Distributed Symmetry Breaking on Power Graphs via Sparsification*

Yannic Maus
TU Graz
Graz, Austria
yannic.maus@ist.tugraz.at

Saku Peltonen
Aalto University
Espoo, Finland
saku.peltonen@gmail.com

Jara Uitto
Aalto University
Espoo, Finland
jara.uitto@aalto.fi

ABSTRACT

In this paper we present efficient distributed algorithms for classical symmetry breaking problems, maximal independent sets (MIS) and ruling sets, in power graphs. We work in the standard CONGEST model of distributed message passing, where the communication network is abstracted as a graph G. Typically, the problem instance in CONGEST is identical to the communication network G, that is, we perform the symmetry breaking in G. In this work, we consider a setting where the problem instance corresponds to a power graph G^k, where each node of the communication network G is connected to all of its k-hop neighbors.

A β-ruling set is a set of non-adjacent nodes such that each node in G has a ruling neighbor within β hops; a natural generalization of an MIS. On top of being a natural family of problems, ruling sets (in power graphs) are well-motivated through their applications in the powerful *shattering* framework [BEPS JACM'16, Ghaffari SODA'19] (and others). We present randomized algorithms for computing maximal independent sets and ruling sets of G^k in essentially the same time as they can be computed in G. Our main contribution is a deterministic poly$(k, \log n)$ time algorithm for computing k-ruling sets of G^k, which (for $k > 1$) improves *exponentially* on the current state-of-the-art runtimes. Our main technical ingredient for this result is a deterministic sparsification procedure which may be of independent interest.

We also revisit the shattering algorithm for MIS [BEPS J'ACM'16] and present different approaches for the post-shattering phase. Our solutions are algorithmically and analytically simpler (also in the LOCAL model) than existing solutions and obtain the same runtime as [Ghaffari SODA'16].

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**; *Sparsification and spanners*; *Pseudorandomness and derandomization.*

KEYWORDS

distributed algorithm, CONGEST model, maximal independent set, ruling sets, power graphs, sparsification, shattering

*The full version of the paper is available at [54].

PODC '23, June 19–23, 2023, Orlando, FL, USA

ACM ISBN 979-8-4007-0121-4/23/06.
https://doi.org/10.1145/3583668.3594579

ACM Reference Format:
Yannic Maus, Saku Peltonen, and Jara Uitto. 2023. Distributed Symmetry Breaking on Power Graphs via Sparsification. In *ACM Symposium on Principles of Distributed Computing (PODC '23), June 19–23, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3583668.3594579

1 INTRODUCTION

In this paper we provide efficient deterministic and randomized algorithms for symmetry breaking problems on power graphs, that is, we compute maximal independent sets and ruling sets on the power graph G^k for an integer k, where G is the input graph. To illustrate the setting, let us define the central problem of our work. A *maximal independent set (MIS)* S of a graph G is a set of non-adjacent nodes such that every node v of G is *dominated* by a node in S, that is, there is a node in S with distance at most 1 from v. *Ruling sets* generalize this notion by relaxing the distance of domination. In the power graph G^k, any two nodes nodes of G are connected if they are at most k hops apart. Hence, in an MIS of G^k any two nodes have distance at least $k+1$ and every node of G is dominated by a node of S with distance at most k. MIS and ruling sets have been studied extensively in the classic message passing models of distributed computing, i.e., the LOCAL and the CONGEST model, e.g., see [4, 6, 8, 12, 31, 33, 50, 53, 59].

Understanding symmetry breaking on power graphs is crucial as they appear naturally in various settings. A classic example is given by the frequency assignment problem. In order to avoid interference in a network of wireless transmitters, one wants to assign frequencies to the nodes of a communication network such that all neighbors of each node receive different frequencies. The problem is a vertex coloring problem on the power graph G^2 [41, 42, 49]. Problems on power graphs also appear as subroutines when solving problems for G. One example is the state-of-the-art randomized algorithm to compute an MIS (of G) that relies on the computation of ruling sets of power graphs, both in the LOCAL model [12, 31] and in the CONGEST model [33]. A second example is the current state-of-the-art deterministic algorithms for computing MIS [25]. A third example where such ruling sets appear as subroutines is the algorithm to compute spanners in [24].

In Section 1.2, we provide further examples and further motivate (symmetry breaking) problems on power graphs. The main model of our work is the CONGEST model of distributed computing.

The challenges of working with power graphs in the CONGEST *model.* In the LOCAL and the CONGEST model, a communication network is abstracted as an n-node graph G with nodes representing computing entities and edges communication links [52, 56]. Nodes are equipped with $O(\log n)$-bit IDs. In order to solve some problem in the network, the nodes communicate with each other in

synchronous rounds. In each round, nodes are allowed to perform arbitrary local computations and send messages to each of their neighbors in G. In the LOCAL model messages can be of *unbounded* size, while in the CONGEST model message sizes are restricted to $O(\log n)$ bits. The *time complexity* of an algorithm is the number of rounds until each node has computed its own part of the solution, e.g., whether it is contained in an independent set or not. Classically in the literature, an algorithm for a problem—think of the MIS problem— assumes that the communication network is also the problem instance. In contrast, in our work the power graph G^k serves as the problem instance while G remains the communication network. In the LOCAL model, the setting does not yield any major difficulties as an algorithm designed for G (e.g., to compute an MIS of G), can be run on G^k with a multiplicative overhead of k rounds (to compute an MIS of G^k). However, such a statement is not true in the CONGEST model. In fact, in the CONGEST model, a node does not even know its degree in the problem instance G^k and if many vertices want to send different messages to their neighbors in G^k, congestion appears in the communication network and the messages cannot be delivered efficiently. This seemingly small difference, has huge effects and makes it much more challenging to construct algorithms.

1.1 Our Contributions

An *r-ruling set* S of a graph G is a set of non-adjacent nodes such that for every node v of G there is a node in S with distance at most r from v. Hence, in an r-ruling set of G^k any two nodes have distance at least $k+1$ and the domination distance is $r \cdot k$. In the literature this often appears as an (α, β)-ruling set where α specifies the minimum distance between nodes and β the domination distance. As ruling sets relax the domination guarantee of maximal independent sets, they are usually easier to compute. Often, ruling sets are sufficiently powerful to replace MIS computations as subroutines in algorithmic applications. Our main contribution is the first efficient deterministic algorithm to compute k-ruling sets for G^k. Throughout the paper $\widetilde{O}(x)$ omits factors that are logarithmic in x and Δ refers to the maximum degree of the graph G, even when we are solving a problem on the power graph G^k.

Theorem 1.1 (k-ruling set of G^k). *Let $k \geq 1$ be an integer (potentially a function of n). There is a deterministic distributed algorithm that computes a k-ruling set of G^k in $\mathrm{poly}(k, \log n)$ time in the* CONGEST *model. More detailed, the round complexity is $\widetilde{O}(k^2 \cdot \log^4 n \cdot \log \Delta)$ rounds.*

Theorem 1.1 computes a $(k+1, k^2)$-ruling set of G. For a constant $k > 1$, this improves *exponentially* on the previous state of the art that required $O(n^{1/k})$ rounds [24]. This previous algorithm is an extension to G^k of a classic deterministic algorithm for computing $O(\log_B n)$-ruling sets (of G) in $O(B \cdot \log_B n)$ rounds [6, 24, 45, 50, 59], where B is a parameter that can trade domination for runtime.

Our main technical contribution in order to obtain Theorem 1.1 is a novel sparsification procedure. On a very high level, we turn G^k into a much sparser representation $\bar{G}$ such that an MIS of $\bar{G}$ is a good ruling set on the original G^k. The benefit is that we can communicate more efficiently on $\bar{G}$.

Randomized symmetry breaking in power graphs. Our second result is an algorithm for MIS of G^k with (essentially) the same runtime as the state of the art for computing an MIS of G.

Theorem 1.2. *There is a randomized distributed algorithm that computes a maximal independent set of G^k in $\widetilde{O}(k^2 \log \Delta \cdot \log \log n + k^4 \log^5 \log n)$ rounds of the* CONGEST *model, with high probability.*

The best result that can be achieved with previous work is an $O(k \log n)$-round algorithm based on Luby's algorithm for computing an MIS (see [54, Section 8.1] for details) [4, 53]. Theorem 1.2 compares favorably to the state of art for computing an MIS of G in $O(\log \Delta \log \log n + \mathrm{poly} \log \log n)$ rounds in the CONGEST model [33]. The randomized complexity of MIS on G^k in LOCAL is $O(k^2 \log \Delta + k \,\mathrm{poly} \log \log n)$, simply by running the best known algorithm for MIS on G on the power graph [31]. By combining Theorem 1.2 with the sparsification methods of [48] and [13] we obtain the following corollary.

Corollary 1.3. *There is a randomized distributed algorithm that computes a β-ruling set of G^k in $\widetilde{O}(\beta \cdot k^{1+1/(\beta-1)} (\log \Delta)^{1/(\beta-1)} + \beta \cdot k \log \log n + k^4 \log^5 \log n)$ rounds.*

The only previous randomized algorithm for computing a ruling set of G^k is a $O(k \log \log n)$-ruling set algorithm that works in $O(k^2 \log \log n)$ rounds [33]. The technique used in [33] can inherently not compute $O(1)$-ruling sets of G^k.

Simplifying shattering for MIS. Ruling sets of power graphs are essential in the the current state-of-the-art randomized algorithms to compute an MIS in LOCAL [12, 31] and the CONGEST model [33]. In our last contribution, we revisit[1] the *current* state-of-the-art algorithms for computing an MIS in G in LOCAL [12, 31] and CONGEST [33]. These results are based on the by-now-standard *shattering technique* that uses a random process to solve the respective problem on most of the graph such that the remaining unsolved parts only consists of (many) *small connected components*. The components are small in the sense that a certain power graph ruling set of a single component has at most logarithmically many nodes. This fact is exploited to solve the remaining problem on these small components efficiently. The challenge here is that to algorithmically exploit the fact that these ruling sets are of a small size, one also has to compute them. Intuitively, one wants to just run a classic $(5, O(\log \log n))$-ruling set algorithm on each of the remaining connected components in parallel. The crux is that the distance between ruling set nodes needs to be measured in G (for the ruling set to be small) and not just within the small component. But a node cannot tell efficiently whether a node in distance 5 in G is in the same small component as itself or not. Hence, the nodes cannot easily determine whether they can both be contained in the ruling set or not and one cannot readily treat different small components independently.

Thus, the arXiv version of [12] presents an involved two-phase shattering procedure with an involved analysis. It turns out that the presented analysis of bounding the size of the ruling set has a small (but crucial) mistake (see Sections 3 and [54, Section 7] for details).

[1] We revisited the details of these algorithms as they rely on the shattering framework that we also use to obtain Theorem 1.2. However, the setting in Theorem 1.2 is more involved.

The journal version of the respective work presents a fix through an even more involved analysis [12]. The later works [31, 33] build upon the arXiv version of [12]. While we believe that both works can be adapted to build upon the arguments in the journal version of [12], this cannot be done in a black-box manner. Both works use a different ruling set algorithm than [12] and the internals of the ruling set algorithm are crucial for the arguments in [12].

As our last contribution, we revisit the shattering framework and present two different algorithmic and analytical solutions for computing an MIS that work in the LOCAL and the CONGEST model. While these do not improve upon the complexity of [12, 31, 33], our approach is simpler and provides a fix for results that build upon the arXiv version of [12].

Theorem 1.4. *There are randomized* LOCAL *and* CONGEST *algorithms that w.h.p. compute a maximal independent set on any n-node graph with maximum degree Δ and that run in $O(\log \Delta) + \text{poly} \log \log n$ rounds and in $O(\log \Delta \log \log n) + \text{poly} \log \log n$ rounds, respectively.*

Even though Theorem 1.4 deals with the classic setting of computing an MIS of the graph G (and not its power graph), the flaw in the current argument is closely related to properly dealing with ruling sets in power graphs. Some other works in the literature build upon the ideas of the arXiv version of [12], e.g., algorithms for Δ-coloring [37] or the Lovász Local Lemma [26], and thus undergo a similar issue. However, we note that the aforementioned power graph ruling sets are only relevant when $\Delta \geq \text{poly} \log n$ as otherwise one can use simpler approaches to exploit the smallness of the components. But for $\Delta \geq \text{poly} \log n$ it is not necessary to correct the flaw in the two papers. For the Δ-coloring problem a very recent paper provided a genuinely different algorithm to solve the problem for $\Delta \geq \text{poly} \log n$, making a fix unnecessary [27]. For the Lovász Local Lemma problem, one can use the algorithm by Chung, Pettie and Su to solve the problem in $O(\log n)$ rounds [18], which is faster than the $O(\Delta^2) + \text{poly} \log \log n$ rounds of [26] when $\Delta \geq \text{poly} \log n$. Last but not least, let us remark that the state-of-the-art randomized $(\Delta + 1)$-coloring algorithms are not impacted by the flaw as they only rely on shattering when $\Delta \leq \text{poly} \log n$ [16, 43]. For more details on the background of Theorem 1.4 and our solutions, see Section 3 and [54, Section 7]. Note that Theorem 1.4 is also central to countless results in the literature as the algorithm belongs to the most used subroutines in the area. Also Theorem 1.2 and Corollary 1.3 rely on extensions of the algorithmic ideas used for Theorem 1.4.

1.2 Why Should We Care About Problems on Power Graphs?

We have already mentioned that ruling sets of power graphs are important subroutines. In this section, we provide various further reasons for studying CONGEST algorithms on power graphs.

Derandomization and learning distant information. The main challenge when working with power graphs is that even though we aim at solving problems on G^k, the communication network is just G itself and a node cannot immediately communicate with its neighbors in G^k. For an example, consider the simple task of learning some small individual piece information from each of your neighbors. In G, this problem is trivial and can clearly be achieved in a single CONGEST round. However, as soon as we turn to G^2, we suffer a huge overhead in the number of rounds. Solving this task requires $\Omega(\Delta)$ rounds in the worst case, where Δ is the maximum degree of the graph which may be very large.

But learning large amounts of information that is not stored at immediate neighbors of G is an important ingredient of recent efficient LOCAL algorithms. Prime examples of this behavior are the general derandomization results [36, 39, 57] in the LOCAL model. Note that besides the aspect of learning large amounts of information, another crucial ingredient to these results is the computation of so called network decompositions of the power graph G^k, usually for a non-constant k. It is a major open problem of the area to determine for which type of problems such derandomizations can be obtained in the CONGEST model.

On the positive side, for selected problems on G, there are efficient CONGEST algorithms that are all based on limiting the amount of information that one has to aggregate for an efficient derandomization of a (simple) randomized procedure. Examples are given by algorithms for $(\Delta + 1)$-coloring [10, 38], MIS [15, 25], or minimum dominating set approximations [21, 25]. Our sparsification results are also based on derandomization (in a more extreme setting) and hence add to the class of problems that can be efficiently derandomized in the CONGEST model. In Section 3 we detail on how we circumvent the necessity to learn large amounts of information used by the sparsification process.

Power graph and virtual graph problems as subroutines. Another prevalent ingredient to many recent results is that they solve intermediate problems on virtual graphs. For example, in the state of the art deterministic algorithm for MIS in LOCAL and CONGEST, we must simulate an algorithm (for some intermediate problem) on G^2 [25]. Hence, despite the fact that we consider the setting where the communication network equals the input graph, intermediate steps require solving problems on power graphs. In LOCAL, the unbounded message sizes allow us to communicate in G^2 with a constant overhead and hence, algorithms for the intermediate steps in the virtual graph G^2 are straight-forward to implement with a constant overhead in the runtime. In contrast, handling these intermediate steps is much more involved in the CONGEST model. In general, we believe that the study of power graph problems contributes to the general important theme of detaching the input graph from the communication network. Problems on power graphs serve as a clean abstraction to develop tools that can be used whenever the problem instance and the communication network are not exactly the same.

Robustness of algorithms and techniques. More broadly, studying problems on power graphs also serves as a clean abstraction to develop algorithmic techniques that are *robust* in the sense that they can work in settings with stronger communication restrictions. In the long run we also expect this kind of research to lead to algorithms that are more model independent. For example, many recent MPC algorithms are fast implementations of communication efficient LOCAL or CONGEST algorithms [7, 17, 19, 20, 40]. An extreme case of communication efficiency are algorithms in the

beeping model which is similar to the CONGEST model with 1-bit messages. An excellent example is the pre-shattering phase of Ghaffari's MIS algorithm that when computing an MIS for G works in the beeping model [32]. Due to its robust design, it has been used (with slight adaptations) to obtain the state-of-the-art in other settings, such as MPC and LCA [34, 40]. Theorem 1.2 is also based on an extension of this result to G^k that works in a stronger version of the beeping model, allowing $O(\log\log n)$-sized messages. See [54, Section 8] for details.

Another aspect that makes algorithms for power graphs more robust is that they have to operate in the setting where nodes do not know their degree (in G^k). Even though there are ways to remove the necessity of knowing your degree from an algorithm [47], these cannot be applied in a black-box manner in all settings.

1.3 Further Related Work

Recent years have seen several results for problems in power graphs in the CONGEST model, reaching from verifying solutions efficiently (or showing that this is not possible) [29], and answering several computational questions in the settings, e.g., for the already discussed problem of distance-2 coloring [41, 42] or optimization problems [11].

Ruling sets. We first focus on results in LOCAL. We have already discussed the deterministic ruling set algorithm of [45, 50, 59]. Actually, the result is more general and it provides an $O(\log_B C)$-ruling set of G in $O(B \cdot \log_B C)$ time if the graph is equipped with a vertex coloring with C colors. By choosing the parameter B appropriately and combining it with Linial's algorithm that computes a $O(\Delta^2)$-coloring in $O(\log^* n)$ rounds, one can compute a β-ruling set in $O(\beta \cdot \Delta^{2/\beta} + \log^* n)$ rounds.

Gfeller and Vicari [30] provide a randomized sparsification algorithm to compute a $O(\log\log n)$-dominating set[2] $S \subseteq V$ in $O(\log\log n)$ rounds, such that the maximum degree of $G[S]$ is $O(\log^5 n)$. This can be combined with the aforementioned approach to obtain a randomized an $O(\log\log n)$-ruling set algorithm with round complexity $O(\log\log n)$. Ghaffari extends this approach to G^k in the CONGEST model and obtains an $O(k\log\log n)$-ruling set in $O(k^2\log\log n)$ rounds [33].

Kothapalli and Pemmaraju [48] provide another sparsification method that computes a dominating set with degree $O(\Delta'\log n)$ in $O(\log\Delta/\log\Delta')$ rounds. Ghaffari uses multiple iterations of this sparsification method to compute a β-ruling set in $O(\beta\log^{1/\beta}\Delta)$ + poly $\log\log n$ rounds in LOCAL [31]. He also provides similar but slightly weaker results in CONGEST [33].

The recently developed powerful round elimination technique [14] has been used to prove lower bounds for the computation of ruling sets in LOCAL [8, 9]. Parameterized by the number of nodes the lower bounds for β-ruling sets is $\Omega(\log n/(\beta\log\log n))$ for deterministic algorithms and $\Omega(\log\log n/(\beta\log\log\log n))$ for randomized algorithms, as long as β is at most $\approx \sqrt{\log n/\log\log n}$ and $\approx \sqrt{\log\log n/\log\log\log n}$, respectively. As a function of the maximum degree Δ, the lower bound is $\Omega(\beta\cdot\Delta^{1/\beta})$. In [54, Table 1], we provide an overview of known ruling set and MIS algorithms and contrast them with our results.

Related work for graph sparsification. A fundamental building block of our most involved result (Theorem 1.1) is the method of graph sparsification [2, 5, 19, 20, 32, 40, 44]. Roughly speaking, the idea is to turn the input graph into a sparser representation that still allows us to solve the given problem. Sparsification has been used in many different settings and computational models and often the exact properties we want from the sparsification vary depending on the setting.

Global sparsification. In models with $\widetilde{O}(n)$ memory, such as streaming, sketching, congested clique, and linear memory MPC, the goal is to reduce the total (global) number of edges in the graph. If the sparser representation has roughly *n edges*, then it can be processed locally in constant time. For example, the current state-of-the-art for connectivity/MST [46, 55], MIS [1, 35], and $(\Delta+1)$-vertex-coloring (and list-coloring) [3, 5, 17, 20] are based on this method.

Local sparsification. Another sparsification approach is to sparsify the graph *locally*. To get an intuition, suppose that the output of each node v depends only on some local information, i.e., the output of node v can be decided by examining the graph in the small T-hop neighborhood around node v. Then, a local sparsification reduces the number of edges in that neighborhood and, at the same time, preserves the property that a correct output for v can be determined from the local neighborhood of v in the sparser graph. Combined with the memory-hungry graph exponentiation technique [51], this approach has been successfully used in sublinear models such as the low-space MPC model. For example, the current state-of-the-art for MIS [19, 40] and $(\Delta+1)$-vertex-coloring [17, 20] are based on this method.

In the context of the CONGEST model, locally sparsifying the communication graph is a natural way to avoid congestion. In one previous work, this line of thinking was used for vertex coloring G^2 [41]. However, in the context of coloring, one can split the *problem* into independent instances with disjoint color palettes. This property creates a fundamental difference between coloring and other symmetry breaking problems, such as MIS and ruling sets, which do not enjoy the luxury of splitting into disjoint instances. Sparsified graphs are also of independent interest, for example in the context of finding distance preserving spanners, both in cases of linear memory models [22, 23] and local sparsification [28].

1.4 Roadmap

Space constraints prevent us from providing full details in the proceedings version of the paper. We compromise with a rather detailed technical overview that we present in Section 3. See [54] for the full version of the paper. Section 2 introduces our notation and tools on limited independence. As mentioned in the introduction, our main technical contribution is a deterministic sparsification lemma for power graphs (see Lemma 3.1 in Section 3 for the precise statement). In Section 4, we provide the essential ingredients for this sparsification lemma. It is based on derandomizing a randomized sparsification procedure via the method of conditional expectation. In Section 4.1 we present the randomized process and prove the properties that are necessary for its derandomization. In Section 4.2

[2]The definition of ruling sets is different in [30]: what we refer to as a t-dominating set is called a t-ruling set in [30], while a $(2,t)$-ruling set is called an independent t-ruling set.

we sketch the most crucial ingredients for the derandomization. Finally, in Section 4.3 we describe how to simulate the deterministic sparsification algorithm on power graphs to iteratively sparsify $G, G^2, \ldots, G^k$.

2 NOTATION AND k-WISE INDEPENDENT RANDOM VARIABLES

We always use $G = (V, E)$ to refer to the original input graph. Similarly, the degree $d(v)$ and (non-inclusive) neighborhood $N(v)$ of a vertex do not change throughout our algorithms. For $s \geq 0$, the *power graph* G^s is the graph where $V(G^s) = V$ and $E(G^s) = \{\{v, w\} \in V \times V : \text{dist}_G(v, w) \leq s\}$. A subgraph of the power graph induced by a set of nodes X is denoted $G^s[X]$. Note that this is not the same as $(G[X])^s$: the latter only contains edges formed by paths only using nodes in X. $N^s(v)$ is called the *distance-s neighborhood of v*, which is the neighborhood of v in G^s. Let $d^s(v) := |N^s(v)|$. For any $X \subseteq V$, let $N^s(X) := \cup_{v \in X} N^s(v)$. We use *distance-$s$ X-neighborhood* to refer to $N^s(v, X) := N^s(v) \cap X$. *Distance-$s$ X-degree* is defined as $d^s(v, X) := |N^s(v, X)|$.

$\mathcal{I} \subseteq V$ is α-independent in G, if for all distinct $v, w \in \mathcal{I}$, $\text{dist}_G(v, w) \geq \alpha$. For $\alpha = 2$, we simply say that $\mathcal{I}$ is *independent* in G. For any $S \subseteq V$, $Q \subseteq S$ is a β-dominating set of S, if for all $u \in S$, there exists some $v \in Q$ such that $\text{dist}_G(u, v) \leq \beta$. When $S = V$, we say that Q is a β-dominating set. An (α, β)-*ruling set* of a graph $G = (V, E)$ is a subset $Q \subseteq V$, such that Q is α-independent and β-dominating. α and β are generally referred to as the *independence* and *domination* parameters, respectively. Note that a $(2, 1)$-ruling set is a maximal independent set of G and a $(k+1, k)$-ruling set is a maximal independent set of G^k. A set $S \subseteq V$ is k-connected in G if for all $S' \subset S : \text{dist}_G(S', S \setminus S) \leq k$. Equivalently, S is k-connected if $G^k[S]$ is connected.

A collection of discrete random variables $X_1, \ldots, X_n$ is *k-wise independent* if for any $I \subseteq [n]$ with $|I| \leq k$ and any values x_i, we have $\mathbb{P}(\wedge_{i \in I} X_i = x_i) = \prod_{i \in I} \mathbb{P}(X_i = x_i)$. We can simulate such variables by picking a random hash function from a family of k-wise independent hash functions:

Definition 2.1. For $N, L, k \in \mathbb{N}$ such that $k \leq N$, a family of functions $\mathcal{H} = \{h : [N] \to [L]\}$ is k-wise independent if for all distinct $x_1, \ldots, x_k \in [N]$, the random variables $h(x_1), \ldots, h(x_k)$ are independent and uniformly distributed in $[L]$ when h is chosen uniformly at random from $\mathcal{H}$.

Lemma 2.2 (Corollary 3.34 in [60]). *For every a, b, k, there is a family of k-wise independent hash functions $\mathcal{H} = \{h : \{0,1\}^a \to \{0,1\}^b\}$ such that choosing a random function from $\mathcal{H}$ takes $k \cdot \max\{a, b\}$ random bits, and evaluating a function from $\mathcal{H}$ takes time $\text{poly}(a, b, k)$.*

3 TECHNICAL OVERVIEW

3.1 Deterministic Sparsification

Our main technical ingredient is a deterministic sparsification procedure. From a high level point of view, we gradually sparsify the input graph, i.e., we compute a sequence of sparser node sets $V \supseteq Q_0 \supseteq Q_1 \supseteq \ldots \supseteq Q_k$, while ensuring that every node in V remains within constant distance of the set. More precisely, for each $1 \leq s \leq k$, the distance from any node in Q_{s-1} to Q_s is at most 2 in G^s (at most $2s$ in G), and the set Q_s is sparse in G^s, that is, every node has a bounded number of Q_s-neighbors in G^s. We use the sparsity of Q_s to efficiently limit the congestion when computing the sparser set Q_{s+1}. At the end Q_{k-1} is sparse enough to efficiently simulate any algorithm on $G^k[Q_{k-1}]$, i.e., the subgraph of G^k induced by Q_{k-1}. For example, to compute a k-ruling set of G^k, we can use sparsification to find Q_{k-1} and then compute an MIS of Q_{k-1} on the power graph G^k, where only nodes in Q_{k-1} are allowed to enter the MIS.

We believe that this sparsification procedure is of independent interest and may be helpful for other problems and in other models of computation. We summarize it in the next lemma. Recall, the distance-k Q-degree of a node v is the number of neighbors in $N^k(v) \cap Q$.

Lemma 3.1 (Sparsification in Power Graphs). *Let $k \geq 1$ (potentially a function of n). There is a deterministic distributed algorithm that, given a subset $Q_0 \subseteq V$, finds a set of vertices $Q \subseteq Q_0$ such that for all $v \in V$,*

- *(bounded distance-k Q-degree): $d^k(v, Q) \leq 72 \log n = O(\log n)$*
- *(domination): $\text{dist}_G(v, Q) \leq k^2 + k + \text{dist}_G(v, Q_0)$*

The algorithm runs in $O(\text{diam}(G) \cdot k \cdot \log^2 n \cdot \log \Delta + k^2 \cdot \log \Delta)$ rounds in the CONGEST *model.*

Using a network decomposition, we can replace the diameter factor in Lemma 3.1 with $O(k \cdot \log n)$ (see [54, Section 5.4] for more details and in particular [54, Lemma 5.8] for the precise statement). For $k = 1$ (and if initialized with $Q_0 = V$), we obtain a polylogarithmic-round algorithm to compute a set Q with domination distance 2, such that all nodes have at most $O(\log n)$ neighbors in Q. Note that the degree bound is a *local property*. If one instead wants to compute a set with the same domination distance and *globally* minimize its size, it is known that even constant approximation algorithms require near-quadratic time in CONGEST [11].

How does the sparsification work? The core idea of our deterministic sparsification is to derandomize the following randomized sampling algorithm: First assume that the graph G^k is Δ^k-regular. We can sample every node with probability $O(\log n / \Delta^k)$ into a set Q. Then every node in V gets $\Theta(\log n)$ distance-k Q-neighbors, with high probability. Even though this is a trivial 0-round algorithm, it is non-trivial to derandomize it, as the constraints imposed by the nodes depend on the decision of nodes in distance $\Theta(k)$. Generally, there is little hope that one can directly (and deterministically) find good random bits (aka good conditions) in order to fulfill the constraints of all nodes. Instead, we perform a more fine-grained sparsification method, in which we gradually sparsify the graph, each time slightly losing in the domination property. This can be viewed as first computing the set Q_k for $k = 1$, then for $k = 2$, and so on.

Continuing in the randomized setting, we explain our approach for finding Q_1 in G. The sampling algorithm has two objectives. For all $v \in V$, the number of neighbors in Q_1 should be at most $O(\log n)$, while the distance to Q_1 should be at most 2 (or at most some constant). Since degrees are not uniform, it is difficult to find the right sampling probability that satisfies the two objectives for all nodes. Hence, the process is slowed down to $O(\log \Delta)$ *stages*. All nodes in Q_0 start as *active* ($Q_0 = V$ in our applications). In

each stage, we sample active nodes to Q_1. Initially, the sampling probability is low, so that high degree nodes get at least one sampled neighbor, while not exceeding the $O(\log n)$ bound. At the end of each stage, the distance-2 neighborhood of sampled nodes is deactivated. This guarantees that high degree nodes do not get more sampled neighbors in later stages, effectively decreasing the maximum *active degree*. The decrease in active degree means that we can sample in the next stage with a slightly higher probability. In the end, only nodes with low active degree remain, which can be included in the sampled set. The result Q_1 is a 2-dominating set, while all $v \in V$ have at most $O(\log n)$ Q_1-neighbors.

Sparsifying G. Our deterministic sparsification for G is based on derandomizing the previous sampling algorithm. The randomized analysis works with $O(\log n)$-wise independence. This allows simulating the randomness in one stage with a $O(\log^2 n)$-bit random seed. Derandomization is done stage by stage, using the method of conditional expectations. In each stage, we need to guarantee two events: (1) each node gets at most $O(\log n)$ sampled neighbors, while (2) each high active degree node gets at least one of its neighbors sampled. Both events only depend on the events in the immediate neighborhood in G, which makes the required information easily available for each node. The bits of the seed are fixed one by one. In order to fix a single bit b (to 0 or 1) of the seed, each node computes conditional expectations for its two events, for both choices of b. To compute the conditional expectations, a node needs to know the values of the already fixed bits of the seed and the IDs of its active neighbors. In G, learning the relevant identifiers can be done in one round, because the two events are determined by the decisions of active neighbors in the immediate neighborhood. Then, the conditional expectations of all nodes are aggregated (summed up) at a leader node who can then decide on the better choice for the bit b. Then all nodes proceed with the next bit. The method of conditional expectation implies that all nodes' events hold at the end of this process.

Sparsifying power graphs. Fix some $1 \le s \le k$. The sth *iteration* of the sparsification is simulated on the power graph G^s. The output of the previous iteration acts as the initial set of active nodes Q_{s-1}. The sth iteration results in $Q_s \subseteq Q_{s-1}$, where Q_s is sparse in G^s, while weakening the domination distance of Q_{s-1} by at most 2 in G^s (or $2s$ in G). The sum of increases in distance over s iterations is $\sum_{j=1}^{s} 2j \le s^2 + s$, hence Q_s is a (s^2+s)-dominating set of G (when initialized with $Q_0 = V$).

A main challenge of our work is to ensure that all nodes can obtain the necessary information for derandomization (distance-s neighbor's random bits and IDs) when sparsifying power graphs. To guarantee sparsity in G^s, all nodes must remain as observers (and also relay messages), taking part in the derandomization. Here, in a nutshell, the sparsity with regard to the previous iteration helps to learn the required information. More formally, we build communication tools to efficiently run the algorithm on G^s, relying on the sparsity of Q_{s-1}.

Communication tools. In order to benefit from the sparsity of $Q := Q_s$, we develop communication tools (see [54, Lemma 4.2]) that allow us to execute basic communication primitives on power graphs. The sparsity of Q in G^s can be used to efficiently send messages from Q to their neighbors in G^{s+1}. The communication algorithms include sending broadcasts from nodes in Q to their neighbors in G^{s+1} in $O(s+\log n)$ rounds, and simulating one round of a CONGEST algorithm on $G^{s+1}[Q]$ in $O(s+\log^2 n)$ rounds (then, one can basically assume that the algorithm is running on the communication network $G^{s+1}[Q]$ with $O(s+\log^2 n)$ overhead). The efficiency is based on the bounded number $\widehat{\Delta}$ of distance-s neighbors in Q for *all* nodes in G. This effectively bounds the number of messages forwarded through any edge in the graph. For example with broadcasts, for any edge $\{v, w\}$ of the communication network, the number of nodes $x \in Q$ whose broadcast must be forwarded from v to w is at most $d^s(v, Q) \le \widehat{\Delta}$, because the message is forwarded for at most $s+1$ hops. The communication tools are also used to obtain the sparsification lemma (Lemma 3.1), where the sparsity of Q_s in G^s is used to run the $(s+1)$th iteration of sparsification on G^{s+1} efficiently. In our ruling set application, we use the communication tools to simulate an MIS algorithm on G^k.

Deterministic k-ruling set of G^k. Our deterministic sparsification algorithm is used to compute a sparse subset $Q := Q_{k-1} \subseteq V$, while maintaining constant domination distance to the rest of the graph. After sparsifying the graph, we compute an MIS of $G^k[Q]$. Using our communication tools, we can simulate any MIS algorithm on $G^k[Q]$ in a black-box manner. In general, this approach yields a $(k+1, \beta+k)$-ruling set of G, where β is the domination distance of Q (see [54, Lemma 6.3] for the formal statement). With our sparsification algorithm, the result is a $(k+1, k^2)$-ruling set of G, or equivalently a k-ruling set of G^k. Messaging between distance-k neighbors in Q can be implemented with a $(k+\log^2 n)$-factor slowdown. Combined with the state of the art MIS algorithm [25], we compute a $(k+1, k^2)$-ruling set in $\text{poly}(k, \log n)$-rounds (Theorem 1.1). For a constant $k > 1$ this improves exponentially upon prior work that required $O(k^2 \cdot n^{1/k})$ rounds [24, 50, 59].

3.2 Randomized MIS of G^k

For our randomized results, we use the *shattering* framework, where a randomized *base algorithm* is used to solve the problem efficiently on most parts of the graph. The remaining connected components (in G^k) are small, with high probability. The small connected components are solved with a different algorithm. Also see the next paragraph on Theorem 1.4 for further details on the shattering framework. For the base algorithm, we cannot use a black-box simulation of Ghaffari's MIS algorithm from [31], as simulation on G^k would be prohibitively expensive. Hence, we use the *BeepingMIS* algorithm of [32] (with a minor but crucial modification), which works in a simple beeping model of communication. However, we need to be careful when forwarding the beeps: With cycles in the communication network, nodes may confuse the beep of a neighbor with that of their own, when $k \ge 3$. To avoid this, we equip the beeps with an identifier of the beeping node. Nodes forward an arbitrary subset of at most two beeps, which is enough for any beeping node to distinguish if there is a beeping neighbor or not. In the *post-shattering phase*, the remaining unsolved parts of the graph form small connected components in G^k, each with $N = O(\Delta^{4k} \cdot \log n)$ nodes, with high probability. To find a solution in the remaining components, we run $O(\log_N n)$ executions

of the BeepingMIS algorithm in parallel, which guarantees that at least one of the executions succeeds, with high probability in n. To limit congestion, we assign unique IDs to the nodes in the connected component from $[N]$. This bounds the total communication to $O(\log_N n \cdot \log N) = O(\log n)$ (for simulating one step for all instances). This approach achieves the same runtime as the state of the art algorithm for MIS of G [33], up to slowdown factors of k.

3.3 Shattering in G, in LOCAL and CONGEST.

Since the seminal work of [12] the shattering technique has become an essential tool in the area. The technique has two phases. In the *pre-shattering phase* the problem at hand (e.g., the MIS problem) is solved in large parts of the graph via a very efficient randomized process such that with high probability only small components—think of components of polylogarithmic size—remain afterwards. In the *post-shattering phase* one employs a different algorithm, usually a deterministic algorithm, that *finishes* the small components very efficiently by exploiting their small size. The difficulty here is that the components are actually of size poly $\Delta \cdot \log n$, which is not small for large values of Δ.

Fortunately, components have further beneficial properties: Fix a small component C and some suitable β and $\alpha \geq 5$ (the constant 5 depends on the properties of the pre-shattering phase and works for the MIS algorithms in [12, 31, 33], other problems may need other values). Then it is known that any (α, β)-ruling set R_C of C has at most $O(\log n)$ nodes. If we had such a ruling set available, we could exploit its size algorithmically by creating a virtual graph as follows: Each node in R_C forms a connected *ball* of nodes around it, such that each node in C joins a unique ball. Then, one can build a virtual graph H with one vertex for each such ball. Two balls are connected in H if the respective balls contain nodes that are adjacent in the original graph G. Since $|V(H_C)| = |R_C| = O(\log n)$, and since one round of communication in H_C can be simulated in G in $O(\beta)$ rounds (in the LOCAL model) one can compute a network decomposition (ND) of H very efficiently, e.g., with the algorithm of [57]. The details do not matter for the current exposition, but it is known that an ND of H_C implies an ND of C which can be used to compute an MIS on C.

The challenge. The challenge is that it is unclear how to compute a ruling set R_C. The algorithm would have to run on all small components in parallel and running it on the induced subgraph does not work as the resulting ruling set needs to be 5-independent in G; 5-independence in $G[C]$ is insufficient. It is also critical if a node $v \in C, v \neq R_C$ is dominated by some node in the ruling set of some other small component $C' \neq C$. To illustrate the difficulty, observe that a node cannot tell efficiently whether a node in distance 5 in G is in the same small component as itself or not. Hence, the nodes cannot easily determine whether they can both be contained in the ruling set or not.

The solution. Inspired by a combination of the different versions of [12], we present the following solution. After the pre-shattering phase, we perform a second randomized pre-shattering phase that is run on all small components in parallel (it works w.h.p. in n and there are at most n components so we can perform a union-bound over the error probabilities of different components). It splits each component into so called *tiny components*. Then, we prove the following lemma.

Lemma (Informal version of [54] Lemma 7.5). Let $\alpha = 5$ and let β be an integer and consider a small component C. Then, each (α, β)-ruling set of its tiny components has at most $O(\log n)$ nodes. Both distances of the ruling set are measured in the graph induced by C and the bound on the size even holds if a node of one tiny component is dominated by a node in another tiny component.

The benefit of this lemma is that we can run a ruling set algorithm (in order to dominate all remaining nodes in tiny components) on the graph induced by the small components, making the algorithm fully independent between different small components. Instead of worrying whether a node in distance 5 in G is within the same component as oneself, one can simply ignore all edges that are not contained in $G[C]$ and work in the graph induced by small components.

Proof of the lemma. While the whole setup is rather complicated and very similar to the one in [12], the proof of the lemma is very simple. If we fix one specific ruling set R_C as in the lemma, then the core shattering argument of [12] states that w.h.p. in n the set R_C is of size $O(\log n)$. The intuitive reason for this fact is that each node only remains undecided with a small probability and these events are independent for nodes that are at least 5 hops apart. Hence, w.h.p. not more than $O(\log n)$ nodes with pairwise distance 5 can remain undecided. Now, we obtain the lemma by performing a union bound over all such ruling sets. The crucial point in this solution is that one can rely on the small size of C (that holds w.h.p. after the first pre-shattering phase) to show that there are only poly n many of these sets making the union bound a feasible approach.[3]

Previous solutions. The arXiv version of [12] has a very similar two-phase setup, but uses a different (and faulty[4]) approach to prove a statement that is similar to our lemma. The (correct) journal version of [12] also uses a similar two-phase structure, but then uses a different and more involved argument to show that a network decomposition of the virtual graph H can be computed efficiently. Besides some other technical details, the main difference is that they show that each ball satisfies certain power graph connectivity requirements if nodes are assigned to rulers *on-the-fly* while computing the ruling set (their proof is insufficient if nodes simply join the ball of the closest ruler). Hence, the proof requires internals of the used ruling set algorithm and it is difficult to formally black-box it, as it is done in later works.

Our second solution. Our second solution shows that using the internals of current state-of-the-art ruling set algorithms one can omit the second pre-shattering phase. While this is simpler than the proof in the journal version of [12], we believe that our solution that is based on the two-phase structure is more approachable. However, the version with a single pre-shattering phase is easier to generalize to G^k, which is needed for Theorem 1.2 and Corollary 1.3.

[3] The informed reader will notice that this is different from (and simpler than) the standard proof for showing that small components emerge after the first pre-shattering phase. In that setting a similar union-bound is not possible.
[4] See [54, Section 7] for details.

Algorithm 1: Randomized sparsification

Input: $\Delta_A \geq \max_{v \in V} d(v, A)$, Each $v \in V$ knows if it is in a set of initially active nodes $A \subseteq V$.

Set $r := \lfloor \log \Delta_A - \log\log n \rfloor - 5$ and $H_1 := A$

for stage $i = 1, \ldots, r$ **:**

  $M_i := \emptyset$

  foreach $v \in H_i$ *in parallel* **:**

    Join M_i with probability $\frac{24 \cdot 2^i \cdot \log n}{\Delta_A}$

  $H_{i+1} := H_i \setminus (M_i \cup N^2(M_i))$

$M_{r+1} := H_{r+1}$

return $Q := \cup_{i=1}^{r+1} M_i$

4 SPARSIFICATION OF POWER GRAPHS

The goal of this section is to prove Lemma 3.1, our main sparsification result for power graphs. First, in the next two sections, we prove the following deterministic sparsification result. Lemma 4.1 is standalone and does not refer to power graphs. In Section 4.3, we use it iteratively to obtain a sequence $V \supseteq Q_1 \supseteq Q_2 \supseteq \ldots \supseteq Q_k$ where Q_i is sparse in the power graph G^i.

Lemma 4.1 (Deterministic Sparsification). *Let $A \subseteq V$ be a set of initially active vertices. There is a deterministic distributed algorithm that finds a set of vertices $Q \subseteq A$ such that for all $v \in V$,*

- *(bounded Q-degree): $d(v, Q) \leq 72 \log n = O(\log n)$*
- *(domination): $\mathrm{dist}_G(v, Q) \leq 2 + \mathrm{dist}_G(v, A)$*

Let $\Delta_A \geq \max_{v \in V} d(v, A)$ be an input parameter given to all nodes, which is at least the maximum number of active neighbors and a power of 2. The algorithm runs in $O(\mathrm{diam}(G) \cdot \log^2 n \cdot \log \Delta_A)$ rounds in CONGEST.

4.1 Randomized Sparsification via Sampling

We start with a *randomized* sparsification algorithm to find a certain sparse set of vertices, satisfying the properties of Lemma 4.1, with high probability. See Algorithm 1. The algorithm consists of $r := \lfloor \log \Delta_A - \log\log n \rfloor - 5$ stages. Let $H_1 := A$, where $A \subseteq V$ is any set of initially active nodes. For $1 \leq i \leq r$, H_i is a set of *active nodes* in the respective stage. In each stage i, we sample a set $M_i \subseteq H_i$. Each node $v \in H_i$ is included in M_i with probability $\frac{24 \cdot 2^i \cdot \log n}{\Delta_A}$, where the decisions of the nodes are $8 \log n$-wise independent. We deactivate all sampled nodes, as well as active nodes that have a sampled node within 2 hops in G. This is done by sending a flag from each sampled node, propagated for two hops, where multiple incoming flags can be forwarded as one. Once nodes are deactivated, they stay inactive forever. Let $H_{i+1} = H_i \setminus (M_i \cup N^2(M_i))$ be the remaining active nodes. After r stages, the algorithm returns $Q := \cup_{i=1}^{r+1} M_i$, consisting of the sampled sets $M_1, \ldots, M_r$ and the remaining active nodes $M_{r+1} := H_{r+1}$.

Definition 4.2 (Active degree). For $v \in V$, its *active degree* in stage i is defined as $d(v, H_i)$. We say that $v \in V$ has a *high active degree in stage i* if $d(v, H_i) \geq \Delta_A / 2^i$.

The active degree of a node changes throughout the algorithm. Note that inactive nodes are never reactivated, so we have $d(v, H_1) \geq d(v, H_2) \geq \cdots \geq d(v, H_{r+1})$ for all $v \in V$. Also, whether v itself is active or inactive does not affect its active degree directly. Next, we prove the properties that hold after each stage with high probability.

Theorem 4.3 (Theorem 2.5 in [58]). *Let X be the sum of p-wise independent $[0, \lambda]$-valued random variables with expectation $\mu = \mathbb{E}(X)$ and let $\delta \leq 1$. Then $\mathbb{P}(|X - \mu| \geq \delta\mu) \leq e^{-\lfloor \min\{p/2, \delta^2\mu/(3\lambda)\} \rfloor}$.*

Lemma 4.4 (Stage i of randomized sparsification). *Fix some stage $1 \leq i \leq r$ and let $H_i \subseteq V$ be a set of active nodes, such that all nodes $v \in V$ have at most $\Delta_A / 2^{i-1}$ neighbors in H_i. The ith stage of Algorithm 1 returns M_i and H_{i+1} such that for all $v \in V$:*

(i) *$d(v, M_i) \leq 72 \log n$, with probability at least $1 - 1/n^3$.*

(ii) ***if** $d(v, H_i) \geq \Delta_A / 2^i$, **then** $v \in M_i \cup N(M_i)$, with probability at least $1 - 1/n^3$.*

(iii) *$d(v, H_{i+1}) < \Delta_A / 2^i$, with probability at least $1 - 1/n^3$.*

One stage requires 2 rounds. The claims hold if the random choices are only $8 \log n$-wise independent.

Proof. For an active node $w \in H_i$, let X_w be an indicator variable for the event that w is sampled.

Proof of (i). Let $v \in V$ be any vertex. By assumption, v has at most $\Delta_A / 2^{i-1}$ active neighbors. Let W be a set of fake vertices, added to the set of active neighbors of v such that v has exactly $\Delta_A / 2^{i-1}$ active neighbors. Define the indicator variable X_w for fake vertices $w \in W$ similarly as for real active nodes. Clearly v has at most $72 \log n$ real sampled neighbors whenever at most $72 \log n$ vertices in $N(v, H_i) \cup W$ are sampled. The probability of this event can be lower bounded using Theorem 4.3. Let $X = \sum_{w \in N(v, H_i) \cup W} X_w$ be a sum of the indicator variables, with expected value $\mu := \mathbb{E}[X] = \frac{\Delta_A}{2^{i-1}} \frac{24 \cdot 2^i \cdot \log n}{\Delta_A} = 48 \log n$. Let $\delta = \frac{1}{2}$ in Theorem 4.3:

$$\mathbb{P}(X \geq 72 \log n) \leq \mathbb{P}(|X - 48 \log n| \geq \frac{1}{2} 48 \log n) \leq e^{-\lfloor \min\{8 \log n/2, 48 \log n/12\} \rfloor} = e^{-\lfloor 4 \log n \rfloor} \leq n^{-3}$$

Hence, v has at most $72 \log n$ neighbors in M_i with probability at least $1 - 1/n^3$.

Proof of (ii). Let $v \in V$ be a node with at least $\Delta_A / 2^i$ active neighbors. We will compute a lower bound for the probability that v is adjacent to a sampled node. Fix any subset $S \subseteq N(v, H_i)$ of active neighbors such that $|S| = \Delta_A / 2^i$. Let $X = \sum_{w \in S} X_w$ be a sum of indicator variables, with expected value $\mu := \mathbb{E}[X] = \frac{\Delta_A}{2^i} \frac{24 \cdot 2^i \cdot \log n}{\Delta_A} = 24 \log n$. At least one neighbor is sampled when $X > 0$. The probability of $X = 0$ can be upper bounded using Theorem 4.3. Let $\delta = \frac{3}{4}$:

$$\mathbb{P}(X = 0) \leq \mathbb{P}(|X - \mu| \geq (3/4) \cdot \mu) \leq e^{-\lfloor \min\{8 \log n/2, 72 \log n/16\} \rfloor} = e^{-\lfloor 4 \log n \rfloor} \leq n^{-3}$$

Hence, at least one active neighbor of v is sampled with probability at least $1 - 1/n^3$.

Proof of (iii). First, consider $v \in V$ with $d(v, H_i) < \Delta_A / 2^i$. Since nodes are never reactivated, the active degree of v will be less than $\Delta_A / 2^i$ in stage $i + 1$ as well. Now, let $v \in V$ be any node with high active degree in stage i, that is, $d(v, H_i) \geq \Delta_A / 2^i$. The algorithm deactivates the distance-2 neighborhood of sampled nodes. Hence, all of $N(v, H_i)$ is deactivated, whenever some $w \in N(v, H_i)$ is sampled. By (ii), this happens with probability at least $1 - \frac{1}{n^3}$. In this

case, the active degree $d(v, H_j)$ of v is zero for all remaining stages $i < j \leq r$. □

We are now ready to prove a randomized version of Lemma 4.1, running in $O(\log \Delta)$ rounds: The $r = O(\log \Delta_A) = O(\log \Delta)$ stages of Algorithm 1 produce $Q := \cup_{i=1}^{r+1} M_i$, consisting of the sampled sets $M_1, \ldots, M_r$ and the remaining active nodes $M_{r+1} := H_{r+1}$. Lemma 4.4 (i) states that any node has $O(\log n)$ neighbors in any M_i, with high probability, given that maximum active degree decreased to $\Delta_A/2^{i-1}$ in the previous stage. The maximum active degree decreases for any node, with high probability by Lemma 4.4 (iii). The sets M_i and M_j are at distance at least 2 from each other for any $i \neq j$, so any node does not have neighbors in more than one M_i. Taking a union bound over all stages, we get that any node has at most $O(\log n)$ neighbors in the whole $Q = \cup_{i=1}^{r+1} M_i$. This also includes the remaining active nodes $M_{r+1} := H_{r+1}$. Next, we construct a deterministic sparsification algorithm by derandomizing the random choices of the active nodes in each stage.

4.2 Deterministic Sparsification via Derandomization (Sketch)

We construct a deterministic sparsification algorithm to prove Lemma 4.1. The deterministic algorithm is referred to as DetSparsification (see [54, Algorithm 2] for the pseudocode). The structure of DetSparsification is the same as the randomized sparsification algorithm (Algorithm 1). As before, the input is a set of active nodes $A \subseteq V$ and a maximum active degree parameter $\Delta_A \geq \max_{v \in V} d(v, A)$, where each $v \in V$ knows whether $v \in A$, and the value of Δ_A. There are $r = \lfloor \log \Delta_A - \log \log n \rfloor - 5$ stages. For $1 \leq i \leq r$, H_i is the set of active nodes in the ith stage, where $H_1 := A$ and $H_i \subseteq H_{i-1}$. Fix some stage $1 \leq i \leq r$. In the ith stage, DetSparsification selects a set $M_i \subseteq H_i$ by derandomizing the sampling procedure of the ith stage of Algorithm 1, in $O(\mathrm{diam}(G) \cdot \log^2 n)$ rounds. This is described in the next lemma. The rest of the algorithm works exactly like the randomized version. The remaining active nodes are $H_{i+1} = H_i \setminus (M_i \cup N^2(M_i))$. After r stages, the algorithm returns $Q := \cup_{i=1}^{r+1} M_i$, consisting of the sampled sets $M_1, \ldots, M_r$ and the remaining active nodes $M_{r+1} := H_{r+1}$.

Lemma 4.5 (Derandomizing ith stage). *The round complexity of a stage is $O(\mathrm{diam}(G) \cdot \log^2 n)$. Fix a stage $1 \leq i \leq r$ and let $H_i \subseteq V$ be a set of active nodes, such that all nodes $v \in V$ have at most $\Delta_A/2^{i-1}$ neighbors in H_i. The ith stage of DetSparsification returns M_i and H_{i+1} such that for all $v \in V$:*

(i) $d(v, M_i) \leq 72 \log n$,
(ii) ***if*** $d(v, H_i) \geq \Delta_A/2^i$, ***then*** $v \in M_i \cup N(M_i)$,
(iii) $d(v, H_{i+1}) < \Delta_A/2^i$.

Proof Sketch. The result is obtained by derandomizing the algorithm of Lemma 4.4 where the randomness of nodes is provided by a *random seed* of length $O(\log^2 n)$. Given a seed (and the tools from Section 2), there is a deterministic mapping from IDs to a decision whether to join M_i or not and the random choices of nodes are $8 \log n$-wise independent, which is sufficient to obtain the probabilistic guarantees of Lemma 4.4. The goal is to deterministically fix the bits of the random seed such that, for all nodes, the events of (i) and (ii) occur. This implies (iii). The complements of these events are regarded as *bad events*. The events (i) and (ii) occur for each node, with high probability by Lemma 4.4. In expectation, the total number (of all nodes) of bad events occurring is less than one. We use the method of conditional expectations to fix the random seed, such that none of the bad events occur. The bits of the random seed are fixed one by one. Given the values of previously fixed bits and a candidate value for the next bit to be fixed, each node computes the expected number of bad events occurring for it (a real number between zero and two), where the randomness is over the remaining unfixed (uniformly random) bits. The events only depend on the decisions of active neighbors. The crucial point is that, given the IDs of the active neighbors (as this lemma talks about sparsifying G and not a power graph these IDs can be learned in one round; for power graphs additional work is needed, see [54, Section 5.3] where we iterate this lemma), these expected values can be computed in zero communication rounds. The conditional expectations are aggregated to a leader node in $O(\mathrm{diam}(G))$ rounds and the leader fixes the next bit such that the conditional expectation is minimized, and informs the graph about its choice. □

Proof of Lemma 4.1. Run the algorithm DetSparsification. The number of stages is $r = \lfloor \log \Delta_A - \log \log n \rfloor - 5$. Note that we can assume that $\Delta_A \geq 2^5 \log n$,[5] which makes r non-negative. DetSparsification returns $Q = \cup_{i=1}^{r+1} M_i \subseteq A$, where $M_1, \ldots, M_r$ are the sets selected in stages $1 \leq i \leq r$ and $M_{r+1} = H_{r+1}$ is the set of remaining active nodes.

Domination property: We start by showing the domination property of Q, that is, $\forall v \in V : \mathrm{dist}_G(v, Q) \leq 2 + \mathrm{dist}_G(v, A)$. Let $v \in V$ be any node and let $w \in A$ be any initially active node that is closest to v, i.e., $\mathrm{dist}_G(v, w) = \mathrm{dist}_G(v, A)$. There are three possible outcomes for w in the algorithm: (1) $w \in Q$. Now $\mathrm{dist}_G(v, Q) = \mathrm{dist}_G(v, A)$. (2) w is deactivated in some stage i. By definition, there exists some $w' \in N^2(w)$ such that $w' \in M_i$. Hence $\mathrm{dist}_G(v, Q) \leq \mathrm{dist}_G(v, A) + 2$. (3) w is never deactivated. Hence $w \in H_{r+1}$. The remaining active nodes $M_{r+1} := H_{r+1}$ are included in Q. Hence $\mathrm{dist}_G(v, Q) = \mathrm{dist}_G(v, A)$. In all cases, $\mathrm{dist}_G(v, Q) \leq 2 + \mathrm{dist}_G(v, A)$.

Sparsity: We prove the claim about bounded Q-degree, $\forall v \in V : d(v, Q) \leq 72 \log n$. For any stage $1 \leq i \leq r$, Lemma 4.5 states that any $v \in V$ has at most $72 \log n$ neighbors in M_i, assuming that the maximum active degree at the start of stage i is at most $\Delta_A/2^{i-1}$. The assumption holds for $i = 1$, since $\Delta_A \geq \max_{v \in V} d(v, A)$ by definition. For stages $i = 2, \ldots, r$, this is guaranteed by Lemma 4.5 (iii), which states that the maximum active degree at the end of stage $i - 1$ is at most $\Delta_A/2^{i-1}$. Finally, $M_{r+1} = H_{r+1}$ consists of the nodes who are still active after r stages of sampling. By Lemma 4.5 (iii), the maximum active degree after stage $r = \lfloor \log \Delta_A - \log \log n \rfloor - 5$ is

$$\frac{\Delta_A}{2^{\lfloor \log \Delta_A - \log \log n \rfloor - 5}} \leq \frac{\Delta_A}{2^{\log \Delta_A - \log \log n - 6}} = 64 \log n$$

Hence, any $v \in V$ has at most $72 \log n$ neighbors in M_i, for any $1 \leq i \leq r + 1$. Finally, nodes in M_i and M_j, $i \neq j \in 1, \ldots, r$ do not have any common neighbors, because the distance-2 neighborhood of sampled nodes is deactivated. Nodes in M_{r+1} and M_i, $1 \leq i \leq r$ do not have any common neighbors for the same reason. We conclude that each node has at most $72 \log n = O(\log n)$ neighbors in Q.

[5] If $\Delta_A < 2^5 \log n$, then $\max_{v \in V} d(v, A) < 2^5 \log n$ by definition of Δ_A. Return the initial set of active nodes A, which now satisfies both conditions of Lemma 4.1.

The runtime of DetSparsification consists of $\lfloor \log \Delta_A - \log\log n \rfloor - 5$ stages, each taking $O(\mathrm{diam}(G) \cdot \log^2 n)$ rounds by Lemma 4.5. The total runtime is $O(\mathrm{diam}(G) \cdot \log \Delta \cdot \log^2 n)$. □

4.3 Sparsification in Power Graphs (Sketch)

We present a deterministic sparsification algorithm for G^k, proving Lemma 3.1. Our algorithm consists of k iterations of the DetSparsification algorithm (Lemma 4.1). In iteration $1 \le s \le k$, the algorithm is simulated on a power graph G^s, using our communication tools [54, Section 4]. The result $Q_s \subseteq V$ of the sth iteration is used as the initial active nodes in the $(s+1)$th iteration. We must guarantee that DetSparsification can be efficiently simulated on G^s, while ensuring that all nodes stay within $O(k^2)$ distance from the selected nodes. We do this by maintaining the following invariants for $Q_0 \supseteq Q_1 \supseteq \cdots \supseteq Q_k$. After the sth iteration, Q_s satisfies:

- **I1.1** (bounded distance-s Q_s-degree) $\forall v \in V : d^s(v, Q_s) \le 72 \log n$
- **I1.2** (bounded distance-$(s+1)$ Q_s-degree) $\forall v \in V : d^{s+1}(v, Q_s) \le 72\Delta \log n$
- **I2** (domination) $\forall v \in V : \mathrm{dist}_G(v, Q_s) \le \sum_{j=1}^{s} 2j + \mathrm{dist}_G(v, Q_0) = s^2 + s + \mathrm{dist}_G(v, Q_0)$
- **I3** (knowledge of distance-$(s+1)$ Q_s-neighborhood) All $v \in V$ know the set of IDs in $N^{s+1}(v, Q_s)$. Moreover, for each $x \in Q_s$, there is a BFS tree of depth $s+1$ rooted in x. Each $v \in V$ knows, for each T_x it belongs to, the ID of the root x, ancestor$(T_x, v) \in N(v)$ and descendants$(T_x, v) \subseteq N(v)$.

Proof Sketch of Theorem 1.1. Using a network decomposition, we can replace the diameter factor in Lemma 3.1 with an $O(k \cdot \log n)$-factor (see [54, Section 5.4] for more details). Then, we apply this result for $k-1$ iterations to find $V \supseteq Q_0 \supseteq Q_1 \supseteq \cdots \supseteq Q_{k-1}$ while in each iteration we satisfy the invariants I1.1, I1.2, I2, and I3. Then Q_{k-1} has domination distance $k^2 - k$ due to invariant (I3). Using the invariants (specifically (I1.1) and (I3)), we can use our communication tools (in particular [54, Lemma 4.6]) to efficiently simulate the MIS algorithm of [25] on $G^k[Q_{k-1}]$, resulting in a $(k+1, k^2)$-ruling set of G. □

ACKNOWLEDGMENTS

Saku Peltonen is supported by the Academy of Finland, Grant 334238.

REFERENCES

[1] Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. 2021. Correlation Clustering in Data Streams. *Algorithmica* 83, 7 (2021), 1980–2017. https://doi.org/10.1007/s00453-021-00816-9

[2] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. 2012. Graph Sketches: Sparsification, Spanners, and Subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*. 5—-14. https://doi.org/10.1145/2213556.2213560

[3] Noga Alon and Sepehr Assadi. 2020. Palette Sparsification Beyond (Δ+1) Vertex Coloring. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020*. 6:1–6:22. https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2020.6

[4] Noga Alon, László Babai, and Alon Itai. 1986. A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. *Journal of Algorithms* 7, 4 (1986), 567–583.

[5] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. 2019. Sublinear Algorithms for (Δ + 1) Vertex Coloring. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, Timothy M. Chan (Ed.). SIAM, 767–786. https://doi.org/10.1137/1.9781611975482.48

[6] Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. 1989. Network decomposition and locality in distributed computation. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 364–369.

[7] Alkida Balliu, Sebastian Brandt, Manuela Fischer, Rustam Latypov, Yannic Maus, Dennis Olivetti, and Jara Uitto. 2022. Exponential Speedup Over Locality in MPC with Optimal Memory. In *Proceedings of the International Symposium on Distributed Computing (DISC)*. 9:1–9:21. https://doi.org/10.4230/LIPIcs.DISC.2022.9

[8] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. 2022. Distributed Δ-coloring plays hide-and-seek. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, Stefano Leonardi and Anupam Gupta (Eds.). ACM, 464–477. https://doi.org/10.1145/3519935.3520027

[9] Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. 2022. Distributed Lower Bounds for Ruling Sets. *SIAM J. Comput.* 51, 1 (2022), 70–115. https://doi.org/10.1137/20m1381770

[10] Philipp Bamberger, Fabian Kuhn, and Yannic Maus. 2020. Efficient Deterministic Distributed Coloring with Small Bandwidth. In *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*, Yuval Emek and Christian Cachin (Eds.). ACM, 243–252. https://doi.org/10.1145/3382734.3404504

[11] Reuven Bar-Yehuda, Keren Censor-Hillel, Yannic Maus, Shreyas Pai, and Sriram V. Pemmaraju. 2020. Distributed Approximation on Power Graphs. In *Proceedings of the 39th Symposium on Principles of Distributed Computing* (Virtual Event, Italy) *(PODC '20)*. Association for Computing Machinery, New York, NY, USA, 501–510. https://doi.org/10.1145/3382734.3405750

[12] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. 2016. The Locality of Distributed Symmetry Breaking. *J. ACM* 63, 3, Article 20 (jun 2016), 45 pages. https://doi.org/10.1145/2903137 http://arxiv.org/abs/1202.1983.

[13] Tushar Bisht, Kishore Kothapalli, and Sriram V. Pemmaraju. 2014. Brief announcement: Super-fast t-ruling sets. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, Magnús M. Halldórsson and Shlomi Dolev (Eds.). ACM, 379–381. https://doi.org/10.1145/2611462.2611512

[14] Sebastian Brandt. 2019. An Automatic Speedup Theorem for Distributed Problems. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC)*, Peter Robinson and Faith Ellen (Eds.). ACM, 379–388. https://doi.org/10.1145/3293611.3331611

[15] Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. 2017. Derandomizing Local Distributed Algorithms under Bandwidth Restrictions. In *31st Int. Symp. on Distributed Computing (DISC)*.

[16] Yi-Jun Chang, Wenzheng Li, and Seth Pettie. 2020. Distributed (Δ+1)-Coloring via Ultrafast Graph Shattering. *SIAM J. Comput.* 49, 3 (2020), 497–539. https://doi.org/10.1137/19M1249527

[17] Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. 2019. The Complexity of (Δ + 1)-Coloring in Congested Clique, Massively Parallel Computation, and Centralized Local Computation. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*. 471–480. https://doi.org/10.1145/3293611.3331607

[18] Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. 2017. Distributed algorithms for the Lovász local lemma and graph coloring. *Distributed Comput.* 30, 4 (2017), 261–280. https://doi.org/10.1007/s00446-016-0287-6

[19] Artur Czumaj, Peter Davies, and Merav Parter. 2020. Graph Sparsification for Derandomizing Massively Parallel Computation with Low Space. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures* (Virtual Event, USA) *(SPAA '20)*. Association for Computing Machinery, New York, NY, USA, 175–185. https://doi.org/10.1145/3350755.3400282

[20] Artur Czumaj, Peter Davies, and Merav Parter. 2021. Improved Deterministic (Δ+1) Coloring in Low-Space MPC. In *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen (Eds.). ACM, 469–479. https://doi.org/10.1145/3465084.3467937

[21] Janosch Deurer, Fabian Kuhn, and Yannic Maus. 2019. Deterministic Distributed Dominating Set Approximation in the CONGEST Model. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, Peter Robinson and Faith Ellen (Eds.). ACM, 94–103. https://doi.org/10.1145/3293611.3331626

[22] Michael Dinitz and Yasamin Nazari. 2020. Massively Parallel Approximate Distance Sketches. In *23rd International Conference on Principles of Distributed Systems (OPODIS 2019)*. 35:1–35:17. https://doi.org/10.4230/LIPIcs.OPODIS.2019.35

[23] Michal Dory, Orr Fischer, Seri Khoury, and Dean Leitersdorf. 2021. Constant-Round Spanners and Shortest Paths in Congested Clique and MPC. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC)*. 223–233. https://doi.org/10.1145/3465084.3467928

[24] Michael Elkin and Shaked Matar. 2019. Near-Additive Spanners In Low Polynomial Deterministic CONGEST Time. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, Peter Robinson and Faith Ellen (Eds.). ACM, 531–540. https://doi.org/10.1145/3293611.3331635

[25] Salwa Faour, Mohsen Ghaffari, Christoph Grunau, Fabian Kuhn, and Václav Rozhoň. 2022. Local Distributed Rounding: Generalized to MIS, Matching, Set Cover, and Beyond. https://doi.org/10.48550/ARXIV.2209.11651

[26] Manuela Fischer and Mohsen Ghaffari. 2017. Sublogarithmic Distributed Algorithms for Lovász Local Lemma, and the Complexity Hierarchy. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria (LIPIcs)*, Andréa W. Richa (Ed.), Vol. 91. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 18:1–18:16. https://doi.org/10.4230/LIPIcs.DISC.2017.18

[27] Manuela Fischer, Yannic Maus, and Magnús M. Halldórsson. 2022. Fast Distributed Brooks' Theorem. *CoRR* abs/2211.07606 (2022). https://doi.org/10.48550/arXiv.2211.07606 arXiv:2211.07606

[28] Sebastian Forster, Martin Grösbacher, and Tijn de Vos. 2022. An Improved Random Shift Algorithm for Spanners and Low Diameter Decompositions. In *25th International Conference on Principles of Distributed Systems (OPODIS 2021)*. 16:1–16:17. https://doi.org/10.4230/LIPIcs.OPODIS.2021.16

[29] Pierre Fraigniaud, Magnús M. Halldórsson, and Alexandre Nolin. 2020. Distributed Testing of Distance-k Colorings. In *Structural Information and Communication Complexity - 27th International Colloquium, SIROCCO 2020, Paderborn, Germany, June 29 - July 1, 2020, Proceedings (Lecture Notes in Computer Science)*, Andrea Werneck Richa and Christian Scheideler (Eds.), Vol. 12156. Springer, 275–290. https://doi.org/10.1007/978-3-030-54921-3_16

[30] Beat Gfeller and Elias Vicari. 2007. A Randomized Distributed Algorithm for the Maximal Independent Set Problem in Growth-Bounded Graphs. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing* (Portland, Oregon, USA) *(PODC '07)*. Association for Computing Machinery, New York, NY, USA, 53–60. https://doi.org/10.1145/1281100.1281111

[31] Mohsen Ghaffari. 2016. An Improved Distributed Algorithm for Maximal Independent Set. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, Robert Krauthgamer (Ed.). SIAM, 270–277. https://doi.org/10.1137/1.9781611974331.ch20

[32] Mohsen Ghaffari. 2017. Distributed MIS via All-to-All Communication. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*. 141–149. https://doi.org/10.1145/3087801.3087830

[33] Mohsen Ghaffari. 2019. Distributed Maximal Independent Set using Small Messages. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, Timothy M. Chan (Ed.). SIAM, 805–820. https://doi.org/10.1137/1.9781611975482.50

[34] Mohsen Ghaffari. 2022. Local Computation of Maximal Independent Set. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*. IEEE, 438–449. https://doi.org/10.1109/FOCS54457.2022.00049

[35] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrovic, and Ronitt Rubinfeld. 2018. Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*. 129–138. https://doi.org/10.1145/3212734.3212743

[36] Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. 2018. On Derandomizing Local Distributed Algorithms. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, Mikkel Thorup (Ed.). IEEE Computer Society, 662–673. https://doi.org/10.1109/FOCS.2018.00069

[37] Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, and Yannic Maus. 2021. Improved distributed Δ-coloring. *Distributed Comput.* 34, 4 (2021), 239–258. https://doi.org/10.1007/s00446-021-00397-4

[38] Mohsen Ghaffari and Fabian Kuhn. 2021. Deterministic Distributed Vertex Coloring: Simpler, Faster, and without Network Decomposition. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 1009–1020. https://doi.org/10.1109/FOCS52979.2021.00101

[39] Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. 2017. On the complexity of local distributed graph problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, Hamed Hatami, Pierre McKenzie, and Valerie King (Eds.). ACM, 784–797. https://doi.org/10.1145/3055399.3055471

[40] Mohsen Ghaffari and Jara Uitto. 2019. Sparsifying Distributed Algorithms with Ramifications in Massively Parallel Computation and Centralized Local Computation. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*. https://doi.org/10.1137/1.9781611975482.99

[41] Magnús M. Halldórsson, Fabian Kuhn, and Yannic Maus. 2020. Distance-2 Coloring in the CONGEST Model. In *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*, Yuval Emek and Christian Cachin (Eds.). ACM, 233–242. https://doi.org/10.1145/3382734.3405706

[42] Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Alexandre Nolin. 2020. Coloring Fast Without Learning Your Neighbors' Colors. In *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference (LIPIcs)*, Hagit Attiya (Ed.), Vol. 179. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 39:1–39:17. https://doi.org/10.4230/LIPIcs.DISC.2020.39

[43] Magnús M. Halldórsson and Alexandre Nolin. 2021. Superfast Coloring in CONGEST via Efficient Color Sampling. In *Structural Information and Communication Complexity - 28th International Colloquium, SIROCCO 2021, Wrocław, Poland, June 28 - July 1, 2021, Proceedings (Lecture Notes in Computer Science)*, Tomasz Jurdzinski and Stefan Schmid (Eds.), Vol. 12810. Springer, 68–83. https://doi.org/10.1007/978-3-030-79527-6_5

[44] James W. Hegeman, Gopal Pandurangan, Sriram V. Pemmaraju, Vivek B. Sardeshmukh, and Michele Scquizzato. 2015. Toward Optimal Bounds in the Congested Clique:Graph Connectivity and MST. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*. 91–100. https://doi.org/10.1145/2767386.2767434

[45] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. 2021. A Deterministic Almost-Tight Distributed Algorithm for Approximating Single-Source Shortest Paths. *SIAM J. Comput.* 50, 3 (2021). https://doi.org/10.1137/16M1097808

[46] Tomasz Jurdziński and Krzysztof Nowicki. 2018. MST in $O(1)$ Rounds of Congested Clique. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2620–2632. https://doi.org/10.1137/1.9781611975031.167

[47] Amos Korman, Jean-Sébastien Sereni, and Laurent Viennot. 2013. Toward more localized local algorithms: removing assumptions concerning global knowledge. *Distributed Comput.* 26, 5-6 (2013), 289–308. https://doi.org/10.1007/s00446-012-0174-8

[48] Kishore Kothapalli and Sriram V. Pemmaraju. 2012. Super-Fast 3-Ruling Sets. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India (LIPIcs)*, Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan (Eds.), Vol. 18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 136–147. https://doi.org/10.4230/LIPIcs.FSTTCS.2012.136

[49] Sven Oliver Krumke, Madhav V. Marathe, and S. S. Ravi. 2001. Models and Approximation Algorithms for Channel Assignment in Radio Networks. *Wirel. Networks* 7, 6 (2001), 575–584. https://doi.org/10.1023/A:1012311216333

[50] Fabian Kuhn, Yannic Maus, and Simon Weidner. 2018. Deterministic Distributed Ruling Sets of Line Graphs. In *Structural Information and Communication Complexity - 25th International Colloquium, SIROCCO 2018, Ma'ale HaHamisha, Israel, June 18-21, 2018, Revised Selected Papers (Lecture Notes in Computer Science)*, Zvi Lotker and Boaz Patt-Shamir (Eds.), Vol. 11085. Springer, 193–208. https://doi.org/10.1007/978-3-030-01325-7_19

[51] Christoph Lenzen and Roger Wattenhofer. 2010. Brief Announcement: Exponential Speed-up of Local Algorithms Using Non-Local Communication. In *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*. 295–296. https://doi.org/10.1145/1835698.1835772

[52] Nati Linial. 1992. Locality in Distributed Graph Algorithms. *SIAM J. Comput.* 21, 1 (1992), 193–201.

[53] Michael Luby. 1986. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM J. Comput.* 15 (11 1986), 1036–1053. https://doi.org/10.1145/22145.22146

[54] Yannic Maus, Saku Peltonen, and Jara Uitto. 2023. Distributed Symmetry Breaking on Power Graphs via Sparsification. arXiv:2302.06878 [cs.DS]

[55] Krzysztof Nowicki. 2021. A Deterministic Algorithm for the MST Problem in Constant Rounds of Congested Clique. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*. 1154—-1165. https://doi.org/10.1145/3406325.3451136

[56] David Peleg. 2000. *Distributed computing : a locality sensitive approach*. SIAM.

[57] Václav Rozhoň and Mohsen Ghaffari. 2020. Polylogarithmic-Time Deterministic Network Decomposition and Distributed Derandomization. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing* (Chicago, IL, USA) *(STOC 2020)*. Association for Computing Machinery, New York, NY, USA, 350–363. https://doi.org/10.1145/3357713.3384298

[58] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. 1995. Chernoff–Hoeffding Bounds for Applications with Limited Independence. *SIAM Journal on Discrete Mathematics* 8, 2 (1995), 223–250. https://doi.org/10.1137/S089548019223872X

[59] Johannes Schneider, Michael Elkin, and Roger Wattenhofer. 2013. Symmetry breaking depending on the chromatic number or the neighborhood growth. *Theoretical Computer Science* 509 (10 2013). https://doi.org/10.1016/j.tcs.2012.09.004

[60] Salil P. Vadhan. 2012. *Pseudorandomness*. Now Publishers. https://books.google.fi/books?id=iam4lAEACAAJ

Received 10 January 2023; accepted 26 March 2023