Fast Distributed Vertex Splitting with Applications

Magnús M. Halldórsson
Reykjavik University, Iceland

Yannic Maus
TU Graz, Austria

Alexandre Nolin
Reykjavik University, Iceland[1]
CISPA, Saarbrücken, Germany

Abstract

We present poly log log n-round randomized distributed algorithms to compute vertex splittings, a partition of the vertices of a graph into k parts such that a node of degree $d(u)$ has $\approx d(u)/k$ neighbors in each part. Our techniques can be seen as the first progress towards general poly log log n-round algorithms for the Lovász Local Lemma.

As the main application of our result, we obtain a randomized poly log log n-round CONGEST algorithm for $(1+\varepsilon)\Delta$-edge coloring n-node graphs of sufficiently large constant maximum degree Δ, for any $\varepsilon > 0$. Further, our results improve the computation of defective colorings and certain tight list coloring problems. All the results improve the state-of-the-art round complexity exponentially, even in the LOCAL model.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Graph problems, Edge coloring, List coloring, Lovász local lemma, LOCAL model, CONGEST model, Distributed computing

Digital Object Identifier 10.4230/LIPIcs.DISC.2022.26

Related Version *Full Version*: https://arxiv.org/abs/2208.08119

1 Introduction

Consider the following fundamental load-balancing problem: Partition the vertices of an n-node degree-Δ graph into two parts so that each node has at most $(1+\varepsilon)\Delta/2$ neighbors in each part, where $\varepsilon > 0$ is an arbitrary given constant. When Δ is large enough (say, superlogarithmic), such a *2-splitting* is trivially achieved w.h.p. without communication. Can it be solved fast distributively for arbitrary Δ?

The 2-splitting problem can be formulated as an instance of the *Lovász local lemma* (LLL). Consider some *"bad"* events over a probability space. The celebrated Lovász local lemma states that if the events satisfy certain limited dependencies, then there is a positive probability that none of them happens [16]. In the 2-splitting problem, the probability space is spanned by each node picking a part uniformly at random and there is a bad event for each node that occurs when too many of its neighbors are in any one of the parts. In the constructive version of the LLL, the objective is to also compute an assignment avoiding all bad events, and using known distributed LLL algorithms, it can be solved in $O(\log n)$ distributed rounds [41, 13]. For small Δ, there is a faster $O(\Delta^2 + \text{poly}\log\log n)$-round algorithm [18], but it does not improve the case of arbitrary Δ. This leaves a major open problem: Can we close the gap between the $O(\log n)$ upper bound and $\Omega(\log_\Delta \log n)$ lower bound [2]? Clarifying this for 2-splitting would be the first step towards resolving the complexity of general distributed LLLs.

[1] Alexandre Nolin changed affiliation from Reykjavik University to CISPA after this work was completed.

Our central technical result is to answer this question affirmatively, even in the bandwidth-constrained CONGEST model, by giving a $\operatorname{poly}\log\log n$-round algorithms for 2-splitting and various other vertex- and edge splitting problems. Note that these are also exponential improvements for the LOCAL model. Such splitting problems are pervasive in distributed graph algorithmics [27, 28, 17, 19, 25, 23, 31, 2]. They can be viewed as questions of rounding and discrepancy, and they are frequently the major building block in solving various classic problems when using a divide-and-conquer approach.

We illustrate the reach of the techniques by giving much faster algorithms for two classic coloring problems.

▶ **Theorem 1** (Edge coloring). *For any constant $\varepsilon > 0$, there is a $\operatorname{poly}\log\log n$-round randomized* CONGEST *algorithm to compute a $(1+\varepsilon)\Delta$-edge coloring on any graph with maximum degree $\Delta \geq \Delta_0$ where Δ_0 is a sufficiently large constant.*

Notice that as a function of n alone, previous methods use at least $\Omega(\log n)$ time, even in the LOCAL model. The problem has a $\Omega(\log_\Delta \log n)$ lower bound [10]. Previously, $\operatorname{poly}\log\log n$-round algorithms were only known for $2\Delta - 1$-edge coloring [15, 29], and $O(\log_\Delta n + \operatorname{poly}\log\log n)$-round algorithms for using any smaller number of colors [14, 42, 13, 15, 10], even in the LOCAL model. Tackling this problem in CONGEST is non-trivial as it depends on LLL, which only has efficient known CONGEST solutions for the constant-degree case [39].

In the second application, the (L, T)-*list coloring* problem, each node of a graph is given a list of at least L colors such that any color in its list appears in at most T neighbors' lists. We ask for a valid node coloring where each node receives a color from its list, with the ratio L/T as small as possible. Observe that the degree of a node can be much larger than its list of colors, and thus greedy approaches are insufficient, even centrally.

▶ **Theorem 2** (List coloring). *There is a $\operatorname{poly}\log\log n$-round randomized* LOCAL *algorithm for the list coloring problem, for any T and L with $L \geq (1+\delta)T$, for any $\delta > 0$ and any $\Delta \geq \Delta_0$, for some absolute constant Δ_0.*

Previous algorithms either used $O(\log n)$-rounds [13] or required $L/T \geq C_0$ for a (large) constant C_0 [18]. See Appendix B for more related work on list coloring.

1.1 Contributions on Splitting problems

The main ingredient for both of the above applications is our efficient method to split graphs into small degree subgraphs. A *k-vertex splitting problem* with *discrepancy z* is a partition of the vertex set into k parts $V_1, \ldots, V_k$ such that, for each $i \in [k]$, each node $v \in V$ has $d(v)/k \pm z$ neighbors in V_i. Intuitively, splitting a graph into k parts with a discrepancy of $\varepsilon\Delta/k$ is useful to solve various problems that are easier on low-degree graphs. These problems must be resilient to imperfect splits, which is ensured in coloring problems by having a surplus of colors.

A Chernoff bound argument shows that such splittings are quite easy for high degrees ($\Delta \gg k \log n$). We obtain the following theorem:

▶ **Theorem 3.** *There exists a universal constant $c_1 > 0$ s.t.: For any $\varepsilon > 0$, maximum degree $\Delta \leq \operatorname{poly}\log n$, and $k \leq c_1 \cdot (\varepsilon^4 \Delta / \ln \Delta)$, there is a distributed randomized* LOCAL *algorithm to compute a k-vertex splitting with discrepancy $\varepsilon\Delta/k$ in $O(1/\varepsilon) + \operatorname{poly}\log\log n$ rounds.*

The $\operatorname{poly}\log\log n$ term in the runtime of Theorem 3 stems from solving LLL instances of size $N = \operatorname{poly}\log n$ deterministically. Any improvement on such algorithms immediately carries over to our result. However, there is a lower bound of $\Omega(\log_\Delta \log n)$ rounds for randomized and $\Omega(\log_\Delta n)$ rounds for deterministic algorithms for the respective splitting problems (and hence also for the LLL problem) [2]. These lower bounds even hold for a weak variant of the vertex splitting problem, in which each node only needs to have one neighbor of each color class.

Variants. Our main applications require subtle variations of the splitting problem. To this end, we solve a more general problem, where we separate the two functions of each node: as a variable (which part is it assigned to) and as an event (whether its neighborhood is evenly split). In the *bipartite k-vertex splitting problem with discrepancy z*, we have a set V^{L} of nodes for the events and a set V^{R} of nodes for the variables, with an edge between every dependent variable-edge pair. We wish to partition V^{R} into k parts such that each event vertex $u \in V^{\mathsf{L}}$ has $d(u)/k \pm z$ neighbors in each part.

▶ **Theorem 4.** *There exists a universal constant $c_2 > 0$ s.t.: For any $\varepsilon > 0$, maximum degree $\Delta \leq \operatorname{poly}\log n$ and $k \leq c_2 \cdot (\varepsilon^4 \Delta_{\mathsf{L}} / \ln \Delta)$, there is a distributed randomized* LOCAL *algorithm to compute a bipartite k-vertex splitting problem with discrepancy $\varepsilon\Delta_{\mathsf{L}}/k$ in $O(1/\varepsilon) + \operatorname{poly}\log\log n$ rounds.*

We also devise CONGEST versions of Theorems 3 and 4 that are essential to our edge coloring result in CONGEST. The formal statement appears in Theorem 20 and requires k to be a $O(\log^2 \log n)$ factor smaller than in Theorems 3 and 4.

In a *d-defective c-coloring*, each of the c color classes induces a graph of maximum degree d. Defective colorings are frequently used in divide-and-conquer approaches to other coloring problems [34, 3] and they have been studied in several works, e.g., [34, 3, 33, 26, 37], usually stating variations of deterministic algorithms for computing d-defective coloring with $O((\Delta/d)^2)$ colors. As any vertex splitting is also a defective coloring, Theorem 3 implies a $\operatorname{poly}\log\log n$-round algorithm for $(1+\varepsilon)\Delta/k$-defective k-coloring. Previous algorithms for $(1+\varepsilon)\Delta/k$ defective colorings either used $O(k^2)$ colors [34, 37] or a logarithmic number of rounds through solving the respective LLL problem [13].

1.2 Challenges to Fast and Efficient Splitting

Known approaches to splitting (or any of the other problems we consider) all build on the *Lovász Local Lemma* (LLL) for the low-degree case ($\Delta < \log n$). This hits a wall, since there are no strongly sublogarithmic time distributed LLL algorithms known, in spite of intensive efforts [12, 10].

There are two known approaches to distributed LLL algorithms. The breakthrough Moser-Tardos method [41, 13] is based on stochastic local search, which appears to inherently require logarithmic rounds. The other approach is to use the *shattering* technique, solving most of the problem quickly, leading to small remaining "shattered" subgraphs for which we can afford to apply slower techniques. This was introduced by Beck [5] in the centralized setting and Alon [1] in the parallel setting.

Fischer and Ghaffari [18] proposed a shattering-based distributed algorithm, modeled on an earlier sequential algorithm of Molloy and Reed [40]. Using recent network decompositions [46], their method runs in $O(\Delta^2 + \operatorname{poly}\log\log n)$ time, which is fast for low-degree graphs ($\Delta \leq \operatorname{poly}\log\log n$) but doesn't improve the general case. To understand the issue, let us examine more closely the reasoning behind the method of [18] in the context of 2-splitting.

A random assignment (of the nodes into the parts) is easily seen to satisfy the lion's share of the vertices, where "satisfied" means having discrepancy within the stated bound. Each node is so likely to be satisfied that the remaining subgraph is indeed *shattered*: the connected components induced by the set of unsatisfied nodes are of small size (assuming $\Delta \leq \text{poly} \log n$, which is the hard case). One natural approach is to *undo* the assignment to the unsatisfied nodes, and then solve the problem separately on the unsatisfied nodes. However, this causes new problems: Nodes that were previously satisfied may become hard to satisfy. For instance, suppose a node v has neighbors $u_1, \ldots, u_t$ assigned to the first part and nodes $u_{t+1}, \ldots, u_{2t}$ assigned to the second part, for a perfect split. But it is now possible that all of $u_1, u_2, \ldots, u_t$ are retracted, having their assignment undone. Then, satisfying v now requires assigning its neighbors back to the second part, which leaves little flexibility, and there may not be a valid solution.

Fischer and Ghaffari [18] (and [40]) fix this by sampling the random variables (i.e., which part each node is assigned to) only *gradually*, i.e., at most one variable per event is sampled simultaneously. Along with "freezing" (or deferring) certain nodes, this ensures that no vertex experiences too heavy a setback caused by retractions. The gradual sampling is achieved by first computing a distance-2 coloring of the graph using $O(\Delta^2)$ colors, and then sampling only the nodes of a single color class at a time. The downside is that this unavoidably requires time complexity at least Δ^2.

A different type of challenge appears when aiming for bandwidth-efficient algorithms. Even if one drastically improves upon the sketched $O(\Delta^2)$ "pre-shattering" procedure from [18], the deterministic procedure used in the "post-shattering" phase of their algorithm to complete the obtained partial solution makes heavy use of the unlimited bandwidth of the LOCAL model. In fact, while both types of randomized distributed LLL methodologies [38, 13] are themselves frugal in terms of bandwidth, known deterministic LLL algorithms are based on bandwidth-hungry generic derandomization results [27, 23, 46].

1.3 Our Methods in a Nutshell

Fast splitting. Our approach is to sample gradually – like in [40, 18] – but faster. We group the variables (representing the part assigned to a node) into *buckets* and then sample the variables one bucket at a time. This is crucially done so that the impact of any given bucket on any given event is limited (namely, the number of neighbors of a node in any given bucket is upper bounded), so that we can recover from bad probabilistic assignments. Intuitively, a node might have to "give up" on all of its neighbors inside a bucket, i.e., it may be that their assignment is chosen adversarially. As we can guarantee that each event has to give up on at most one bucket, it turns out to suffice to use a constant number of buckets to get a good split, or more generally $O(1/\varepsilon)$ buckets to get $(1+\epsilon)$-approximate split. Generating this bucket assignment is itself a splitting problem (that we term a *q-divide*) requiring the use of LLL, but one with less moving parts and a much simpler analysis in LOCAL. In the CONGEST model, it still requires a novel post-shattering phase.

Post-shattering in CONGEST. We solve the post-shattering phase as a sequence of successive relaxations, one for each disjoint group of clusters of the network decomposition. In effect, we solve a new LLL for each cluster group, with progressively stricter criteria (due to the accumulated discrepancy). Each relaxation is solved by a randomized, rather than a deterministic, algorithm. Namely, we run $O(\log n)$ independent instances of the Moser-Tardos process on the cluster, and since each succeeds with constant probability, we achieve at least one valid solution, w.h.p. This parallel instance technique was introduced by Ghaffari [21] for problems like $\Delta + 1$-coloring, a simpler setting where the problem is always solvable on clusters processed *later*, regardless of how the *earlier* clusters are solved.

In our edge-coloring application, we use splitting to whittle down the degree parameter to a manageable size. Once degrees are down to $\operatorname{poly}\log\log n$, we can simulate the known algorithms from the LOCAL model, including derandomization techniques, to solve them also in $\operatorname{poly}\log\log n$ CONGEST rounds.

For our list coloring results, we use our splitting procedure to first reduce the parameter L and T to $\operatorname{poly}\log\log n$ while keeping the initial ratio $L/T \geq (1+\delta)$ almost intact. Then, in additional color pruning steps we amplify the ratio until it is larger than a sufficiently large constant, at which point the problem can be solved efficiently via a known LLL-based method [18].

1.4 Further Related Work

The only known LLL algorithm in the CONGEST model is by Maus and Uitto [39] who provide a $\operatorname{poly}\log\log n$-round algorithm for LLLs with a polynomial criterion and constant dependency degree. In this work, we observe that their runtime remains $\operatorname{poly}\log\log n$, even if the dependency degree is as large as $\operatorname{poly}\log\log n$, see Lemma 21 for details.

Edge coloring. Dubhashi, Grable, and Panconesi [14] gave a distributed algorithm for $(1+\epsilon)\Delta$-edge coloring based on the Rödl nibble method. Their results only apply to large values of Δ. Elkin, Pettie, and Su [15] extended the reach to arbitrary Δ by reduction to distributed LLL, and obtained improved complexity of $O(\log^* \Delta \cdot \lceil \log n/\Delta^{1-o(1)} \rceil)$. Chang et al. [10] improved the complexity to $O(\log_\Delta n + \log^{3+o(1)} \log n)$ for $\varepsilon^{-1} \in O(1)$, and to $O(\log n)$ for $\varepsilon^{-1} \in \tilde{O}(\sqrt{\Delta})$.

There are clear tradeoffs between the number of colors and the time complexity. Computing $(2\Delta-1)$-edge coloring can be achieved in $\operatorname{poly}\log\log n$ rounds [15] (even in CONGEST [29]), and even in $O(\log^* n)$ rounds for $\Delta \geq \log^2 n$ [15, 29]). Chang et al. [10] showed via the round elimination method that computing a $(2\Delta-2)$-edge coloring requires $\Omega(\log_\Delta \log n)$ rounds. A $\operatorname{poly}(\Delta, \log n)$-round algorithm is known for $\Delta+2$-coloring [47] and very recently for $\Delta+1$-coloring [6]. Chang et al. [10] showed that an (possibly randomized) algorithm for $\Delta+1$-coloring that can start with any partial coloring requires $\Omega(\Delta \log n)$ rounds. They also showed that $(1+\log\Delta/\sqrt{\log\Delta})\Delta$-edge coloring can be found in $O(\log n)$ rounds.

Splitting. Ghaffari and Su [28] gave three LOCAL algorithms for splitting the edges of a graph into two parts such that each node has at most $(1+\varepsilon)\Delta/2$ incident edges in each part, rounded up for their randomized result.

Their deterministic algorithms achieve complexity $O(\varepsilon^{-1}\Delta^2 \log^5 n)$ when $\Delta \geq c \cdot \varepsilon^{-1} \log n$, and complexity $O(\varepsilon^{-3} \log^7 n)$ when $\Delta \geq c \cdot \varepsilon^{-2} \log n$, where c is a suitable absolute constant. Their randomized algorithm solves the problem for all Δ in $O(\varepsilon^{-2}\Delta^2 \log^4 n)$ rounds. These results were later improved by [25] to $O(\varepsilon^{-1-o(1)} \log n)$ rounds for deterministic algorithms and $O(\varepsilon^{-1-o(1)} \log\log n)$ for randomized algorithms, with stronger guarantees on the split. However, it is unclear whether these edge-splitting algorithms can be extended to the CONGEST model, as the algorithms communicate simultaneously over various long paths in the network. The importance of splitting problems for the area was highlighted in [27] and [2]. The latter gave various direct reductions of the maximal independent set problem and coloring problems to splitting problems. In addition, they studied several weak variants of the splitting problem, e.g., splitting into two parts such that each node needs to have at least one neighbors in each part. They show that even these have a $\Omega(\log_\Delta \log n)$ lower bound for randomized algorithms and $\Omega(\log_\Delta n)$ for deterministic algorithms. They also obtain a $\operatorname{poly}\log\log n$-round algorithm for the weak variant in the special case of regular graphs.

1.5 Outline

In Section 2, we define the models, the setup for the Lovász Local Lemma and introduce notation. Section 3 contains our algorithm for the q-divide that does not just serve as a warm-up for our more involved splitting algorithms, but is also used as a subroutine in the latter. Our main splitting algorithm is presented in Section 4 for LOCAL and in Section 5 for CONGEST. In Section 6, we present our splitting applications for edge coloring in LOCAL, and in Appendix A for CONGEST. In Appendix B, we provide more details on our second application, i.e., list coloring. The appendix also contains our results on bipartite splitting (Theorem 4). Further details, and the missing full proofs appear in the full version [30].

2 Models, Lovász Local Lemma, Shattering, and Notation

LOCAL and CONGEST model [36, 43]. In the LOCAL model, a communication network is abstracted as an n-node graph with maximum degree Δ. Nodes communicate in *synchronous rounds*, in each of which, a node can perform arbitrary local computations and send messages of arbitrary size to each of its neighbors. Initially, each node is unaware of the network topology and at the end of the computation a node has to output its own part of the solution, e.g., the colors of its incident edges in an edge coloring problem. The main complexity measure is the number of rounds until each node has produced an output. The CONGEST model is identical, with the additional restriction that messages contain $O(\log n)$ bits.

Distributed Lovász Local Lemma. There are random variables Var and (bad) events $\mathcal{X}$ at the nodes. Each event X depends on a subset $\mathsf{Var}(X)$ of the random variables. Let $p(X)$ denote the probability that event X occurs. As usual, we want to find an assignment to the variables so that none of the events occur. We form the dependency graph $H = (\mathcal{X}, E_H)$ on the events, where two events $X_1, X_2 \in \mathcal{X}$ are adjacent if they depend on a common variable, i.e., if $\mathsf{Var}(X_1) \cap \mathsf{Var}(X_2) \neq \emptyset$.

In a distributed setting, we assume that each variable and each event is associated with some node of the communication graph G. For most LLL algorithms it is essential that the dependency graph can be simulated efficiently in the communication network. In the LOCAL model, one round of communication in H can be simulated in t rounds if the variables $\mathsf{Var}(X)$ upon which the event X depends are within distance t in G of the node where X resides. Let d be the maximum degree of H, while Δ is the max degree of G.

Normally, an LLL is specified in terms of a function f, such that $p(X)f(d) \leq 1$. The original specification of Lovász has $f(d) = e \cdot d$ and ensures the existence of an assignment of the variables such that all bad events are avoided. In the study of distributed LLL algorithms, the functions d^2 [13], $c \cdot d^8$ [18] (both *polynomial criteria*), and 2^d (*exponential criterion*) [7, 9, 8] have appeared in the literature.

▶ **Example 5** (k-vertex splitting with discrepancy $6\Delta/k$ is an LLL)**.** Let each node in the graph pick one of k parts, $V_1, \ldots, V_k$, uniformly at random. Introduce a *bad event* X_v for each node $v \in V$ that holds if the number of neighbors of v within any one part deviates from the expected value by more than $6\Delta/k$, i.e., if $|N(v) \cap V_i| \neq d(v)/k \pm 6\Delta/k$, for some $i \in [k]$. Formally, there is one *variable* for each vertex indicating the part that the vertex joins. As the event X_v shares variables only with the events in its 2-hop neighborhood, the dependency degree of the LLL is $d \leq \Delta^2$. A Chernoff bound shows that $\Pr(X_v) \leq \exp(-\Omega(\Delta)) = \exp(-\Omega(\sqrt{d}))$. Hence, this splitting problem is an LLL with exponential criterion, if Δ is above an absolute constant.

The constant 6 is chosen somewhat arbitrarily in order to make the Chernoff bound-based claim simple. In the following sections, we aim at splittings with discrepancy $(1+\varepsilon)\Delta/k$.

Shattering. Our algorithms make use of the influential shattering technique[2] in which one first uses a randomized algorithm to set the values of some of the variables such that *unsolved* parts of the graph induce *small connected components*, which are solved in the *post-shattering phase*. The following lemma shows that the remaining components are indeed small.

▶ **Lemma 6** (Lemma 4.1 of [11]). *Consider a randomized procedure that generates a subset* $\mathsf{Bad} \subseteq V$ *of vertices. Suppose that for each* $v \in V$*, we have* $\Pr[v \in \mathsf{Bad}] \leq \Delta^{-3c}$*, and the events* $v \in \mathsf{Bad}$ *and* $u \in \mathsf{Bad}$ *are determined by non-overlapping sets of independent random variables for nodes with distance larger than* $2c$*. Then, w.p.* $1 - n^{-\Omega(c')}$*, each connected component in* $G[\mathsf{Bad}]$ *has size at most* $(c'/c)\Delta^{2c} \log_\Delta n$*.*

The following standard result solves these small components efficiently.

▶ **Lemma 7** ([46]). *There is a deterministic* LOCAL *LLL algorithm with polynomial criterion that runs in* $\operatorname{poly}\log N$ *rounds on instances of size* N*, even with an ID space of size exponential in* N*.*

Proof Sketch. The result follows with the derandomization of the distributed version of Moser-Tardos [41] via the network decomposition by Rozhon and Ghaffari [46], as explained in [46]. Note that the exponential ID space is not an obstacle in the LOCAL model as it can be circumvented by first computing a $\Theta(T)$-distance coloring with $\operatorname{poly} N$ colors, e.g., by using Linial's coloring algorithm [35] if the algorithm runs in T rounds. ◀

Notation and concentration bounds. Given a graph $G = (V, E)$ and a subset $S \subseteq V$ the induced graph $G[S]$ is the graph with vertex set S that contains all edges of E with both endpoints in S. Similarly, for an edge set $F \subseteq E$, the induced graph $G[F]$ is the graph with edge set F that contains all vertices that appear in an edge of F. We denote $[n] = \{0, \ldots, n-1\}$. We use the following standard concentration bounds (see, e.g., [14]).

▶ **Lemma 8** (Chernoff bounds,[14]). *Let* $\{X_i\}_{i=1}^r$ *be a family of independent binary random variables with* $\Pr[X_i = 1] = q_i$*, and let* $X = \sum_{i=1}^r X_i$*. For any* $\delta > 0$*,* $\Pr[|X - \mathbb{E}[X]| \geq \delta\,\mathbb{E}[X]] \leq 2\exp(-\min(\delta, \delta^2)\,\mathbb{E}[X]/3)$*.*

▶ **Corollary 9.** *With* X *of the same form as in Lemma 8,* $\forall \mu, z$ *s.t.* $z \leq \mu$ *and* $\mathbb{E}[X] \leq \mu$*,* $\Pr[|X - \mathbb{E}[X]| \geq z] \leq 2\exp(-z^2/(3\mu))$*.*

3 Warm-Up: Computing q-divides

For an integer $q \geq 1$, a *q-divide* of a graph is a partition of its vertices into q parts ("buckets") $V_1, \ldots, V_q$ such that each vertex has at most $8\Delta/q$ neighbors in each bucket. We show:

▶ **Theorem 10.** *For any* $\Delta \leq \operatorname{poly}\log n$ *and* $q \in [1, (1/6)\Delta/\ln\Delta]$*, there is a* LOCAL *algorithm to compute a q-divide in* $\operatorname{poly}\log\log n$*-rounds.*

We use q-divide as a subroutine in our k-splitting algorithm of Theorem 3. Additionally, the techniques to compute a q-divide serve as a warm up for the more involved algorithm for vertex splitting. There are two crucial differences between a (tight) k-splitting and a

[2] The technique has been used extensively for efficient algorithms for various local distributed graph problems and in particular symmetry breaking problems such as sinkless orientation [28], $\Delta+1$-vertex coloring [11], Δ-coloring [24], Maximal Independent Set [20], Maximal Matching [4], $(2\Delta-1)$-Edge-Coloring [4], and also for general LLL algorithms on small degree graphs [18].

q-divide: (1) A splitting guarantees both the minimum and maximum number of neighbors of a node inside each part, while a q-divide gives only an upper bound, (2) the upper bounds asked for by a q-divide are loose, i.e., we deviate by a factor 8 from a perfect partition, while a splitting is within a $(1 \pm \varepsilon)$-factor.

A q-divide is guaranteed to exist by LLL when $q \in O(\Delta/\log\Delta)$. A q-divide can also be defined as a $8\Delta/q$-defective $8\Delta/q$-frugal q-coloring, where x-frugal means that each color appears no more than x times in each neighborhood.

▶ Remark 11. For $\Delta/q = \Omega(\log n)$, there is a trivial zero round CONGEST algorithm for q-dividing. Each vertex assigns itself to a bucket uniformly at random ; each vertex has $d(v)/q \leq \Delta/q$ neighbors in each bucket in expectation. By Lemma 8 (Chernoff bound), for each vertex v and $i \in [q]$, $\Pr[|N(v) \cap V_i| > 8\Delta/q] \leq \exp(-\Delta/q) \in n^{-\Omega(1)}$. Therefore, w.h.p., no vertex has more than $8\Delta/q$ neighbors in a bucket.

For smaller Δ/q we give an algorithm based on shattering (that also works for large Δ/q).

The algorithm is parameterized with a threshold parameter $z(v)$ for each vertex v. For Theorem 10 we set $z(v) = 8\Delta/q$ for all nodes. In the full version, we compute slightly different versions of q-divides with different choices of $z(v)$.

Algorithm. **Phase I:** (Pre-shattering) Each vertex picks one of the first $q/2$ buckets u.a.r. Whenever a node has more than $z(v)$ neighbors in a bucket, it deselects these, i.e., these neighbors are removed from the bucket. **Phase II:** (Post-shattering) The post-shattering instance is formed by all nodes that are not assigned to any bucket, together with their neighbors. The objective is to add each unassigned node to one of the last $q/2$ buckets, such that each node has at most $z(v)$ neighbors in each bucket. In Lemma 13, we show that this problem is an LLL instance with a polynomial criterion, and in Lemma 12 that it is induced by connected components of small size. We solve it via Lemma 7 in LOCAL.

▶ **Lemma 12.** *For threshold discrepancy $z(v) = 8\Delta/q$ for all $v \in V$, the connected components participating in the post-shattering phase of the algorithm are of size $\mathrm{poly}(\Delta) \cdot \log n$, w.h.p.*

Proof. For $j \in [q/2]$ and node v, let $D_j(v)$ be the number of neighbors of v in bucket j. We have $E[D_j] = 2\Delta/q$ for each of the $q/2$ buckets. By Chernoff (Lemma 8), a node v has an unusually high number of neighbors ($> z(v) = 8\Delta/q = (1+3)2\Delta/q$) in a given bucket w.p. at most $\exp(-2\Delta/q) \leq \Delta^{-12}$, using $q \leq (1/6)\Delta/\ln\Delta$. A node v takes part in the post-shattering phase if one of its neighbors or v itself renounced its choice of bucket, i.e., if a node in its distance-2 neighborhood had too many neighbors in one of the buckets. This occurs w.p. at most $q \cdot \Delta^2 \cdot \Delta^{-12} \leq \Delta^{-9}$, and is fully determined by the random choices of nodes inside the 3-hop ball around v. Hence, by Lemma 6, the graph is shattered into components of size $O(\Delta^6 \log n)$, w.h.p. ◀

▶ **Lemma 13.** *When $z(v) = 8\Delta/q$, for all $v \in V$, the instances formed in the post-shattering phase are LLL problems with criterion $f(d) = (q/2)\exp(-2\sqrt{d}/q)$ and $d \leq \Delta^2$. For $q \leq (1/6)\Delta/\ln\Delta$, the error probability of the LLL is upper bounded by d^{-5}.*

Proof. Consider the following probabilistic process. Each node picks each part in $[q] \setminus [q/2]$ u.a.r., i.e., with probability $p = 2/q$. For $j \in [q] \setminus [q/2]$, let D_j denote the random variable describing the number of neighbors in bucket j. We have $E[D_j] \leq 2\Delta/q$. Let X_v denote the "bad" event that node v has more than $z(v) = 8\Delta/q$ neighbors in one of the $q/2$ buckets. We analyze the LLL formed by the events X_v and their underlying variables.

The event X_v is fully determined by the random choices of direct neighbors of v. Hence, two bad events X_v and X_w are dependent on a shared variable iff v and w are at distance 2 or less, and each bad event shares a variable with at most Δ^2 other events. Therefore,

the dependency graph of the LLL has maximum degree at most $d \leq \Delta^2$. By Chernoff (Lemma 8), X_v occurs w.p. at most $(q/2)\exp(-2\Delta/q)$. Hence, the LLL has criterion $f(d) = (q/2)\exp(-2\sqrt{d}/q)$, which ranges from being polynomial to exponential depending on how small q is compared to $\Delta \geq \sqrt{d}$. In the worst case, the bound $q \leq (1/6)\Delta/\ln\Delta$ implies that $f(d) \leq (\Delta/(12\ln\Delta))\Delta^{-12} \leq d^{-5}$. ◀

Proof of Theorem 10. The problem is solved by the algorithm above. The runtime is $O(1)$ rounds for the pre-shattering phase, and $\operatorname{poly}\log\log n$ rounds for the post-shattering phase via Lemma 7. To apply this lemma we require Lemma 12 that shows that any component in the post-shattering phase has size $\log n \cdot \operatorname{poly}\Delta = \operatorname{poly}\log n$, w.h.p., and that Lemma 13 shows that these components form LLLs with a polynomial criterion. ◀

Note that in the special case of $\Delta/q = \Omega(\log n)$ we get the stronger property that, w.h.p., there will no post-shattering phase (see Remark 11).

4 Vertex Splitting in LOCAL

In this section, we prove the following result on vertex splitting.

▶ **Theorem 3.** *There exists a universal constant $c_3 > 0$ s.t.: For any $\varepsilon > 0$, maximum degree $\Delta \leq \operatorname{poly}\log n$, and $k \leq c_3 \cdot (\varepsilon^4\Delta/\ln\Delta)$, there is a distributed randomized* LOCAL *algorithm to compute a k-vertex splitting with discrepancy $\varepsilon\Delta/k$ in $O(1/\varepsilon) + \operatorname{poly}\log\log n$ rounds.*

When Δ is logarithmically larger than k, there is an easy solution.

▶ **Observation 14.** *If $k \leq \varepsilon^2\Delta/(9\ln n)$, the trivial zero round algorithm in which each node picks one of the k parts u.a.r. results in a k-vertex splitting with discrepancy $\varepsilon\Delta/k$, w.h.p.*

Proof. For a node v and class i, let D be the number of neighbors of v that picked class i. Then $\mathbb{E}[D] = d_v/k$. Let $\mu := \Delta/k$, $z = \varepsilon\Delta/k$. By Corollary 9 (Chernoff bound)

$$\Pr[|D - \mathbb{E}[D]| \geq \epsilon\Delta/k] \leq 2\exp(-z^2/(3\mu)) = 2\exp(-\varepsilon^2\Delta/(3k)) \ .$$

This is at most $2n^{-3}$ when $k \leq \varepsilon^2\Delta/(9\ln n)$, so by union bound over all nodes v and classes i we get a k-splitting w.h.p. ◀

4.1 Shattering for ε-Vertex-Splitting in $O(1/\varepsilon)$ Rounds

Due to Observation 14, the most challenging case for a a $\operatorname{poly}\log\log n$-round algorithm is when $\Delta \leq \operatorname{poly}\log n$ and $\Delta/k = O(\log n)$ holds. Next, we present our algorithm.

FastShattering. Find a q-divide χ for $q = 24/\varepsilon$. To avoid confusion between this partition of the nodes and that of the k-splitting we are computing, let us refer to χ as a *schedule* of the nodes, made of q *slots*, which we denote by $N_1, \ldots, N_q$. Go through the q slots of χ sequentially, and temporarily assign each node in this slot one of the k parts uniformly at random. If a node has received too few or too many neighbors in a part when processing a slot, we retract the last batch of assignments within the neighborhood of that node and freeze those nodes. We also freeze all nodes within distance 3 that are in later slots. All non-frozen nodes (in slot j) keep their assignment permanently. The frozen nodes then get solved in post-shattering (along with all neighbors acting as events, including non-frozen neighbors). For each $j \in [q]$, there is one such post-shattering instance stemming from nodes that were frozen when processing slot j.

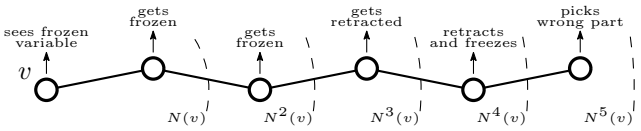

Figure 1 Whether a node joins the post-shattering instance depends on random choices at distance up to 5.

For the rest of this section, set the number of slots to $q = 24/\varepsilon$ and define the following threshold parameters for the pre-shattering and post-shattering phase $z_j^{\mathsf{pre}}(v) = z_j^{\mathsf{post}}(v) = \varepsilon^2\Delta/(72k)$ for all $j \in [q], v \in V$.

Detailed description of FastShattering. During the course of the algorithm nodes are either `frozen` or non-`frozen`. Initially, all nodes are non-`frozen`. Pre-shattering: After computing the q-divide, we iterate through the slots $1, \dots, q$. In each iteration, we temporarily assign the non-`frozen` nodes in slot j by sampling each u.a.r. into one of the k parts. Next, we formalize the event that retracts these assignments. Fix a node $v \in V$, a slot $j \in [q]$ and a part $i \in [k]$. Let $N_j(v)$ be the neighbors of node v in slot j. Let $\hat{d}_j(v)$ denote the *live degree* of node v when processing slot j, i.e., the number of vertices in $N_j(v)$ that are not `frozen` just before processing slot j. Let $D_{i,j}$ denote the number of neighbors of v in $N_j(v)$ that are temporarily assigned to part i. Event $\mathcal{B}_{i,j}^{\mathsf{pre}}(v)$ holds if $D_{i,j}$ deviates from its expectation $\mathbb{E}[D_{i,j}] = \hat{d}_j(v)/k$ by more than the threshold parameter $z_j^{\mathsf{pre}}(v)$. Let $\mathcal{B}_j^{\mathsf{pre}}(v) = \bigvee_{i\in k} \mathcal{B}_{i,j}^{\mathsf{pre}}(v)$ be the event that v sees such a large deviation from its expectation in some part i. Suppose after sampling event $\mathcal{B}_j^{\mathsf{pre}}(v)$ occurs, then node v undoes the temporal assignment of all neighbors in slot j, i.e., of all nodes in $N_j(v)$, and additionally freezes all unassigned nodes in distance 3, i.e., the nodes in $\{u \in \cup_{j'>j} N_j(v) : d(u,v) \leq 3\}$. Add all nodes that become frozen when processing slot j to Bad_j. All (temporal) assignments that do not undergo a retraction are kept permanently. While `frozen` nodes do not sample colors, each node monitors how its neighbors are being colored and thus yields an *event node* in each of the q iterations, regardless of whether it is `frozen` or not.

Post-shattering: For each $j \in [q]$ there is a separate post-shattering LLL instance containing a variable for each node in Bad_j and a bad event node $\mathcal{B}_j^v$ for each node v with a neighbor in Bad_j. The random process of the j-th LLL is as follows: Each node in Bad_j picks one of the k parts independently and u.a.r. For a node v the number of neighbors in Bad_j is denoted by $f_j(v)$. Let $F_{i,j}(v)$ be the number of neighbors of v in part i (restricted to neighbors in Bad_j). Event $\mathcal{B}_{i,j}^{\mathsf{post}}(v)$ holds if $F_{i,j}$ deviates from its expectation $\mathbb{E}[F_{i,j}(v)] = f_j(v)/k$ by more than the threshold parameter $z_j^{\mathsf{post}}(v)$. The *bad event* $\mathcal{B}_j^{\mathsf{post}}(v) = \bigvee_{i\in k} \mathcal{B}_{i,j}^{\mathsf{post}}(v)$ holds if v sees such a large deviation from its expectation in some part i. In Lemmas 6 and 19 we show that for each $j \in [q]$ we indeed obtain an LLL with polynomial criterion that can be solved via Lemma 7 in the LOCAL model. All q instances are solved in parallel; their deviations add up to $(\varepsilon/3) \cdot \Delta/k$ as shown in Lemma 16.

Intuition for the runtime: The pre-shattering phase runs in $O(q) = O(1/\varepsilon)$ rounds. The post-shattering phase runs in $\operatorname{poly}\log\log n$ rounds for the following reason. Each component in each of the q post-shattering instances forms an LLL and is of size $N = \operatorname{poly}(\Delta) \cdot \log n = \operatorname{poly}\log n$, see Lemma 18. As all components are independent, they can can be solved in parallel in $\operatorname{poly}\log N = \operatorname{poly}\log\log n$ rounds in the LOCAL model via Lemma 7.

Notation. We summarize and extend the notation that we need for the analysis.

- $V_1, \dots, V_k$ parts (changing throughout the algorithm),
- $N(v)$ neighbors of v in G, $N_j(v)$ neighbors of v in slot j, $\hat{N}_j(v) \subseteq N_j(v)$ are the live neighbors of v in bucket j, i.e., the unfrozen neighbors of v in slot j just before slot j is processed, $F_j(v) \subseteq N(v)$ neighbors of v in the j-th post-shattering instance,
- $d(v) = |N(v)|$, $d_j(v) = |N_j(v)|$, $\hat{d}_j(v) = |\hat{N}_j(v)|$, $f_j(v) = |\hat{F}_j(v)|$.

▶ **Observation 15.** *In FastShattering, any node can have at most one slot in which (some) of its neighbors get their part assignment undone.*

Proof. Let v be a node with a neighbor $u \in N_j(v)$ that has its assignment retracted during slot j. Then u is adjacent to a node that detected that too few or too many of its neighbors were assigned a given part when processing slot j. That node is at distance at most 2 from v, and it freezes the nodes in slots higher than j within distance 3. Therefore, all the unassigned neighbors of v are frozen, and v will not see another retraction in its neighborhood (in fact, it will not even see an assignment). ◀

4.2 Analysis of Discrepancy

In this section, we bound the deviation in the number of neighbors that a node v sees in the i-th part from $d(v)/k$.

The full proof of the following lemma appears in Appendix C.

▶ **Lemma 16.** *In the final assignment $V_1, \dots, V_k$, i.e., after the pre- and post-shattering phases, each node v has $d(v)/k \pm \varepsilon\Delta/k$ neighbors in every V_i, $i \in [k]$.*

Proof sketch. The discrepancy (deviation from expectation) for a node comes from three sources, (a) slots with neighbors that got retracted, (b) the parts of other slots assigned in the pre-shattering phase, and (c) the deviation summed up over all q post-shattering instances. Due to Observation 15, there can be at most one slot with retracted nodes and the deviation from that slot j^* from the expectation can be upper bounded by $d_{j^*}(v)/k + z \leq 8\Delta/q + z \leq \varepsilon/3\Delta + z$. For each other slot the discrepancy is at most z, and for each of the q instances in the post-shattering phase the discrepancy is also at most z. Thus, with $\sum_{j\in[q]} z_j^{\mathsf{pre}}(v) = \sum_{j\in[q]} z_j^{\mathsf{post}}(v) \leq \varepsilon/3 \cdot \Delta/k$ and $q = 24/\varepsilon$ the total discrepancy adds up to $8\Delta/q + \sum_{j\in[q]} z_j^{\mathsf{pre}}(v) + \sum_{j\in[q]} z_j^{\mathsf{post}}(v) \leq \varepsilon\Delta/k$. ◀

4.3 Analysis of Bad Event Probabilities

Throughout our analysis of the pre-shattering and post-shattering parts of our algorithm, we consider random processes and events which are essentially always the same: nodes in some subgraph each pick a random bucket u.a.r. independently from other nodes, and for each node we analyze the probability that the number of neighbors that pick a given bucket deviates too much from expectation. Recall, that we set $q = 24/\varepsilon$ and $z = \varepsilon^2\Delta/(72k)$ earlier.

▷ Claim 17. Let k, N be positive integers. Let D be a sum of at most N independent Bernouilli random variables of parameter $1/k$, and let $z \leq N/k$. Consider the event $\mathcal{B}$ that D deviates from its expectation by more than z. $\Pr(\mathcal{B}) \leq 2e^{-z^2k/(3N)}$.

In particular, for $N = \Delta$, $k \leq \varepsilon^4\Delta/(2^{19}\ln\Delta)$ and $z = \varepsilon^2\Delta/(72k)$ we obtain $\Pr(\mathcal{B}) \leq \Delta^{-24}$. If D is a sum of only $N = 8\Delta/q$ variables, $k \leq \varepsilon^3\Delta/(2^{17}\ln\Delta)$ suffices for the same bound.

Throughout this paper, D is taken to be a sum of indicator random variables associated to a set of nodes. More precisely, for a subset of nodes in a neighborhood $N(v)$, we consider the sum of the random variable indicating whether each node chose a specific part i out of k choices.

Proof. The general bound on $\Pr(\mathcal{B})$ is from Corollary 9 (Chernoff bound).

When $N = \Delta$, $k \leq \varepsilon^4\Delta/(2^{19}\ln\Delta)$ and $z = \varepsilon^2\Delta/(72k)$, $\exp(-z^2/(3N))$ simplifies to $\exp(-\varepsilon^4\Delta/(3\cdot 72^2 k)) \leq \Delta^{-24}$. When $N = 8\Delta/q$ (recall $q = 24/\varepsilon$), $k \leq \varepsilon^3\Delta/(2^{17}\ln\Delta)$ and the same z as before, $\exp(-z^2k/(3N))$ simplifies to $\exp(-\varepsilon^3\Delta/(72^2k)) \leq \Delta^{-24}$. ◁

4.4 Analysis of FastShattering

Next, we show that the post-shattering instances consist of small connected components.

▶ **Lemma 18.** *After FastShattering, each connected component in each of the q post-shattering instances is of size $\Delta^{10}\log n$, w.h.p.*

The proof of Lemma 18 appears in Appendix C.2. In spirit it is similar to the proof of Lemma 12, but it is more advanced as frozen variables need to be taken care of formally.

4.5 Post-shattering

In this section we show that the q post-shattering instances are indeed LLLs.

▶ **Lemma 19.** *Each connected component in each of the $q = O(1/\varepsilon)$ post-shattering instances forms an LLL with dependency degree d', bad events' probabilities upper bounded by p' such that the polynomial criterion $d'^8 p' < 1$ holds. In* LOCAL*, the dependency graph can be simulated with $O(1)$ overhead in the communication network G.*

Proof. Consider a post-shattering instance $j \in [q]$. The LLL is formally defined in Section 4.1. Recall, that in the associated random process each node in Bad_j joins one of the k parts u.a.r. and that there is a bad event $\mathcal{B}^{\mathsf{post}}(v) = \bigvee_{i\in[k]} \mathcal{B}_i^{\mathsf{post}}(v)$ for each node with neighbor in Bad_j. The event $\mathcal{B}^{\mathsf{post}}(v)$ occurs if too many or too few neighbors join the i-th part. Thus, a bad event only depends on the randomness of adjacent nodes and the dependency degree is at most $d' = \Delta^2$.

By Claim 17 (applied with $N = \Delta$, and $z = z_j^{\mathsf{post}}(v) = \varepsilon^2\Delta/(72k)$), the probability that $\mathcal{B}_i^{\mathsf{post}}(v)$ holds is at most Δ^{-24} if $k \leq \varepsilon^4\Delta/(2^{19}\log\Delta)$. With a union bound over all k parts we obtain the upper bound $p' = k\Delta^{-24} = \Delta^{-23}$ for the probability of each bad event.

Hence, we obtain $p'd'^{11} < 1$. ◀

4.6 Proof of Theorem 3

Assume $k \leq \varepsilon^4\Delta/(2^{19}\log\Delta)$ and recall that $\Delta \leq \operatorname{poly}\log n$. The runtime of Fast-Shattering is linear in the number of slots, i.e., $O(q) = O(1/\varepsilon)$. Next, we show that the post-shattering instances meet the requirements of Lemma 7. Due to Lemma 18 each connected component is of size $\operatorname{poly}(\Delta)\log n = \operatorname{poly}\log n$, w.h.p. Further, due to Lemma 19 each such component forms an LLL with polynomial criterion and the dependency graph can be simulated with $O(1)$ overhead in the communication network G. Thus, we can apply Lemma 7 (in parallel for all q instances) and obtain a runtime of $\operatorname{poly}\log\log n$ rounds for the post-shattering phase. Lemma 16 shows that the deviation of $|N(v)\cap V_i|$ from $d(v)/k$ is upper bounded by $\varepsilon\Delta/k$ for any node $v \in V$.

5 Vertex Splitting in CONGEST

We obtain the following theorem for vertex splitting and bipartite vertex splitting.

▶ **Theorem 20.** *There exists a universal constant $c_4 > 0$ s.t.: For $\varepsilon > 0$, $\Delta \leq \operatorname{poly}\log n$, and $k \leq c_4 \cdot (\varepsilon^4 \Delta / (\ln \Delta \log^2 \log n))$, there are distributed* CONGEST *algorithms to solve the k-vertex splitting problem with discrepancy $\varepsilon\Delta/k$ and to solve the bipartite k-vertex splitting problem with discrepancy $\varepsilon\Delta_{\mathsf{L}}/k$ in $O(1/\varepsilon) + \operatorname{poly}\log\log n$ rounds.*

Theorem 20 requires k to be a $O(\log^2 \log n)$ factor smaller than in Theorems 3 and 4.

As the pre-shattering phase of Theorems 3 and 4 immediately works in the CONGEST model, the main challenge to prove Theorem 20 is to design a new post-shattering method. Recall, the post-shattering phase in the LOCAL model, i.e, the core steps of Lemma 7. Each connected component in the post-shattering phase forms an LLL with a polynomial criterion and has $N = \operatorname{poly}(\Delta) \cdot \log n = (\operatorname{poly}\log\log n) \cdot \log n$ nodes. This small size allows to compute a network decomposition (see Section 5.1) with $\operatorname{poly}\log\log n$ cluster diameter and $O(\log\log n)$ color classes with distance $s = \Omega(\log N) = \Omega(\log\log n)$ between clusters of the same color. The latter is sufficient to derandomize the $O(\log N) = O(\log\log n) \ll s$ round LLL algorithm from [13]. The details of the derandomization are not important, but it is based on *gathering all information in the cluster and close-by nodes*. In the LOCAL model, this can be done in time that is linear in the cluster diameter, i.e., in $\operatorname{poly}\log\log n$ rounds. One can show that in the CONGEST model all information of a cluster can be encoded with $N \cdot \operatorname{poly}\log\log n$ bits. By using a pipelining argument (see the full version and [39] for details) and that the bandwidth of the CONGEST model is $\Theta(\log n)$ bits, one can aggregate all of this information at a cluster leader in $N \cdot \operatorname{poly}\log\log n/\mathsf{bandwidth} + \mathsf{clusterdiameter}$ rounds, as done in [39]. For $\Delta = \operatorname{poly}\log\log n$, we obtain $N = \log n \cdot \operatorname{poly}\log\log n$ and this method runs in $\operatorname{poly}\log\log n$ rounds. In summary, we obtain the following theorem[3] and the corollary thereafter.

▶ **Lemma 21** ([39])**.** *There is a randomized* CONGEST *algorithm with bandwidth $= \Theta(\log n)$ for LLL instances of size $N \leq \log n \cdot \operatorname{poly}\log\log n$, dependency degree $d \leq \operatorname{poly}\log\log n$ and error probability $p < d^{-4}$, that runs in $\operatorname{poly}\log\log n$ rounds.*

The algorithm works with an ID space that is exponential in N and is correct w.h.p in n.

▶ **Corollary 22.** *There is a randomized* CONGEST *algorithms for LLL with error probability p, dependency degree d and criterion $p < d^{-8}$ that uses $\operatorname{poly}\log\log n$ rounds, whenever $d \leq \operatorname{poly}\log\log n$. Here, the dependency graph is also the communication network.*

Proof. The shattering framework of [18], w.h.p., reduces to the LLL problem to LLL problems with error probability p', the same dependency degree d and criterion $p' < d^{-4}$ on instances of size $N = \log n \cdot \operatorname{poly} d$. These can be solved in $\operatorname{poly}\log\log n$ rounds via Lemma 21. ◀

For $\Delta \gg \operatorname{poly}\log\log n$, any such *gather all information* approach inherently requires significantly larger runtimes. The main ingredient for Theorem 20 is a new method for solving the vertex splitting instances in the post-shattering phase, that can deal with degrees as large as $\Delta = \operatorname{poly}\log\log n$ while using only $\operatorname{poly}\log\log n$ rounds. We prove the following theorem.

[3] The proof of Lemma 21 appears in the full version [30]. It is similar to an CONGEST LLL algorithm in [39] for instances of size $N = O(\log n)$ and the case of $d = O(1)$. In fact, following all dependencies on d (and a slightly increased N) in the proof of [39] yields an algorithm with runtime $\operatorname{poly}(d, \log\log n)$, which yields the desired runtime whenever $d = \operatorname{poly}\log\log n$.

▶ **Lemma 23.** *There exists a universal constant $c_5 > 0$ s.t.: For any $\varepsilon > 0$ and any $k \leq c_5 \cdot (\varepsilon^2 \Delta / (\log \Delta \log^2 \log n))$, there is a* $\operatorname{poly}\log\log n$*-round randomized* CONGEST *algorithm with bandwidth $= \Theta(\log n)$ that computes a k-vertex splitting with discrepancy $\varepsilon\Delta/k$ on instances of size $N \leq \operatorname{poly}\log n$.*

The algorithm works with an ID space that is exponential in N and is correct w.h.p in n.

The proof of Lemma 23 uses network decompositions that we introduce in Section 5.1, before proving the lemma in Section 5.2. In Section 5.3, we prove Theorem 20.

5.1 Network Decomposition

A *weak distance-s (C, β)-network decomposition with congestion κ* is a partition of the vertex set of a graph into clusters $\mathcal{C}_1, \ldots, \mathcal{C}_p$ 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 s 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 [39, 22]. For the sake of our proofs we only require that such decompositions can be computed efficiently (Theorem 24) 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 the full version [30] for the precise statement).

▶ **Theorem 24** ([39])**.** *For any constant $C > 0$ and $s \in \operatorname{poly}\log\log n$, 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 $(C \log n, O(s/C \cdot \log^3 n))$-network decomposition with cluster distance s and congestion $O(s \cdot \log^2 n)$ in $O(\log^7 n \cdot s^2)$ rounds.*

5.2 Efficient Post-shattering in CONGEST (Proof of Lemma 23)

In order to devise an efficient CONGEST post-shattering algorithm, we decompose each small component into small clusters via the network decomposition algorithm from Theorem 24. Then, the objective is to iterate through the color classes of the decomposition and when processing a cluster we want to assign all nodes in that cluster to a part. When doing so we ensure that each node of the graph obtains a discrepancy of at most $(\varepsilon/Q)\Delta/k$ in each iteration. Hence, over the Q iterations, each node's discrepancy adds up to at most $\varepsilon\Delta/k$.

Proof of Lemma 23. First, compute a distance-3 network decomposition of the graph with $Q = 2\log\log n$ colors via Theorem 24. Then, iterate through the color classes of the network decomposition, processing all clusters of a color class as it gets considered.

When processing a cluster $\mathcal{C}$, we set up a new instance of the vertex splitting problem as follows: Let $V^{\mathsf{L},\mathcal{C}} = N(\mathcal{C})$ be all nodes that have a neighbor in $\mathcal{C}$; $V^{\mathsf{L},\mathcal{C}}$ may contain many nodes of $\mathcal{C}$ itself. Each node of $\mathcal{C}$ is supposed to join one one the parts $V_1^{\mathcal{C}}, \ldots, V_k^{\mathcal{C}}$ such that for each $i \in [k]$ each node in $v \in V^{\mathsf{L},\mathcal{C}}$ has $d_{\mathcal{C}}(v)/k \pm \varepsilon/Q \cdot \Delta/k$ neighbors in $V_i^{\mathcal{C}}$. After processing all clusters we set $V_i = \bigcup_{\text{cluster } \mathcal{C}} V_i^{\mathcal{C}}$. As clusters processed at the same time are in the same color class, they have distance-3, and no node has neighbors in more than one simultaneously processed cluster. Hence, the deviation of the number of neighbors into one V_i from $d(v)/k$ is bounded by $Q \cdot \varepsilon/Q \cdot \Delta/k = \varepsilon\Delta/k$.

The bounds on k and Q imply that the problem that we solve when processing one cluster is an LLL $\mathcal{L}_{\mathcal{C}}$ with a polynomial criterion: Variables and the random process are given by the nodes of $\mathcal{C}$ choosing one of the parts $V_1^{\mathcal{C}}, \ldots, V_k^{\mathcal{C}}$ uniformly at random. For a node $v \in V^{\mathsf{L},\mathcal{C}}$ introduce a bad event $\mathcal{B}_v^{\mathcal{C}}$ that holds if for any $i \in [k]$ node v does not have $d_{\mathcal{C}}(v)/k \pm \varepsilon/Q \cdot \Delta/k$ neighbors in part $V_i^{\mathcal{C}}$. Due to the distance between clusters no node can have a bad event for more than one of the simultaneously processed clusters.

Due to Claim 17 (applied with $N = \Delta$, $z = \varepsilon/Q \cdot \Delta/k$), we obtain that $\Pr(\mathcal{B}_v^{\mathcal{C}}) \leq k \cdot \exp(-\varepsilon^2\Delta/(3Q^2k))$. Plugging in $Q = 2\log\log n$ and $k \leq C\varepsilon^2\Delta/(\log\Delta\log^2\log n)$, we get $\Pr(\mathcal{B}_v^{\mathcal{C}}) \leq k \cdot \exp(-\log\Delta/(12C)) \leq \Delta^{-19}$ for $C \leq 2^{-8}$. As the dependency degree is at most Δ^2, we obtain an LLL with a polynomial criterion of exponent 9.

The goal is to assign all nodes of $\mathcal{C}$ to a part such that all bad events $\mathcal{B}^{\mathcal{C}}(v)$ for $v \in V^{\mathsf{L},\mathcal{C}}$ are avoided. In order to do so, we run $\ell = 6\log n$ parallel instances of the LLL algorithm of [13] on $\mathcal{L}_{\mathcal{C}}$, each running for $O(\log N) = O(\log\log n)$ rounds. At the end of the proof we reason that these ℓ instances can indeed be run efficiently in parallel, for now, we continue with the remaining steps of the algorithm. We say that an instance is *correct* for an event of $\mathcal{L}_{\mathcal{C}}$ if it is avoided under the computed assignment of the instance. By the properties of the algorithm of [13], each instance is correct for all events of $\mathcal{L}_{\mathcal{C}}$ with probability $\geq (1 - 1/N) \geq 1/2$. Hence, with probability $1 - 1/2^{\ell} = 1 - 1/n^6$ one of the ℓ instances is *correct* for all events of $\mathcal{L}_{\mathcal{C}}$. Then, each node holding an event of $\mathcal{L}_{\mathcal{C}}$ determines which instances are correct, and the nodes agree on a *winning* instance, i.e., one that is correct for all of them.

Assume that nodes know in which instance their bad events are avoided. Then, agreeing on a winning instance can be done efficiently as follows: Let each such node hold a bit string of length $\ell = O(\log n)$ in which the j-th bit indicates whether the bad event is avoided in the outcome of the j-th instance. All nodes can agree on a winning instance in time linear in the cluster's weak diameter by computing a bitwise-AND of the bitstrings (see the full version for details).

In order to determine the status of its events in each of the ℓ instances, node v only needs to know which part each neighbor has chosen in which instance. As there are only k parts, the index of the part can be communicated with $O(\log k)$ bits. Hence, a node u can inform each neighbor about the parts node u chose in all ℓ instances by communicating $\ell \cdot O(\log k) = O(\log n \log\log n)$ bits over each incident edge. Using $\mathsf{bandwidth} = \Theta(\log n)$, this requires $O(\log\log n)$ rounds. The same reasoning is also sufficient to run the ℓ instances of [13] in parallel. In one iteration of [13], the variables of local ID minima in the graph induced by violated events are re-sampled. We just reasoned that a node can determine the status of its events in each of the ℓ instances in $O(\log\log n)$ rounds, and with an additional round we can compute a set of local ID minima of violated events for each instance. Then, nodes can inform neighbors about the instances in which they need to re-sample their part. ◀

5.3 Proof of Theorem 20

The pre-shattering phase of computing the q-divide can immediately be implemented in the CONGEST model. Its post-shattering phase is replaced with the stronger q-vertex splitting result of Lemma 23 (with $\varepsilon = 1$ and $k = q = 24/\varepsilon$) that runs in $\operatorname{poly}\log\log n$ rounds. Note that the hypotheses of Theorem 20 assume that $\Delta/(\log\Delta\log^2\log n)$ is greater than an absolute constant $1/c_4$. With $c_4 \leq c_5/24$, q satisfies the hypotheses of Lemma 23.

The pre-shattering of the main algorithm can also immediately be implemented in the CONGEST model. For each of its q post-shattering instances we use Lemma 23 with $\varepsilon^2/72$ and the same k. Using the proof of Lemma 16, the total discrepancy of the pre-shattering and the post-shattering phase is upper bounded by $(2\varepsilon/3)\Delta/k$ and $(\varepsilon/3)\Delta/k$, respectively.

6 Application: $(1+\varepsilon)\Delta$-edge coloring

In this section, we first prove the LOCAL version of the following theorem.

▶ **Theorem 1** (Edge coloring). *For any constant $\varepsilon > 0$, there is a* $\operatorname{poly}\log\log n$*-round randomized* CONGEST *algorithm to compute a $(1+\varepsilon)\Delta$-edge coloring on any graph with maximum degree $\Delta \geq \Delta_0$ where Δ_0 is a sufficiently large constant.*

We use the following result based on prior work to color small degree graphs.

▶ **Theorem 25** ([18, 15, 10]). *For any constant $\varepsilon > 0$, there is an absolute constant Δ_0 such that for $\Delta \geq \Delta_0$, there is a randomized* LOCAL *algorithm with runtime $O(d^2) + \operatorname{poly}\log\log n$ for $(1+\varepsilon)\Delta$-edge coloring where $d = \operatorname{poly}\Delta$.*

The papers [15, 10] both solve the $(1+\varepsilon)\Delta$-edge coloring problem via a constant number of LLL iterations (for constant $\varepsilon > 0$). Their dependency graph can be simulated in the original network with $O(1)$ overhead and has dependency degree $d = \operatorname{poly}\Delta$. Plugging in the runtime of $O(d^2) + \operatorname{poly}\log\log n$ for solving such LLLs by [18] yields Theorem 25.

High level overview $(1+\varepsilon)\Delta$-edge coloring algorithm. We recursively (two recursion levels) partition the edge set of G into parts that induce small degree subgraphs. Then, we color each subgraph with a disjoint color palette. More detailed, first we partition the edge set into $k = \Theta(\varepsilon^2\Delta/\log n)$ parts such that each part induces a graph of maximum degree at most $\Delta' = \operatorname{poly}\log n$. Then, in another recursive step we partition the edge set of each of these parts further into $k' = \Theta(\varepsilon^4\Delta'/\log^2\log n)$ parts, each with maximum degree $\Delta'' = \operatorname{poly}\log\log n$. We obtain $k \cdot k'$ subgraphs, each with maximum degree at most Δ''. We color each part with a disjoint color palette with $(1+\varepsilon/10)\Delta''$ colors via Theorem 25 in $O((\Delta'')^2) + \operatorname{poly}\log\log n = \operatorname{poly}\log\log n$ rounds. The colors of the $k \cdot k'$ subgraphs sum up to $(1+\varepsilon)\Delta$ colors in total.

Proof of Theorem 1, LOCAL**.** If $\Delta \leq \operatorname{poly}\log\log n$ we skip the first two steps of the algorithm and immediately apply Theorem 25 to compute a $(1+\varepsilon)\Delta$-edge coloring in $\operatorname{poly}\log\log n$ rounds. If $\Delta \leq \operatorname{poly}\log n$, we skip the first step and set $\Delta' = \Delta$, $k = 1$ and $G_1 = G$, otherwise we first partition the graph into $k = (\varepsilon/6)^2\Delta/(9\log n)$ subgraphs $G_1, \ldots, G_k$, each with maximum degree $\Delta' = \Delta/k + (\varepsilon/6)\cdot\Delta/k = \operatorname{poly}\log n$. To this end, let each edge uniformly at random and independently join one of the G_i's. The same Chernoff bound as in Observation 14 shows that w.h.p., the maximum degree of each G_i is upper bounded by Δ'.

In the next step, we use Theorem 20 to split each $G_i, i \in [k]$ in parallel into $k' = c_4(\varepsilon')^4\Delta'/\log^2\log n$ graphs $G_{i,j}, j \in [k']$, each of maximum degree $\Delta'' = \Delta'/k' + \varepsilon'\Delta'/k' = \operatorname{poly}\log\log n$. We set $\varepsilon' = \varepsilon/6$. Recall, that c_4 is the constant from Theorem 20. More formally, we set up the following k bipartite splitting instances $B_i = (V_i^{\mathsf{L}} \cup V_i^{\mathsf{R}}, E_i)$, $i \in [k]$: $V_i^{\mathsf{R}} = E(G_i)$ and $V_i^{\mathsf{L}} = V(G_i)$. Note that the degree $d^{B_i}(v) = d^{G_i}(v)$ for a node $v \in V_i^{\mathsf{R}}$ and $d^{B_i}(e) = 2$ for a node $e \in V_i^{\mathsf{R}}$. Hence, B_i has maximum degree Δ'.

We use Theorem 4 (for each B_i in parallel and with the same k' and ε') to compute a partition of V_i^{R} into $V_{i,1}^{\mathsf{R}}, \ldots, V_{i,k'}^{\mathsf{R}}$ such that each $v \in V_i^{\mathsf{L}}$ has $d^{B_i}(v)/k' \pm \varepsilon'\Delta'/k' = d^G(v)/(k\cdot k') \pm 3\varepsilon'\Delta/(k\cdot k')$ neighbors in each V_i^{R}. Now, for $i \in [k], j \in [k']$ let $G_{i,j} = G_i[V_{i,j}^{\mathsf{R}}]$ and note that $G_{i,j}$ has maximum degree at most $\Delta'' = \Delta'/k' + \varepsilon'\Delta'/k' = \operatorname{poly}\log\log n$.

In the last step, we apply Theorem 25 on each $G_{i,j}, i \in [k], j \in [k']$ in parallel to edge-color $G_{i,j}$ with $(1+\varepsilon/6)\Delta''$ colors in $\operatorname{poly}\Delta'' + \operatorname{poly}\log\log n = \operatorname{poly}\log\log n$ rounds.

The total number of colors used is upper bounded by

$$k \cdot k' \cdot (1+\varepsilon/6)\Delta'' \leq k \cdot (1+\varepsilon/6)^2 \cdot \Delta' \leq (1+\varepsilon/6)^3 \cdot \Delta \leq (1+\varepsilon)\Delta\ . \qquad \blacktriangleleft$$

References

1 Noga Alon. A parallel algorithmic version of the local lemma. *Random Structures & Algorithms*, 2(4):367–378, 1991.

2 Philipp Bamberger, Mohsen Ghaffari, Fabian Kuhn, Yannic Maus, and Jara Uitto. On the complexity of distributed splitting problems. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, 2019. doi:10.1145/3293611.3331630.

3 Leonid Barenboim and Michael Elkin. Distributed $(\Delta + 1)$-coloring in linear (in Δ) time. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 111–120, 2009.

4 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.

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

6 Anton Bernshteyn. A fast distributed algorithm for $\Delta + 1$-edge-coloring. *Journal of Combinatorial Theory, Series B*, 152:319–352, 2022.

7 Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed Lovász local lemma. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 479–488, 2016.

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

9 Sebastian Brandt, Yannic Maus, and Jara Uitto. A sharp threshold phenomenon for the distributed complexity of the Lovász local lemma. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 389–398. ACM, 2019. doi:10.1145/3293611.3331636.

10 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 Transactions on Algorithms (TALG)*, 16(1):1–51, 2019.

11 Yi-Jun Chang, Wenzheng Li, and Seth Pettie. Distributed $(\Delta + 1)$-coloring via ultrafast graph shattering. *SIAM Journal on Computing*, 49(3):497–539, 2020.

12 Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the LOCAL model. *SIAM Journal on Computing*, 48(1):33–69, 2019.

13 Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed algorithms for the Lovász local lemma and graph coloring. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 134–143, 2014. doi:10.1145/2611462.2611465.

14 Devdatt P. Dubhashi, David A. Grable, and Alessandro Panconesi. Near-optimal, distributed edge colouring via the nibble method. *Theor. Comput. Sci.*, 203(2):225–251, 1998. doi:10.1016/S0304-3975(98)00022-X.

15 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 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 355–370, 2015. doi:10.1137/1.9781611973730.26.

16 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.

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

18 Manuela Fischer and Mohsen Ghaffari. Sublogarithmic distributed algorithms for Lovász local lemma, and the complexity hierarchy. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, volume 91 of *LIPIcs*, pages 18:1–18:16. LZI, 2017. doi:10.4230/LIPIcs.DISC.2017.18.

19 Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. Deterministic distributed edge-coloring via hypergraph maximal matching. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 180–191, 2017.

20 Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 270–277, 2016.

21 Mohsen Ghaffari. Distributed maximal independent set using small messages. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 805–820, 2019.

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

23 Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 662–673, 2018.

24 Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, and Yannic Maus. Improved distributed delta-coloring. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 427–436, 2018.

25 Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, Yannic Maus, Jukka Suomela, and Jara Uitto. Improved distributed degree splitting and edge coloring. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, pages 19:1–19:15, 2017.

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

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

28 Mohsen Ghaffari and Hsin-Hao Su. Distributed degree splitting, edge coloring, and orientations. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2505–2523, 2017.

29 Magnús M. Halldórsson and Alexandre Nolin. Superfast coloring in CONGEST via efficient color sampling. In *Proceedings of the International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 2021.

30 Magnús M. Halldórsson, Yannic Maus, and Alexandre Nolin. Fast distributed vertex splitting with applications, 2022. doi:10.48550/ARXIV.2208.08119.

31 David G. Harris. Distributed local approximation algorithms for maximum matching in graphs and hypergraphs. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 700–724, 2019.

32 Penny E. Haxell. A note on vertex list colouring. *Comb. Probab. Comput.*, 10(4):345–347, 2001.

33 Ken-ichi Kawarabayashi and Gregory Schwartzman. Adapting local sequential algorithms to the distributed setting. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, volume 121 of *LIPIcs*, pages 35:1–35:17. LZI, 2018.

34 Fabian Kuhn. Weak graph colorings: distributed algorithms and applications. In *Proceedings of the ACM Symposium on Parallelism in Algorithms and Architecture (SPAA)*, pages 138–144, 2009.

35 Nathan Linial. Distributive graph algorithms – global solutions from local data. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 331–335, 1987.

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

37 Yannic Maus. Distributed graph coloring made easy. In *Proceedings of the ACM Symposium on Parallelism in Algorithms and Architecture (SPAA)*, pages 362–372. ACM, 2021. doi:10.1145/3409964.3461804.

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

39 Yannic Maus and Jara Uitto. Efficient CONGEST algorithms for the Lovász local lemma. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, volume 209 of *LIPIcs*, pages 31:1–31:19. LZI, 2021. doi:10.4230/LIPIcs.DISC.2021.31.

40 Michael Molloy and Bruce Reed. Further algorithmic aspects of the local lemma. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 524–529, 1998.

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

42 Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997. doi:10.1137/S0097539793250767.

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

44 Bruce Reed. The list colouring constants. *Journal of Graph Theory*, 31(2):149–153, 1999. doi:10.1002/(SICI)1097-0118(199906)31:2<149::AID-JGT8>3.0.CO;2-\%23.

45 Bruce Reed and Benny Sudakov. Asymptotically the list colouring constants are 1. *Journal of Combinatorial Theory, Series B*, 86(1):27–37, 2002.

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

47 Hsin-Hao Su and Hoa T. Vu. Towards the locality of Vizing's theorem. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 355–364, 2019.

A Edge coloring in CONGEST (similar to Section 6)

We begin with proving a CONGEST counterpart of Theorem 25 for very low degree graphs.

▶ **Theorem 26.** *For any constant $\varepsilon > 0$, there is an absolute constant Δ_0 such that there is a randomized* CONGEST *algorithm with runtime* $\operatorname{poly}\log\log n$ *for $(1+\varepsilon)\Delta$-edge coloring any n-node graph with maximum degree $\Delta_0 \leq \Delta \leq \operatorname{poly}\log\log n$.*

Proof. Recall from the text after Theorem 25 that the edge-coloring problem can be solved via a constant number of LLL instances that are defined on a dependency graph H such that one round of communication can be simulated in $O(1)$ rounds in LOCAL in the original network [10]. The dependency degree of H is $d = \operatorname{poly}\Delta$. Hence, if $\Delta \leq \operatorname{poly}\log\log n$, one round of communication in H can be simulated in $\operatorname{poly}\log\log n$ CONGEST rounds in the communication network. The result follows via Corollary 22. The base case for the algorithm of [10] is a 5Δ-edge coloring step, which can also be solved in $\operatorname{poly}\log\log n$ CONGEST rounds [29]. ◀

Proof of Theorem 1, CONGEST**.** We use the same high level algorithm as in the LOCAL model, that is, we first split into subgraphs of $\Delta' = \operatorname{poly}\log n$ maximum degree, then in to subgraphs of $\Delta'' = \operatorname{poly}\log\log n$ degree, which we then color with disjoint color spaces, with $(1+\varepsilon/6)\Delta''$ colors each. We refer to the LOCAL version for further the details on this reduction. Here, we only explain which parts differ in the CONGEST model. First, each edge is simulated by one of its endpoints. The reduction to $\operatorname{poly}\log n$ degrees works in zero rounds, just as in the LOCAL model.

The most challenging part is the reduction from $\operatorname{poly}\log n$ degrees to $\operatorname{poly}\log\log n$ degrees. The pre-shattering phases (in computing the q-divide and in the main algorithm) immediately work in the CONGEST model. We only need to reason that the post-shattering phases (of

q-divide and the q instances in the main algorithm) can be solved via Lemma 23. To this end, we need to run $\ell = O(\log n)$ instances of [13] in parallel with bandwidth $= \Theta(\log n)$. Observe that $k \leq \operatorname{poly}\log n$ holds. For each node to be able to evaluate the status of its bad events in all ℓ instances in parallel, we have the node simulating each edge send to the other endpoint the parts it chose in all ℓ instances (one part per instance). As the part index of an edge can be encoded with $O(\log k)$ bits, the indices in the ℓ instances can be communicated with $\ell \cdot O(\log k) = O(\log n \log\log n)$ bits. With the available bandwidth this requires only $O(\log\log n)$ rounds. The other messages needed by the algorithm to simulate the ℓ instances of [13] in parallel, such as whether the edge needs be re-sampled in each instance, similarly never exceed $O(\log\log n)$ rounds.

Once the degrees of the subgraphs are at most $\operatorname{poly}\log\log n$ we use Theorem 26 to color the graphs. Formally, the color space is still large. To really use Theorem 26, vertices color each subgraph with colors in the range from 1 to $\operatorname{poly}\log\log n$, and map their color back to the original color space at the end of the computation. ◀

B Application: List Coloring

▶ **Definition 27.** *In an* (L,T)-list-coloring *instance on a graph* $G=(V,E)$*, each node* $v \in V$ *is given a list* $L(v)$ *of colors of at least* L *such that for each* $c \in L(v)$ *there are at most* T *neighbors* u *of* v *with* $c \in L(u)$*. The parameter* T *is referred to as the* color degree*. Similarly for* $c \in L(v)$*,* $|\{u \in N(v) \mid c \in L(u)\}|$ *is the* color degree *of color* c *for* v*.*

Note that one cannot generally solve such the problem via a greedy approach, not even centrally. Still, the objective is to find solutions for arbitrary L and T with a ratio L/T as small as possible. Reed [44] gave a simple LLL argument for the existence of a solution when $L/T \geq \lceil 2e \rceil$. This was improved to $L/T = 2$ by Haxell [32]. Reed and Sudakov [45] then showed that $L/T = 1 + o(1)$ suffices. Reed's famous list coloring conjecture states that $L = T + 2$ colors always suffice [44]. We recall Reed's argument for the existence if $L/T > 2e$. In the distributed setting, there is an $O(\log n)$ round algorithm for $L/T \geq (1+\delta)$ [13], and a $O(\operatorname{poly}\Delta + \operatorname{poly}\log\log n)$ rounds for $L/T \geq C_0$ for a sufficiently large constant C_0 [18].

LLL formulation (for existence only). Suppose each node picks a color from its list uniformly at random. Define a bad event $\mathcal{B}_{u,v,c}$ for each edge $\{u,v\} \in E$ and each color c if both u and v choose the color c. The probability for such an event is at most $p = 1/|L(v)| \cdot 1/|L(u)| \leq 1/L^2$. The dependency degree of these events is $d = 2L \cdot T$, because it can depend on at most L colors for each of the endpoints of the edge and on T other incident edges for each of these colors. Thus, we obtain the LLL criterion $p \cdot (2L \cdot T) = 2T/L$, and hence for $L/T > 2e$, a the standard criterion $epd < 1$ is satisfied and a solution exists.

Distributed results. Reed's argument leads to an $O(\log^2 n)$-round LOCAL algorithm for any $L/T > 2e$ with the classic Moser-Tardos algorithm [41]. Chang, Pettie and Su [13] gave a algorithm for $L/T = 1+\delta$, with a quite involved analysis, that runs in time $O(\log^* L \max(1, \log n)/D^{(1-\gamma)})$. Fischer and Ghaffari [18] showed using *color pruning* that there exists some (possibly large) constant C to solve (L,T)-list coloring whenever $L/T \geq C$ in $\operatorname{poly}(\Delta, \log\log n)$ rounds.

Our main theorem combines the effectiveness of [13] with the speed of [18].

▶ **Theorem 2** (List coloring)**.** *There is a* $\operatorname{poly}\log\log n$*-round randomized* LOCAL *algorithm for the list coloring problem, for any* T *and* L *with* $L \geq (1+\delta)T$*, for any* $\delta > 0$ *and any* $\Delta \geq \Delta_0$*, for some absolute constant* Δ_0*.*

Similarly to the edge-coloring problem our high level idea is to first reduce the size of the relevant parameters to poly log log n, after which we can solve arbitrary LLLs efficiently on the problem. However, the reduction and the base case (once parameters are of size poly log log n) are significantly more involved than in the edge-coloring problem. We summarize the main technical lemma showing that we can efficiently reduce the parameters L and T while keeping the ratio of list size and color degree almost the same.

▶ **Lemma 28** (List color sparsification). *There exists a universal constant $c_6 > 0$ s.t.: For any $\varepsilon > 1/\operatorname{poly}\log\log n$ and $k \leq c_6 \cdot (\varepsilon^4 L/\log L)$, there is a $\operatorname{poly}(\varepsilon^{-1}, \log\log n)$-round algorithm for the following list coloring sparsification problem: Given a (L,T)-list coloring instance with $T < L \leq \operatorname{poly}\log n$ on an n-node graph $G = (V,E)$ the goal is to compute a sublist $L'(v) \subseteq L(v)$ for each node yielding a (L',T')-list coloring instance on the same graph with*

$$L' = L/k \pm \varepsilon L/k, T' \leq T/k + \varepsilon T/k \text{ and } L'/T' \geq (1-\varepsilon)L/T. \tag{1}$$

We obtain the same properties in zero rounds if $L > \operatorname{poly}\log n$ and $k \leq \varepsilon^2\Delta/(9\ln n)$ holds for $\Delta = L \cdot T$.

C Missing Proofs

C.1 Vertex Splitting: Bounding the Discrepancy

In this section, we bound the deviation in the number of neighbors that a node v sees in the i-th part from $d(v)/k$. For a node v let $N_{\mathsf{pre}}(v)$ ($N_{\mathsf{post}}(v)$) be the neighbors of v that are permanently assigned to a part in the pre-shattering (post-shattering) phase. Also, let $d_{\mathsf{pre}}(v) = |N_{\mathsf{pre}}(v)|$ and $d_{\mathsf{post}}(v) = |N_{\mathsf{post}}(v)|$. Recall, the definition of $z_j^{\mathsf{pre}}(v) = z_j^{\mathsf{post}}(v) = \varepsilon^2/(72k)$ and $q = 24/\varepsilon$, which immediately yields the following claim.

▷ Claim 29. We have $\sum_{j\in[q]} z_j^{\mathsf{pre}}(v) = \sum_{j\in[q]} z_j^{\mathsf{post}}(v) \leq \varepsilon/3 \cdot \Delta/k$.

▶ **Lemma 16** (restated with details). *In the final assignment $V_1, \dots, V_k$, i.e., after the pre-shattering and post-shattering phase, we have the following guarantees on the split for each part $i \in [k]$:*

1. *Node v has $d_{\mathsf{pre}}(v)/k \pm 2\varepsilon/3 \cdot \Delta/k$ neighbors in $V_i \cap N_{\mathsf{pre}}(v)$.*
2. *Node v has $d_{\mathsf{post}}(v)/k \pm \varepsilon/3 \cdot \Delta/k$ neighbors in $V_i \cap N_{\mathsf{post}}(v)$.*

In total, for each $i \in [k]$ any node v has $d(v)/k \pm \varepsilon\Delta/k$ neighbors in V_i.

Proof. We first prove the first claim. The discrepancy (deviation from expectation) for a node v comes from two sources: (a) slots with neighbors that got retracted; (b) other slots. We bound both separately. Consider a vertex v and fix a part $V_i, i \in [k]$. For the rest of the proof let $z_j = z_j^{\mathsf{pre}}(v)$. We partition the vertices in $V_i \cap N_{\mathsf{pre}}(v)$ according to the q slots $N_1(v), \dots N_q(v)$. Due to Observation 15 for at most one j does $N_j(v)$ contain nodes whose values were retracted. Denote this j (if any) by j^*, otherwise set $j^* = \perp$.

▷ Claim 30. If $j^* \neq \perp$, then $|V_i \cap N_{j^*}(v) \cap N_{\mathsf{pre}}(v)| \leq \hat{d}_{j^*}(v)/k + z_j$.

Proof. If v caused the retraction then, $N_{j^*}(v) \cap N_{\mathsf{pre}}(v) = \emptyset$, as v retracted all assignments of nodes in $N_{j^*}(v)$ and froze the nodes (they will only be assigned in the post-shattering phase). Now consider the case that v did not cause the retraction, i.e., $\mathcal{B}_j^{\mathsf{pre}}$ does not occur, and let X_i be the nodes in part i in the temporal assignment of nodes in slot j before any retractions happened (also before the ones caused by nodes $u \neq v$). Since $\mathcal{B}_j^{\mathsf{pre}}$ does not occur, we have $|X_i \cap N_{j^*}| \in \hat{d}_{j^*}/k \pm z_j$. Some nodes of X_i might get retracted by other nodes $u \neq v$, but we obtain $|V_i \cap N_{j^*}(v) \cap N_{\mathsf{pre}}(v)| \leq |X_i \cap N_{j^*}(v)| \leq \hat{d}_{j^*}(v)/k + z_j$. ◁

▷ Claim 31. For each $j \notin [q] \setminus j^*$, we obtain $|V_i \cap N_{\mathsf{pre}}(v) \cap N_j(v)| = \hat{d}_j(v)/k \pm z_j$.

Proof. Since $j \neq j^*$ there are no retracted variables in $N_j(v)$. If the bound in the claim does not hold, then $\mathcal{B}_j^{\mathsf{pre}}(v)$ would have occurred after the sampling, and v would have retracted, a contradiction. ◁

▷ Claim 32. $\sum_{j\in[q],j\neq j^*} \hat{d}_j(v) \leq d^{\mathsf{pre}}(v) \leq \sum_{j\in[q]} \hat{d}_j(v)$.

In the following we omit the explicit dependence on v, e.g., we write $\hat{d}_j$ instead of $\hat{d}_j(v)$. Using, $\sum_{j\in[q]} z_j(v) \leq \varepsilon\Delta/(3k)$ (Claim 29), $\hat{d}_{j*} \leq d_{j*} \leq 8\Delta/q = \varepsilon\Delta/3$ (from the properties of a q-divide), bounds on $|V_i \cap N_{\mathsf{pre}}(v) \cap N_j(v)|$ (Claims 30 and 31), and Claim 32 we obtain

$$|V_i \cap N_{\mathsf{pre}}(v)| \leq \sum_{j\in[q]} \left(\hat{d}_j/k + z_j\right) \leq (d^{\mathsf{pre}} + \hat{d}_{j^*})/k + \sum_{j\in[q]} z_j \leq d^{\mathsf{pre}}/k + 2\varepsilon\Delta/(3k) \quad \text{and}$$
$$|V_i \cap N_{\mathsf{pre}}(v)| \geq \sum_{j\in[q],j\neq j^*} \left(\hat{d}_j/k - z_j\right) \geq \left(d^{\mathsf{pre}} - \hat{d}_{j^*}\right)/k - \varepsilon\Delta/(3k) \geq d^{\mathsf{pre}}/k - 2\varepsilon\Delta/(3k)\ .$$

For the second part of the claim, fix again some $i \in [k]$ and a node v. There are q separate post-shattering instances. Recall, the set of neighbors of a node participating in the j-th instance is denoted by $F_j(v)$ and $f_j(v) = |F_j(v)|$. The solution to the LLL instance yields

$$|V_i \cap F_j(v) \cap N_{\mathsf{post}}(v)| = f_j(v)/k \pm z_j^{\mathsf{post}}(v). \tag{2}$$

Summing over all q post-shattering instances, using $d_{\mathsf{post}}(v) = \sum_{j\in[q]} f_j(v)$ and using Claim 29 to bound $\sum_{j\in[q]} z_j^{\mathsf{post}}(v) \leq \varepsilon\Delta/3 \leq \varepsilon\Delta/2$ yields the second part of the claim. ◀

C.2 Analysis of FastShattering

▶ **Lemma 18.** *After FastShattering, each connected component in each of the q post-shattering instances is of size $\Delta^{10}\log n$, w.h.p.*

Proof of Lemma 18. Let us focus on one post-shattering instance, instance number j, formed of both the nodes in Bad_j that were frozen while processing slot j and all their incident 'events nodes'. Let us say that a node v *triggers* if one of the events $\mathcal{B}_{i,j}^{\mathsf{pre}}$, $i \in [k]$ occurs, i.e., if $D_{i,j}(v)$ deviates too much from expectation. That a node triggers is entirely determined by the random choices of its neighbors. By Claim 17 (applied with $N = 8\Delta/q$, $z = z_j^{\mathsf{pre}}(v) = \varepsilon^2\Delta/(72k)$, and $D_{i,j}(v)$), the probability that a node triggers is at most Δ^{-24}. A variable is `frozen` if it is within distance 3 of a triggering node. A node v joins the post-shattering instance if it is `frozen` itself or one of its neighbors is `frozen`, which depends on whether nodes within distance 4 of v trigger or not, which itself is entirely determined by the random choices within distance 5 of v. Thus, whether two nodes at distance 10 participate in the j-th post-shattering instance depends on two sets of non-overlapping random variables from the processing of slot j.

By a union bound over the Δ^4 nodes in the 4-hop neighborhood and the k parts, a node participates in the j-th post-shattering instance w.p. at most $k(\Delta^4)\Delta^{-24} \leq \Delta^{-19}$. By Lemma 6, the resulting connected components of the post-shattering instance are all of size $O(\Delta^{10}\log_\Delta n)$, w.h.p. ◀

D Bipartite Vertex Splitting and Beyond

Another classic version is to split only one side of a bipartite graph. Given a bipartite graph $(V^{\mathsf{L}} \cup V^{\mathsf{R}}, E)$ and an parameter k the objective is to split the *variable vertices* V^{R} into k parts $V_1^{\mathsf{R}}, \dots, V_k^{\mathsf{R}}$ such that the degree of every *event vertex* $u \in V^{\mathsf{L}}$ into each part V_i^{R} does not deviate from $d(u)/k$ by too much. More formally, each event node $u \in V^{\mathsf{L}}$ comes with a parameter $z(u)$ that bounds the deviation. Let Δ_{L} and Δ_{R} be the maximum degree of nodes in V^{L} and V^{R} respectively. With the same analysis as for Theorem 3 (reasons below) we obtain the following theorem for bipartite vertex splitting.

▶ **Theorem 4.** *There exists a universal constant $c_7 > 0$ s.t.: For any $\varepsilon > 0$, maximum degree $\Delta \leq \operatorname{poly}\log n$ and $k \leq c_7 \cdot (\varepsilon^4 \Delta_{\mathsf{L}} / \ln \Delta)$, there is a distributed randomized* LOCAL *algorithm to compute a bipartite k-vertex splitting problem with discrepancy $\varepsilon \Delta_{\mathsf{L}}/k$ in $O(1/\varepsilon) + \operatorname{poly}\log\log n$ rounds.*

The simpler q-divide problem also naturally extends to this more general setup. The objective of a bipartite q-divide is to partition the variable vertices into q parts such that each event node has at most $8\Delta_{\mathsf{L}}/q$ neighbors in each part.

▶ **Theorem 33.** *For any $q \in [1, (1/6)\Delta_{\mathsf{L}}/\ln \Delta]$, there is a* LOCAL *algorithm to compute a bipartite q-divide in $\operatorname{poly}\log\log n$-rounds.*

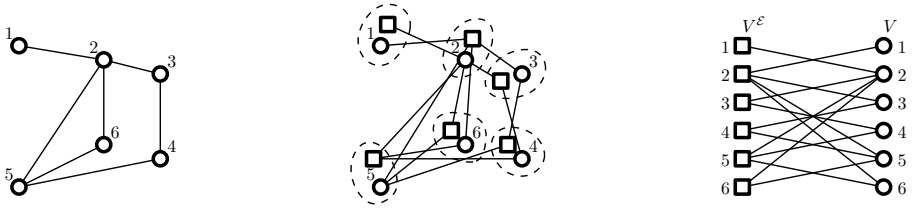

Figure 2 A graph and the bipartite splitting instance obtained from the vertex-splitting problem on it by turning each node into a variable node (circles) and an event node (squares).

To better understand this more general setting and how our results for q-divide and k-split extend to it, let us translate those problems into their bipartite versions. For a graph $G = (V, E)$, we construct a bipartite graph $G' = (V^{\mathsf{L}} \cup V^{\mathsf{R}}, E')$ such that a bipartite k-split (bipartite q-divide) on G' maps to a k-split (q-divide) on G. Let $n = |V|$ be the number of nodes of G and Δ its maximum degree. In bipartite terminology, when computing a k-split on G each node of G is acting both as an event node and a variable node, as we must ensure the proper splitting of its neighborhood as well as assigning it. The bipartite graph corresponding to this problem is the graph $G' = (V^{\mathsf{L}} \cup V^{\mathsf{R}}, E')$ where $|V^{\mathsf{L}}| = |V^{\mathsf{R}}| = n$, $\forall i, j \in |V|^2, v_i v_j \in E \Leftrightarrow \left(v_i^{\mathsf{L}} v_j^{\mathsf{R}} \in E' \wedge v_i^{\mathsf{R}} v_j^{\mathsf{L}} \in E'\right)$. G' has $2n$ nodes and maximum left and right degree $\Delta_{\mathsf{L}} = \Delta_{\mathsf{R}} = \Delta$. See Figure 2 for an illustration of the translation process.

Proof of Theorems 4 and 33. Our proofs of Theorems 3 and 10 naturally extend to the bipartite setting. Intuitively, the algorithms follow the same pattern. We have a simple random assignment procedure that we show properly partitions the neighborhood of a node w.p. $1 - \operatorname{poly}(\Delta)$. In addition, there is a way of running this procedure, retracting some assignments and avoiding to assign some nodes that ensures that only small patches of the graph remain unassigned and the partial assignment that is obtained can be completed to a full assignment. All that we need to show is that our setting of parameters in the bipartite setting are correct, i.e., control the amount of discrepancy as in the previous setting.

Let Δ_{L} be the degree of the left hand side vertices in $V^{\mathcal{E}}$ and let Δ_{R} be the degree of the right hand side vertices in V. Let $\Delta = \max\{\Delta_{\mathsf{L}}, \Delta_{\mathsf{R}}\}$.

All probabilities are exponential in $-\Theta(z)$, or in $\Theta(-z \cdot (z/(d(u)/k)))$ if the degree of a node u is larger than $k \cdot z$.

The union bound in the shattering over distance 5-neighborhoods introduces a multiplicative Δ^5 term. Hence, we require that $e^{\Theta(z)}$ and $e^{\Theta(z)\cdot(z/(d(u)/k))}$ dominate the Δ^5 term. This, clearly holds if $z = \varepsilon^2\Delta/(72k)$ as before, given the assumed upper bound on k.

The proof of the discrepancy (in Section 4.2) remains exactly the same; just note that in the bipartite vertex splitting the discrepancy values ($z(v)$s) depend on Δ_{L} instead of Δ, and hence we obtain a deviation from $d(u)/k$ that is upper bounded by $\varepsilon\Delta_{\mathsf{L}}/k$. ◀

▶ Remark 34. In general, it is not possible to recursively use vertex-splitting to split into smaller and smaller parts. Special properties of an instance (as we have with edge-splitting and when solving list-coloring here) sometime still make it possible.

See the full information for more details on Remark 34