad events that hold under the current variable assignment – Algorithm 1 results in the following theorem.

▶ **Theorem 9** ([20], Algorithm 1)**.** *Given an LLL instance with condition $epd^2 < 1$, the* LOCAL *Algorithm 1 run for $O(\log_{epd^2} n)$ rounds and has error probability $< 1/n$ on any LLL instance with at most n bad events.*

▶ **Definition 10** (Range bounded Lovász Local Lemma)**.** *An instance $(\mathcal{V}, \mathcal{X})$ of the Lovász Local Lemma with bad events $\mathcal{E}_1, \ldots, \mathcal{E}_n$ and dependency graph H is* range bounded *if $|vbl(\mathcal{E}_i)| = \operatorname{poly}\Delta_H$ for each $1 \leq i \leq n$ and the value of each random variable $x \in \mathcal{V}$ can be expressed by at most $O(\log \Delta_H)$ bits.*

Note that most applications of the Lovász Local Lemma in the distributed form are range bounded instances. Algorithm 1 can be executed with the same asymptotic runtime for range bounded LLL instances and uses few random bits while doing so.

▶ **Lemma 11** (Proof deferred to the full version). *For a range bounded LLL instance with condition $epd^2 < 1$ with constant dependency degree there is a randomized* CONGEST *(constant bandwidth) algorithm that runs in $O(\log_{epd^2} n)$ rounds and has error probability $< 1/n$ on any (dependency) graph with at most n nodes. Furthermore, in the algorithm each node only requires access to $O(\log n)$ random values from a bounded range. The unique IDs can be replaced by an acyclic orientation of the edges of the graph.*

The currently fastest randomized algorithm for LLL on bounded degree graphs with polynomial criterion in the LOCAL model is based on two main ingredients. The first ingredient is a deterministic $\operatorname{poly}\log n$-round LLL algorithm obtained via a derandomization (cf. [32, 27]) of the $O(\log^2 n)$-round randomized algorithm [39] using the breakthrough efficient network decomposition algorithm [43]. The second ingredient is the (randomized) shattering framework for LLL instances by [24], which *shatters* the graph into small unsolved components of logarithmic size. Then, these components are solved independently and in parallel with the deterministic algorithm.

We follow the same general approach of shattering the graph via the methods of [24], but need to be more careful to obtain a bandwidth efficient algorithm when dealing with the small components. We begin with a lemma that describes how fast we can gather information in a cluster leader with limited bandwidth. Its proof uses standard pipelining techniques (see e.g. [42, Chapter 3] and fills each b-bit messages with as much information as possible.

▶ **Lemma 12** (Token learning). *Let G be a communication graph on n vertices in which each node can send b bits per round on each edge. Assume a cluster collection with Steiner radius β and congestion κ in which each cluster $\mathcal{C}$ is of size at most N and each vertex holds at most x bits of information. Then, in parallel each cluster leader $\ell_{\mathcal{C}}$ can learn the information of each vertex of $\mathcal{C}$ in $O\big(\kappa \cdot (\beta + N \cdot x/b)\big)$ rounds. In the same runtime the leader $\ell_{\mathcal{C}}$ can disseminate x bits of distinct information to each vertex of $\mathcal{C}$.*

Next, we prove our core derandomization result that we use to obtain efficient deterministic algorithms for the small components that arise in the post-shattering phase. It might be of independent interest.

▶ **Lemma 13** (Derandomization). *Consider an LCL problem P, possibly with promises on the inputs. Assume a $T(n)$-round randomized* CONGEST *(bandwidth $b = \Theta(\log n)$) algorithm $\mathcal{A}$ for P with error probability $< 1/n$ on any graph with at most n nodes that uses at most $O(\log n)$ random bits per node.*

Then for any graph on at most N nodes there is a deterministic $T(N) \cdot \operatorname{poly}\log N$ round CONGEST *(with bandwidth $b = \Theta(N)$) algorithm $\mathcal{B}$ to solve P under the same promises, even if the ID space is exponential in N.*

Proof. For an execution of algorithm $\mathcal{A}$ on a graph with N nodes define for each $v \in V$ an indicator variable X_v that equals to 1 if the verification of the solution fails at node v and 0 otherwise. The value of X_v depends on the randomness of nodes in the $(T(N) + r)$ hop neighborhood of v, where $r = O(1)$ is defined by the LCL problem.

We design an efficient algorithm $\mathcal{B}$ that deterministically fixes the (random bits) of each node $v \in V$ such that $X_v = 0$ for all nodes. Thus, executing $\mathcal{A}$ with these random bits solves problem P. During the execution of $\mathcal{B}$ we fix the random bits of more and more

nodes. At some point during the execution of algorithm $\mathcal{B}$ let $X \subseteq V$ denote the nodes whose randomness is already fixed and let ϕ_X be their randomness (we formally define ϕ_X later). The crucial invariant that we maintain for each cluster C at all times is the following

$$\textbf{Invariant:} \quad E\left[\sum_{v \in V} X_v \mid \phi_X\right] < 1. \tag{1}$$

The invariant will be made formal in the rest of the proof. Initially (when $X = \emptyset$), it holds with the linearity of expectation and as the error probability of the algorithm implies that $E[X_v] = P(X_v = 1) < 1/n$ holds.

Algorithm $\mathcal{B}$

Compute a weak network decomposition with cluster distance strictly more than $4(T(N)+r)$ with $O(\log N)$ color classes, $\beta = T(N) \cdot \operatorname{poly}\log N$ Steiner tree cluster radius and congestion $\kappa = O(\log N)$. Iterate through the color classes of the decomposition and consider each cluster separately. We describe the process for one cluster $\mathcal{C}$ when processing the i-th color class of the network decomposition. We define three layers according to the distance to vertices in $\mathcal{C}$. Denote the vertices in $\mathcal{C}$ as $W^0_{\mathcal{C}}$, the vertices in $V \setminus W^0_{\mathcal{C}}$ in distance at most $r + T(N)$ from $\mathcal{C}$ as $W^1_{\mathcal{C}}$ and the vertices in $V \setminus (W^0_{\mathcal{C}} \cup W^1_{\mathcal{C}})$ with distance at most $2(r + T(N))$ from $\mathcal{C}$ as $W^2_{\mathcal{C}}$. Define $W_{\mathcal{C}} = W^0_{\mathcal{C}} \cup W^1_{\mathcal{C}} \cup W^2_{\mathcal{C}}$. Note that we have $W_{\mathcal{C}} \cap W_{\mathcal{C}'} = \emptyset$ for two distinct clusters $\mathcal{C}, \mathcal{C}'$ in the same color class of the decomposition due to the cluster distance. Further, even if Lemma 13 is applied to a subgraph H (with size at most N) of a communication network G we have $|W_{\mathcal{C}}| \leq N$, as $W_{\mathcal{C}}$ only contains nodes of H.

Dealing with one cluster $\mathcal{C}$: Extend the Steiner tree of $\mathcal{C}$ to $W_{\mathcal{C}}$ using a BFS, ties broken arbitrarily. Notice that no two BFS trees interfere with each other since we handle different color classes separately and two clusters of the same color are in large enough distance. Use the Steiner tree to assign new IDs (unique only within the nodes in $W_{\mathcal{C}}$)) from range $[N]$ to all nodes in $W_{\mathcal{C}}$. Different clusters use the same set of IDs and each ID can be represented with $O(\log N)$ bits. Each node of $W_{\mathcal{C}}$ learns about the new ID of each of its (at most) Δ neighbors in $W_{\mathcal{C}}$ in one round and uses $O(\Delta \cdot \log N) = O(\log N)$ bits to store its adjacent edges. Using Lemma 12, the cluster leader $\ell_{\mathcal{C}}$ learns the whole topology, inputs and already determined random bits of $G[W_{\mathcal{C}}]$ in $O(\kappa \cdot (\beta + (N \cdot \operatorname{poly}\log N)/b))$ rounds. In this step, the input also includes an acyclic orientation of the edges between vertices of $W_{\mathcal{C}}$, where an edge is oriented from $u \in W_{\mathcal{C}}$ to $v \in W_{\mathcal{C}}$ if the original ID of v is larger than the original ID of v.

For a node v let R_v describe its randomness. When we process cluster $\mathcal{C}$ we determine values $\{r_v \mid v \in \mathcal{C}\}$ for the random variables $\{R_v \mid v \in \mathcal{C}\}$. Let X be a set of vertices v for which we have already determined values r_v. Then we denote $\phi_X = \bigwedge_{v \in X}(R_v = r_v)$. The proof of the following claim is deferred to Appendix A.

▷ Claim 14. Assume that Invariant (1) holds for $X = \bigcup_{\mathcal{C} \text{ has color } < i} \mathcal{C}$ before processing the clusters of color i. Then the cluster leaders $\ell_{\mathcal{C}}$ of all clusters $\mathcal{C}$ of color i, can in parallel, find values $\{r_v \mid v \in \mathcal{C}\}$ such that Invariant (1) holds afterwards for $X' = \bigcup_{\mathcal{C} \text{ has color } \leq i} \mathcal{C}$.

Once the cluster leader $\ell_{\mathcal{C}}$ has fixed the randomness for the vertices in $\mathcal{C}$, it disseminates them to all vertices of its cluster via Lemma 12 and we continue with the next color class of the network decomposition. At the end of the algorithm each node knows the values of its random bits and one can execute $T(N)$ with it, as the initial algorithm $\mathcal{A}$ also works with bandwidth b. Due to the Invariant (1) we obtain that the distributed random bits are such that $E[\sum_{v \in V} X_v \mid \phi_V] < 1$ at the end of the algorithm, but as all X_v's are in $\{0, 1\}$ and there is no randomness involved (each vertex is processed in at least one cluster and thus we fix $R_v = r_v$ for each $v \in V$) we obtain that $X_v = 0$ for all $v \in V$, that is, the algorithm does not fail at any vertex.

Runtime: Computing the network decomposition takes $T(N) \cdot \operatorname{poly}\log N$ rounds via Theorem 6. The runtime for processing one color class of the network decomposition is bounded as follows. Collecting the topology and all other information takes $T(N) \cdot \operatorname{poly}\log N$ rounds due to Lemma 12, fixing randomness locally does not require communication. Disseminating the random bits to the vertices of each cluster takes $\operatorname{poly}\log N$ rounds, again due to Lemma 12. As there are only $\log N$ color classes the runtime for computing good random bits for all nodes is $T(N) \cdot \operatorname{poly}\log N$. Executing algorithm $\mathcal{A}$ with these random bits takes $T(N)$ rounds. ◀

▶ **Lemma 15.** *There is a deterministic LLL algorithm for range bounded LLL instances with constant dependency degree d and LLL criterion $epd^2 < 1$ that runs in $O(\operatorname{poly}\log\log n)$ rounds on any graph with at most n nodes if the communication bandwidth is $b = \Theta(n)$.*

Proof. Plug the randomized algorithm of Lemma 11 into the derandomization result of Lemma 13. We can apply the lemma because the feasibility of a computed solution can be checked in 1 round in CONGEST. ◀

Proof of Theorem 1. The goal is to apply the shattering framework of [24]. The pre-shattering phase takes $\operatorname{poly}\Delta + O(\log^* n)$ rounds and afterwards all events with an unset variable induce small components of size $N = \operatorname{poly}(\Delta, \log n)$. Furthermore, each of the small components also forms an LLL instance, but with a slightly worse LLL criterion. Then we apply Lemma 15 to solve all small components in parallel. Next, we describe these steps in detail. The pre-shattering phase begins with computing a distance-2 coloring of the dependency graph H with $O(\Delta^2)$ colors, i.e., a coloring in which each color appears only once in each inclusive neighborhood. In the LOCAL model this takes $O(\log^* n)$ rounds with Linial's algorithm [37]. In the CONGEST model we can compute such a coloring in $O(\operatorname{poly}\Delta_H \cdot \log^* n) = O(\log^* n)$ rounds.

Next, we iterate through the $O(\Delta^2)$ color classes to set some of the variables. The unset variables either have the status *frozen* or *non-frozen*. At the beginning all variables are unset and non-frozen. We iterate through the $O(\Delta_H^2)$ color classes and process all vertices (events) of the same color class in parallel. Each non-frozen variable of a processed event $\mathcal{E}$ is sampled; then the node checks how the conditional probabilities of neighboring events have changed. As the LLL is range bounded this step can be implemented in $O(1)$ rounds. If there is an event $\mathcal{E}'$ (possibly $= \mathcal{E}$) whose probability has increased to at least $p' = \sqrt{p}$, its variables are unset and all variables of event $\mathcal{E}'$ are frozen. The next three observations are proven in [24] and capture the properties of this pre-shattering phase.

▶ **Observation 16.** *For each event $\mathcal{E} \in \mathcal{X}$, the probability of $\mathcal{E}$ having at least one unset variable is at most $(d+1)\sqrt{p}$. Furthermore, this is independent of events that are further than 2 hops from $\mathcal{E}$.*

The following result follows with Observation 16 and the by now standard shattering lemma. We do not discuss the details as it has been done in [24].

▶ **Observation 17** (Small components)**.** *The connected components in H induced by all events with at least one unset variable are w.h.p. in n of size $N = \operatorname{poly}(\Delta_H) \cdot \log n = O(\log n)$.*

The following observation holds as each event with an unset variable fails at most with probability $p' = \sqrt{p}$ when its variables are frozen. It is proven in [24].

▶ **Observation 18.** *The problem on each connected component induced by events with at least one unset variable is an LLL problem with criterion $p'(d+1) < 1$.*

To complete our proof we apply the deterministic algorithm of Lemma 15 on each component in parallel. Let U be the set of nodes that have its random bits not yet determined. Note that any vertex in U is part of one component. For each $u \in U$ the values of already determined random bits in the r-hop ball around u are included in u's input – many nodes already determine their random bits in the pre-shattering phase. Even conditioned on the random bits determined in the pre-shattering phase (formally this provides a promise to the inputs), each instance is a range bounded LLL on a bounded degree graph with at most N nodes and the standard CONGEST bandwidth is $b = \Omega(\log n) = \Omega(N)$, the runtime is $\operatorname{poly}\log N = \operatorname{poly}\log\log n$. All instances can be dealt with independently since by definition, the connected unsolved components share no events nor variables. ◀

The celebrated result by [19] says that an LCL problem either cannot be solved faster than in $\Omega(\log n)$ rounds or can be solved with an LLL algorithm. In our work, we show that their proof can be easily extended to work with range bounded LLLs.

▶ **Lemma 19** ([19], Proof deferred to the full version)**.** *Any LCL problem that can be solved with a randomized $o(\log n)$-round* LOCAL *algorithm with error probability $< 1/n$ on any graph with at most n nodes can be solved via the following procedure: Create a bounded range LLL instance with LLL criterion $p(ed)^{100} < 1$, solve the instance, and run a constant time deterministic* LOCAL *algorithm that uses the solution of the LLL instance as input.*

Lemma 19 of [19] has been used in several other works, e.g., in [3, 13]. Note that [3] sets up the same LLL with the purpose of designing an efficient CONGEST algorithm for it; however, [3] is restricted to the setting where the input graph is a tree which allows for very different methods of solving the respective LLL and admits even an $O(\log\log n)$-round algorithm. We combine Lemma 19 with our LLL algorithm in Theorem 1 to prove Corollary 2.

▶ **Corollary 2.** *There is no LCL problem with randomized complexity strictly between $\operatorname{poly}\log\log n$ and $\Omega(\log n)$ in the* CONGEST *model.*

The results in this section imply randomized $\operatorname{poly}\log\log n$-round algorithms for classic problems such as Δ-coloring on constant degree graphs (as Δ-coloring is an LCL which has an $o(\log n)$-round LOCAL algorithm [29] the result follows along the same lines as Corollary 2) and various defective coloring variants for constant degree graphs by modeling them as range bounded LLLs and applying Theorem 1, see [20] for various such problems and how they can be modeled as LLLs.

Our network decomposition algorithm with few colors in combination with the LOCAL model LLL algorithm from [24] provides the following theorem. Due to the unconstrained message size in the LOCAL model, it does not require the LLL instances to be range bounded.

▶ **Theorem 20.** *Let $\lambda \in \mathbb{N}$ be constant. There exists a deterministic $n^{2/\lambda} \operatorname{poly}\log n$ round* LOCAL *algorithm for LLL instances with criterion $p(ed)^{\lambda} < 1$.*

References

1 Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In *Proc. 30th Symp. on Found. of Computer Science (FOCS)*, pages 364–369, 1989.

2 Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. Almost global problems in the LOCAL model. In *32nd International Symposium on Distributed Computing, DISC*, pages 9:1–9:16, 2018. doi:10.4230/LIPIcs.DISC.2018.9.

3 Alkida Balliu, Keren Censor-Hillel, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Locally checkable labelings with small messages. In *International Symposium on Distributed Computing DISC*, 2021.

4 Alkida Balliu, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. New classes of distributed time complexity. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1307–1318, 2018. doi:10.1145/3188745.3188860.

5 Philipp Bamberger, Fabian Kuhn, and Yannic Maus. Efficient deterministic distributed coloring with small bandwidth. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 243–252, 2020. doi:10.1145/3382734.3404504.

6 Leonid Barenboim. On the locality of some np-complete problems. In *Automata, Languages, and Programming - 39th International Colloquium (ICALP)*, pages 403–415, 2012. doi:10.1007/978-3-642-31585-5_37.

7 Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Morgan & Claypool Publishers, 2013.

8 Leonid Barenboim, Michael Elkin, and Cyril Gavoille. A fast network-decomposition algorithm and its applications to constant-time distributed computation. *Theor. Comput. Sci.*, 751:2–23, 2018. doi:10.1016/j.tcs.2016.07.005.

9 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.

10 József Beck. An algorithmic approach to the Lovász local lemma. *Random Structures & Algorithms*, 2(4):343–365, 1991.

11 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 *ACM Symposium on Theory of Computing (STOC)*, 2016.

12 Sebastian Brandt, Christoph Grunau, and Václav Rozhon. Generalizing the sharp threshold phenomenon for the distributed complexity of the lovász local lemma. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 329–338, 2020. doi:10.1145/3382734.3405730.

13 Sebastian Brandt, Christoph Grunau, and Václav Rozhon. The randomized local computation complexity of the lovász local lemma. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 307–317. ACM, 2021. doi:10.1145/3465084.3467931.

14 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 (PODC)*, pages 389–398, 2019. doi:10.1145/3293611.3331636.

15 Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. Derandomizing local distributed algorithms under bandwidth restrictions. *Distributed Comput.*, 33(3-4):349–366, 2020. doi:10.1007/s00446-020-00376-1.

16 Yi-Jun Chang and Mohsen Ghaffari. Strong-diameter network decomposition. *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, abs/2102.09820, 2021. arXiv:2102.09820.

17 Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. The complexity of distributed edge coloring with small palettes. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2633–2652, 2018. doi:10.1137/1.9781611975031.168.

18 Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model. In *the Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 615–624, 2016.

19 Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the LOCAL model. *SIAM J. Comput.*, 48(1):33–69, 2019. doi:10.1137/17M1157957.

20 Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed Algorithms for the Lovász Local Lemma and Graph Coloring. *Distributed Computing*, 30(4):261–280, 2017. doi:10.1007/s00446-016-0287-6.

21 Janosch Deurer, Fabian Kuhn, and Yannic Maus. Deterministic distributed dominating set approximation in the CONGEST model. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 94–103, 2019. doi:10.1145/3293611.3331626.

22 Michael Elkin and Ofer Neiman. Distributed strong diameter network decomposition. In *Proc. 35th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 211–216, 2016.

23 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.

24 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. doi:10.4230/LIPIcs.DISC.2017.18.

25 Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proc. 27th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 270–277, 2016.

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

27 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. doi:10.1109/FOCS.2018.00069.

28 Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. On Derandomizing Local Distributed Algorithms. In *the Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 662–673, 2018.

29 Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, and Yannic Maus. Improved distributed δ-coloring. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 427–436, 2018. URL: https://dl.acm.org/citation.cfm?id=3212764.

30 Mohsen Ghaffari and Fabian Kuhn. Derandomizing distributed algorithms with small messages: Spanners and dominating set. In *32nd International Symposium on Distributed Computing (DISC)*, pages 29:1–29:17, 2018. doi:10.4230/LIPIcs.DISC.2018.29.

31 Mohsen Ghaffari and Fabian Kuhn. On the use of randomness in local distributed graph algorithms. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 290–299, 2019. doi:10.1145/3293611.3331610.

32 Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *ACM Symposium on Theory of Computing (STOC)*, pages 784–797, 2017. doi:10.1145/3055399.3055471.

33 Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *ACM Symposium on Theory of Computing (STOC)*, pages 784–797. ACM, 2017.

34 Mohsen Ghaffari and Julian Portmann. Improved network decompositions using small messages with applications on mis, neighborhood covers, and beyond. In *33rd International Symposium on Distributed Computing (DISC)*, pages 18:1–18:16, 2019. doi:10.4230/LIPIcs.DISC.2019.18.

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

36 Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Alexandre Nolin. Coloring fast without learning your neighbors' colors. In *34th International Symposium on Distributed Computing (DISC)*, pages 39:1–39:17, 2020. doi:10.4230/LIPIcs.DISC.2020.39.

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

38 Nati Linial and Michael Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993.

39 Robin A. Moser and Gábor Tardos. A Constructive Proof of the General Lovász Local Lemma. *J. ACM*, pages 11:1–11:15, 2010.

40 M. Naor and L. Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

41 Alessandro Panconesi and Aravind Srinivasan. On the complexity of distributed network decomposition. *Journal of Algorithms*, 20(2):581–592, 1995.

42 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.

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

A Proof of Claim 14

▷ Claim 14. Assume that Invariant (1) holds for $X = \bigcup_{\mathcal{C} \text{ has color } < i} \mathcal{C}$ before processing the clusters of color i. Then the cluster leaders $\ell_{\mathcal{C}}$ of all clusters $\mathcal{C}$ of color i, can in parallel, find values $\{r_v \mid v \in \mathcal{C}\}$ such that Invariant (1) holds afterwards for $X' = \bigcup_{\mathcal{C} \text{ has color } \leq i} \mathcal{C}$.

If we considered just a single cluster $\mathcal{C}$, independently from all other clusters with color i, then there are choices for the values of r_v satisfying the claim due to the law of total probability, e.g., as used in the method of conditional expectation, and as $E\big[\sum_{v \in V} X_v \mid \phi_X\big] < 1$ holds before we fix any randomness of vertices inside $\mathcal{C}$.

Proof. The only information that cluster leaders need to perform the necessary calculations is information on the topology of $G[W_{\mathcal{C}}]$, inputs to nodes in $W_{\mathcal{C}}$, the relative order of adjacent IDs of nodes in $G[W_{\mathcal{C}}]$ and already determined randomness of nodes in $X \cap W_{\mathcal{C}}$.

We show that the randomness of all vertices in clusters with color i can be fixed with this knowledge. To this end the cluster leader $r_{\mathcal{C}}$ fixes the randomness $\phi_{\mathcal{C}}$ such that

$$E\Big[\sum_{v \in W_{\mathcal{C}}} X_v \mid \phi_{\mathcal{C}} \wedge \phi_{X \cap W_{\mathcal{C}}}\Big] \leq E\Big[\sum_{v \in W_{\mathcal{C}}} X_v \mid \phi_{X \cap W_{\mathcal{C}}}\Big]$$

holds. Such a choice for $\phi_{\mathcal{C}}$ exists, again due to the law of total probability, and $\ell_{\mathcal{C}}$ has full information to compute such values as all values in the inequality only depend on information that vertices in $W_{\mathcal{C}}$ sent to $\ell_{\mathcal{C}}$. In particular, the randomness of nodes in $\mathcal{C}$ only influences the random variables X_v of nodes in $W^0_{\mathcal{C}} \cup W^1_{\mathcal{C}}$ and computing this influence only requires knowledge from $W^0_{\mathcal{C}} \cup W^1_{\mathcal{C}} \cup W^2_{\mathcal{C}}$.

After all cluster leaders of clusters with color i fix the randomness of the vertices in their clusters according to the above method the invariant is still satisfied since the randomness R_v for $v \in \mathcal{C}$ does not influence the random variable X_u for $u \in V \setminus W_{\mathcal{C}}$. More formally, let X be the set of vertices with fixed randomness before we process the i-th color class of clusters and let Y be the set of vertices whose randomness we fix when processing the i-th color class. Let $X' = X \cup Y$. Let $W = \bigcup_{\mathcal{C} \text{ has color } i} W_{\mathcal{C}}$. Recall that $W_{\mathcal{C}} \cap W_{\mathcal{C}'} = \emptyset$ for $\mathcal{C} \neq \mathcal{C}'$. Due to the linearity of expectation and the aforementioned reasons we obtain

$$
\begin{aligned}
E\left[\sum_{v\in V} X_v \mid \phi_{X'}\right] &= E\left[\sum_{v\in V\setminus W} X_v \mid \phi_{X'}\right] + \sum_{\mathcal{C} \text{ has color } i} E\left[\sum_{v\in W_{\mathcal{C}}} X_v \mid \phi_{X'\cap W_{\mathcal{C}}}\right] \\
&\leq E\left[\sum_{v\in V\setminus W} X_v \mid \phi_{X}\right] + \sum_{\mathcal{C} \text{ has color } i} E\left[\sum_{v\in W_{\mathcal{C}}} X_v \mid \phi_{X\cap W_{\mathcal{C}}}\right] \\
&= E\left[\sum_{v\in V} X_v \mid \phi_{X}\right] < 1.
\end{aligned}
$$

All clusters can be processed in parallel as their distance is strictly more than $4(T(N)+r)$ and the choices in cluster $\mathcal{C}$ do not change the probability for $X_v = 1$ for $v \in \mathcal{C}' \neq \mathcal{C}$. ◁