# Improved Network Decompositions Using Small Messages with Applications on MIS, Neighborhood Covers, and Beyond

**Mohsen Ghaffari**
ETH Zürich, Switzerland
ghaffari@inf.ethz.ch

**Julian Portmann**
ETH Zürich, Switzerland
pjulian@ethz.ch

## Abstract

Network decompositions, as introduced by Awerbuch, Luby, Goldberg, and Plotkin [FOCS'89], are one of the key algorithmic tools in distributed graph algorithms. We present an improved deterministic distributed algorithm for constructing network decompositions of power graphs using small messages, which improves upon the algorithm of Ghaffari and Kuhn [DISC'18]. In addition, we provide a randomized distributed network decomposition algorithm, based on our deterministic algorithm, with failure probability exponentially small in the input size that works with small messages as well. Compared to the previous algorithm of Elkin and Neiman [PODC'16], our algorithm achieves a better success probability at the expense of its round complexity, while giving a network decomposition of the same quality. As a consequence of the randomized algorithm for network decomposition, we get a faster randomized algorithm for computing a Maximal Independent Set, improving on a result of Ghaffari [SODA'19]. Other implications of our improved deterministic network decomposition algorithm are: a faster deterministic distributed algorithms for constructing spanners and approximations of distributed set cover, improving results of Ghaffari, and Kuhn [DISC'18] and Deurer, Kuhn, and Maus [PODC'19]; and faster a deterministic distributed algorithm for constructing neighborhood covers, resolving an open question of Elkin [SODA'04].

**2012 ACM Subject Classification** Theory of computation → Distributed algorithms

**Keywords and phrases** Distributed Graph Algorithms, Network Decomposition, Maximal Independent Set, Neighborhood Cover

**Digital Object Identifier** 10.4230/LIPIcs.DISC.2019.18

**Related Version** A full version of the paper is available at `https://arxiv.org/abs/1908.03500`.

**Funding** The authors' research is supported by the Swiss National Foundation, under project number 200021_184735.

## 1 Introduction

We present an improved deterministic distributed algorithm for constructing network decompositions of power graphs using small messages, as well as some improvements for other problems including randomized construction of maximal independent set, and deterministic construction of sparse neighborhood covers, spanners and dominating set approximation.

After introducing our model of computation, we recall the concept of network decompositions in Section 1.1 as well as a brief summary of all known distributed constructions. In Section 1.2 we present our results and in Section 1.3 we outline our methods and explain how they depart from previous approaches.


**Model.** Throughout, we work with the CONGEST model of distributed computing [27]: The communication network is abstracted as an $n$-node graph $G = (V, E)$. We use $\Delta$ to denote the maximum degree of $G$. There is one processor on each node of the network, which initially knows only its $O(\log n)$-bit identifier. Per round of synchronous communication, every node can send one $O(\log n)$-bit message to each neighbor. Note that this is enough to describe constantly many elements of the network, i.e. vertices or edges. A closely related variant is the LOCAL model [22], where we impose no restriction on the size of messages.

## 1.1 Network Decompositions

*Network decompositions* were introduced by Awerbuch et al. [8], and since then, they have turned out to be one of the key algorithmic tools in distributed algorithms for graph problems. For a given graph $G = (V, E)$, a $(c, d)$ network decomposition of it is defined as a partition of $V$ into *blocks* $V_1, \ldots, V_c$ such that each connected component of the subgraphs $G[V_i]$ has diameter at most $d$. The connected components of each block are usually called *clusters*. This notion of network decomposition is sometimes also referred to as *strong diameter* network decomposition, as we consider the diameter with respect to distances in the induced subgraphs. This is as opposed to *weak diameter* network decompositions, where distances are with respect to the base graph. Intuitively, network decompositions allow us to process graph problems in $c$ sequential stages, where in each stage we process one block, a graph that is made of low-diameter components (diameter $d$). This low-diameter simplifies the task as it opens the road for collecting either the entire topology, in the LOCAL model, or at least some coordination messages, in the CONGEST model. The key point is that the problems in different components of one block can be processed independently, as they have distance at least 1.

In many applications of network decompositions, instead of asking for the clusters to have distance at least 1, we need them to have a larger distance, at least $k$ hops for some parameter $k \geq 2$. This is crucial for applications where the problem is such that the answer in one node can impact nodes beyond its neighbors. Thus, a natural extension of network decomposition is the following: a $k$-hop separated network decomposition or decomposition of $G^k$ requires that any two nodes $u, v$ from different clusters of the same color are at distance more than $k$ in $G$. We note that clusters do not have to be connected in $G$, which means that it is a *weak diameter* decomposition of $G$.

While the authors of [8] used network decompositions to solve symmetry breaking problems, such as maximal independent set or $(\Delta + 1)$-vertex coloring, various other applications were discovered later. Examples in the LOCAL model include the computation of sparse spanners and linear-size skeletons by Dubhashi et al. [16] or distributed approximation algorithms for the graph coloring and minimum dominating set problems by Barenboim et al. [10, 11]. For the CONGEST model, Ghaffari and Kuhn [21] showed that $k$-hop separated network decompositions can be used for computing spanners and approximating minimum dominating set.

**State of the Art – Deterministic Constructions.** There are four known deterministic distributed constructions of network decompositions, successively improving either quantitatively or qualitatively [8, 20, 21, 26]. Awerbuch et al. [8] provided an algorithm for computing $(2^{O(\sqrt{\log n \log\log n})}, 2^{O(\sqrt{\log n \log\log n})})$ network decompositions of an $n$ node graph $G$ in $2^{O(\sqrt{\log n \log\log n})}$ rounds, which works in the CONGEST model. Subsequently, this was improved by Panconesi and Srinivasan [26] showing that all $2^{O(\sqrt{\log n \log\log n})}$ terms could be replaced by $2^{O(\sqrt{\log n})}$. However, their algorithm requires large messages.

For computing network decompositions with higher levels of separation, Ghaffari and Kuhn [21] gave a $k \cdot 2^{O(\sqrt{\log n \log\log n})}$ round CONGEST-model algorithm for computing a $(2^{O(\sqrt{\log n \log\log n})}, 2^{O(\sqrt{\log n \log\log n})})$ network decomposition of $G^k$, which works with small messages. Note that extending network decomposition algorithms to compute a decomposition of $G^k$ is trivial in the LOCAL model: As nodes can send messages of arbitrary size, communication on $G^k$ can be simulated in $k$ rounds of communication on $G$. Thus, with a $k$ factor overhead in the round complexity (and a $k$ factor increase in the diameter with respect to distances in $G$), we can use any LOCAL-model network decomposition algorithm to also compute $k$-hop separated decompositions.

Recently, Ghaffari [20] showed that a $(2^{O(\sqrt{\log n})}, 2^{O(\sqrt{\log n})})$ network decomposition can also be computed in $2^{O(\sqrt{\log n})}$ rounds in the CONGEST model. However, his construction cannot extend to $G^k$, which is one of the issues we address in this paper. In contrast to all previous approaches, this algorithm has the useful property, that it can handle large identifiers. This means that the length of identifiers does not influence the parameters of the resulting network decomposition.

**State of the Art – Randomized Constructions.** For randomized algorithms, there are stronger results: Linial and Saks [23] showed that $(O(\log n), O(\log n))$ network decompositions exist and gave a distributed algorithm, which finds a $(O(\log n), O(\log n))$ network decomposition in $O(\log^2 n)$ rounds, with high probability[1] (w.h.p). The construction of Linial and Saks [23] only guarantees that clusters have weak diameter $O(\log n)$. More recently, Elkin and Neiman [18] provided a randomized distributed algorithm that computes strong diameter $(O(\log n), O(\log n))$ network decomposition in $O(\log^2 n)$, w.h.p, and also works in the CONGEST model. Both of these algorithms can be easily extended to produce a $(O(\log n), O(k \log n))$ decomposition of $G^k$ in $O(k \log^2 n)$ rounds without requiring larger messages.

We remark that the fact that these algorithm succeed with probability $1 - 1/\text{poly}(n)$ prevents them from being directly used in our randomized MIS algorithm. This is because after the shattering, only components of size $N \ll n$ remain, which means that the algorithms only succeed with probability $1 - 1/\text{poly}(N)$ in computing a $(O(\log N), O(\log N))$ network decomposition.

## 1.2 Our Results

We present a deterministic distributed CONGEST-model algorithm for computing network decompositions of $G^k$:

▶ **Theorem 1.** *There is a deterministic distributed algorithm that in any $N$-node network $G$, which has $S$-bit identifiers and supports $O(S)$-bit messages for some arbitrary $S$, computes a $(g(N), g(N))$ network decomposition of $G^k$ in $kg(N) \cdot \log^* S$ rounds, for any $k$, and $g(N) = 2^{O(\sqrt{\log N})}$.*

We highlight the following three properties, whose combination is new to our algorithm and is crucial for our applications in the next subsections: (A) it is able to compute a network decomposition of $G^k$ in the CONGEST model, (B) it can handle large identifiers, and (C)

[1] As usual, we use the phrase *with high probability* to denote that an event holds with probability at least $1 - n^{-c}$ for any constant $c$, where $c$ may influence other constants.

its bounds are as good as a simulation of the algorithm of [26] on $G^k$ in the LOCAL model. More precisely, property (B) says that the size of identifiers only affects the round complexity but not the quality of the computed network decomposition.

In particular for our application to MIS, property (B) is crucial. The lack of this ability to handle large identifiers is what made previous algorithms, such as of Ghaffari and Kuhn [21], not applicable. We refer to Section 1.3 for a more in-depth explanation of these issues.

## Applications: MIS, Neighborhood Cover, and Beyond

Network decompositions have a wide range of applications and due to previous work, our new algorithms leads to an improvement for a number of problems. While some of these results are immediate, for the application to MIS, we also present a randomized algorithm for network decompositions whose failure probability is exponentially small in the input size.

**The MIS Problem.** The Maximal Independent Set Problem (MIS) asks for a set $S$ of nodes, such that no two neighboring nodes are in $S$ and moreover, for each node $v$, either $v$ or at least one of its neighbors is in $S$. It is one of the most well-studied distributed graph problems. One reason for its importance is that other fundamental graph problems, such as $(\Delta+1)$-vertex coloring, maximal matching, or 2-approximation of vertex cover reduce to it [22, 24].

Luby [24] as well as Alon, Babai and Itai [2] gave randomized distributed MIS algorithms in the CONGEST model that have round complexity $O(\log n)$. The first significant improvement over this run time was due to Barenboim, Elkin, Pettie, and Schneider [12], who gave a randomized distributed $O(\log^2 \Delta) + 2^{O(\sqrt{\log\log n})}$ round algorithm. This bound was then improved by Ghaffari [19] to $O(\log \Delta) + 2^{O(\sqrt{\log\log n})}$, which remains the state of the art. However, both these improvements do not work in the CONGEST model, as they require messages of up to $\text{poly}(\Delta, \log n)$ bits to gather certain local topologies. The only improvement upon the algorithms of [2, 24] in the CONGEST model is due to Ghaffari [20], who gave a randomized distributed algorithm that runs in $\min\{\log\Delta \cdot 2^{O(\sqrt{\log\log n})}, O(\log\Delta\cdot\log\log n) + 2^{O(\sqrt{\log\log n\cdot\log\log\log n})}\}$ rounds.

We improve this result for all values of $\Delta$ and obtain the following:

▶ **Theorem 2.** *There is a randomized distributed algorithm, with $O(\log n)$-bit messages, that computes an MIS in $O\big(\log\Delta\cdot\sqrt{\log\log n}\big) + 2^{O(\sqrt{\log\log n})}$ rounds, w.h.p.*

Apart from our improved network decomposition, this result contains a randomized algorithm, that transforms a network decomposition of $G^k$ into a decomposition of $G$ with improved parameters. This transformation works in the CONGEST model and succeeds with probability exponential in the input size, which is crucial for its application in solving MIS. For a more detailed overview, see Section 1.3.

**Neighborhood Covers and MST.** Neighborhood covers, as introduced by Awerbuch and Peleg [4] are another form of locality-preserving graph representations and closely related to network decompositions. A $s$-sparse $k$-neighborhood cover of diameter $d$ is defined as a collection of clusters $C \subseteq V$ such that (A) for each cluster $C$, we have a rooted spanning tree of $G[C]$ with diameter at most $d$, (B) each $k$-neighborhood of $G$ is completely contained in some cluster, and (C) each node of $G$ is in at most $s$ clusters. Like network decompositions, this form of graph representation has many applications in distributed computing, such as in routing [9], shortest paths [1], job scheduling and load balancing [7], or broadcast and multicast [6].

Awerbuch and Peleg [4] also gave distributed constructions for sparse neighborhood covers using messages of unbounded size, however they do not extend to the CONGEST model. More recently, Ghaffari and Kuhn [21] gave the first CONGEST model algorithm for computing sparse neighborhood covers. They showed that a $(c, d)$ network decomposition of $G^{2k}$ can be transformed into a $c$-sparse $k$-neighborhood cover of diameter $O(k \cdot d)$ in $O(c(d + k))$ rounds. Together with their $k \cdot 2^{O(\sqrt{\log n \log\log n})}$ round algorithm for computing a $(2^{O(\sqrt{\log n \log\log n})}, 2^{O(\sqrt{\log n \log\log n})})$ network decomposition of $G^k$, this yields a $k \cdot 2^{O(\sqrt{\log n \log\log n})}$ round algorithm for computing $2^{O(\sqrt{\log n \log\log n})}$-sparse $k$-neighborhood covers of diameter $k \cdot 2^{O(\sqrt{\log n \log\log n})}$.

Using our network decomposition, we improve upon the result of [21] and show that all $2^{O(\sqrt{\log n \log\log n})}$ terms can be replaced by $2^{O(\sqrt{\log n})}$. The proof is deferred to the full version of the paper.

▶ **Corollary 3.** *There is a deterministic distributed algorithm, that for every $k \geq 1$, computes a $2^{O(\sqrt{\log n})}$-sparse $k$-neighborhood cover of diameter $k \cdot 2^{O(\sqrt{\log n})}$ of and $n$-node graph $G$ in $k \cdot 2^{O(\sqrt{\log n})}$ rounds of* CONGEST.

We also resolve an open question by Elkin [17], who devised a randomized CONGEST model algorithm for minimum spanning tree, that runs in $\tilde{O}(\mu(G, \omega) + \sqrt{n})$ rounds, where $\mu(G, \omega)$ is the *MST-radius* of $G$. The MST-radius $\mu(G, \omega)$ is defined as the smallest value $t$, such that every edge not belonging to the MST of $G$ is the heaviest edge in some cycle of length at most $t$. However, the only part involving randomness is the construction of neighborhood covers. The author remarks that the only obstacle towards a deterministic algorithm is that there are no known constructions of sparse neighborhood covers in the CONGEST model. Using Corollary 3, we get a deterministic distributed CONGEST-model algorithm for computing MST in $2^{O(\sqrt{\log n})} \cdot (\mu(G, \omega) + \sqrt{n})$ rounds. For a discussion on how to use Sparse Neighborhood Covers in computing MST, we refer to [17] or the full version of this paper.

**Other Problems: Spanners and Dominating Set Approximation.** Due to previous applications of $k$-hop separated network decompositions by Ghaffari and Kuhn [21] as well as Deurer, Kuhn, and, Maus [15] we obtain the following deterministic CONGEST model algorithms: A $2^{O(\sqrt{\log n})}$ round algorithm for computing a $(2k - 1)$-stretch spanner with size $O(kn^{1+1/k} \log n)$, and a $O(\log \Delta)$-approximation algorithm for minimum dominating set in $2^{O(\sqrt{\log n})}$ rounds. For a discussion, see the full version of this paper.

## 1.3 Method Overview and Comparison with Prior Approaches

We first discuss our method for deterministic network decomposition, and then discuss our contribution to the MIS problem.

**Network Decomposition.** The general outline is shared by all known deterministic algorithms for network decomposition [8, 20, 21, 26]. This method is often referred to as recursive clustering: In every step, a number of clusters is merged to form new clusters while some other clusters are added to the output and discarded from the algorithm. However, there are several challenges in applying this approach in the CONGEST model, and even more so when aiming to compute a decomposition of $G^k$.

Let us address these challenges in two themes, (A) communication within clusters, and (B) communication between clusters: Our approach entails the fact that clusters can become disconnected in the base graph $G$, even if they are connected in $G^k$. While this means that

we have more freedom in how we merge clusters, it also requires us to take extra care to allow for intra-cluster communication. As clusters can now "overlap", a single edge of $G$ could be used by many clusters for communication. We will have a two stage process to "introduce" clusters to their neighboring clusters (in $G^k$), which will enable us to bound the "overlap" between clusters. To be more precise we will argue that any edge is used by at most $2^{O(\sqrt{\log n})}$ many clusters for communication. This allows us to have simultaneous communication in all clusters with just a $2^{O(\sqrt{\log n})}$ factor overhead.

For challenge (B), we would like to simulate communication in $G_C$ on $G$, where $G_C$ is the virtual graph obtained by contracting each cluster into a node and connecting two clusters if they contain nodes that are adjacent in $G^k$. However, this simulation is not directly possible in the CONGEST model, as nodes in $G_C$ can have degree much larger than $\Delta$, leading to congestion for communication within clusters. The solution to this problem will also be the introduction process mentioned above. Roughly speaking, we will ignore some edges from $G_C$ as well as some vertices of high degree. This allows for an efficient simulation of communication in $G_C$ on the base graph $G$.

**Maximal Independent Set.** For solving MIS, we follow the outline of the *shattering technique*, first introduced into distributed computing by Barenboim et al. [12], and also used for the MIS problem by [19, 20]. There are two parts: In the pre-shattering phase, we solve the problem for a large portion of the graph, leaving only a number of "small" connected components. Then, in the post-shattering phase, we solve the problem on the remaining parts.

In the pre-shattering phase, we use the $O(\log \Delta)$-round algorithm of Ghaffari [19], which works with just single-bit messages. Afterwards, we are left with "small" components. For now, assume that they have size[2] $O(\log n)$. By computing a network decomposition of each component, we can further simplify the problem: We go through the color classes, one by one, each time computing an MIS of the new color, that does not conflict with the MIS of the previous colors. We solve the problem by running $O(\log n)$ independent copies of Ghaffari's $O(\log \Delta)$-round randomized MIS algorithm algorithm [19], all in parallel. This parallel execution is possible in the CONGEST model because the algorithm from [19] only uses single-bit messages. With high probability (i.e. at least $1 - 1/\text{poly}(n)$), at least one of these independent runs succeeds in computing an MIS. Using the fact that we are solving the problem in a graph of low diameter, we can efficiently coordinate all nodes to find a successful run.

The main challenge is obtaining a suitable network decomposition: We could use the network decomposition algorithm from Theorem 1 and get an MIS algorithm with round complexity $\log \Delta \cdot 2^{O(\sqrt{\log \log n})}$. This only matches the previous work of Ghaffari [20]. Also, randomized algorithms for network decomposition are hard to apply, as we are computing decompositions of graphs that only contain $N = O(\log n)$ nodes. This means that the success probability of randomized algorithms for network decomposition, such as [18, 23], will only be $1 - 1/\text{poly}(N) \ll 1 - 1/\text{poly}(n)$.

We get around these issues in two steps: First, we compute a $k$-hop separated network decomposition of each component. Then, we use this network decomposition to boost the success probability of a randomized network decomposition algorithm, inspired by [14, 18, 25]. While Ghaffari [20] used a similar idea to also get an $O(\log \Delta \cdot \log \log n) + 2^{O(\sqrt{\log \log n \cdot \log \log \log n})}$

[2] They do not contain $O(\log n)$ nodes, but rather up to $O(\Delta^4 \log n)$ many vertices. However, we will see that we can efficiently cluster them into $O(\log n)$ clusters of diameter only $O(\log \log n)$.

round algorithm for MIS, we improve upon it in two ways: First, our network decomposition of $G^k$ has better bounds, and second, we use a randomized process for computing a refined network decomposition. This randomized process allows us to further reduce the number of colors needed, from $O(\log \log n)$ to $O(\sqrt{\log \log n})$.

## 1.4 Mathematical Notation

For a graph $G = (V, E)$ and two nodes $u, v \in V$, we define $d_G(u, v)$ to be the hop distance between $u$ and $v$. For a node $v \in V$ and a set $U \subset V$, $dist_G(v, U)$ is the minimum distance between $v$ and any $u \in U$. For an integer $k \geq 1$ we define the $k^{th}$ *power* $G^k = (V, E')$ of $G$ to be the graph with an edge $\{u, v\} \in E'$ whenever $d_G(u, v) \leq k$. Given a node $v \in V$, we define $N_{G,k} := \{u \in V : d_G(u, v) \leq k\}$ to be the *k-hop neighborhood* of $v$.

For two integers $\alpha \geq 1$ and $\beta \geq 0$ and a node set $B \subseteq V$, we call $B^* \subseteq B$ a $(\alpha, \beta)$-ruling set of $G$ w.r.t. $B$ if (A) for any two nodes $u, v \in B^*$ we have $d_G(u, v) \geq \alpha$, and (B) $\forall u \in B \setminus B^*$, there is a node $v \in B^*$ such that $d_G(u, v) \leq \beta$. If $B = V$, $B^*$ is simply called an $(\alpha, \beta)$ ruling set of $G$.

# 2 Network Decomposition

In this section we describe our algorithm for computing network decompositions of power graphs in the CONGEST model, as outlined in Theorem 1. It matches the bounds of the algorithm by Panconesi and Srinivasan [26], but improves upon it in three aspects that are crucial to our applications: our algorithm works in the CONGEST model, can tolerate large identifiers and is able to produce a network decomposition of $G^k$. While the first two properties were already achieved by Ghaffari [20], the third property is new to our approach.

Note that it is trivial to achieve such a decomposition using messages of unbounded size, by just simulating communication in $G^k$ on $G$ (with a $k$ factor overhead). In the CONGEST model this idea is not directly applicable and presents two challenges: (A) how do we deal with clusters being disconnected in the base graph and (B) how do we get around simulating all communication in $G^k$ on $G$? For the first issue we will bound the number of clusters that are overlapping. For the second issue we will see that not all communication is necessary.

Before proceeding to the algorithm, we restate Theorem 1 in slightly more detail:

▶ **Theorem 4.** *There is a deterministic distributed algorithm that in any $N$-node network $G$, which has $S$-bit identifiers and supports $O(S)$-bit messages for some arbitrary $S$, computes a $(g(N), g(N))$ network decomposition of $G^k$ in $kg(N) \cdot \log^* S$ rounds, for any $k$ and $g(N) = 2^{O(\sqrt{\log N})}$. Additionally we can simulate one round of communication within clusters of $G^k$ in $k2^{O(\sqrt{\log N})}$ rounds of communication on $G$.*

▶ Remark 5. If we initially have $N$ clusters, each with an $S$-bit center identifier and with radius at most $r$, the algorithm of Theorem 4 computes a $(g(N), rg(N))$ network decomposition of $G^k$ in $krg(N) \cdot \log^* S$ rounds.

**Proof of Theorem 4.** We first note that the recursive nature of the algorithm makes it directly applicable to use with an initial clustering, as described in Remark 5.

**Overall Structure.** The algorithm consists of phases $i = 1, \ldots, \sqrt{\log N}$, each of which runs in $k \cdot 2^{O(\sqrt{\log N})} \cdot \log^* S$ rounds. During each phase, the (remaining) vertices are partitioned into vertex-disjoint clusters. Each cluster has one center node (which will be the identifier of the cluster), as well as a tree rooted at the center that spans all vertices of this cluster

(and potentially also contains vertices of other clusters). However we have that any two nodes of the cluster are connected by a path of length at most $k$ in this tree. Note that both edges and vertices of $G$ can be included in multiple trees.

Initially, every node is its own cluster. During each phase some clusters join each other to form some new clusters, while the other clusters are colored and removed from the algorithm. Let $d = 2^{O(\sqrt{\log N})}$. We maintain the following invariants during the $i^{th}$ phase:

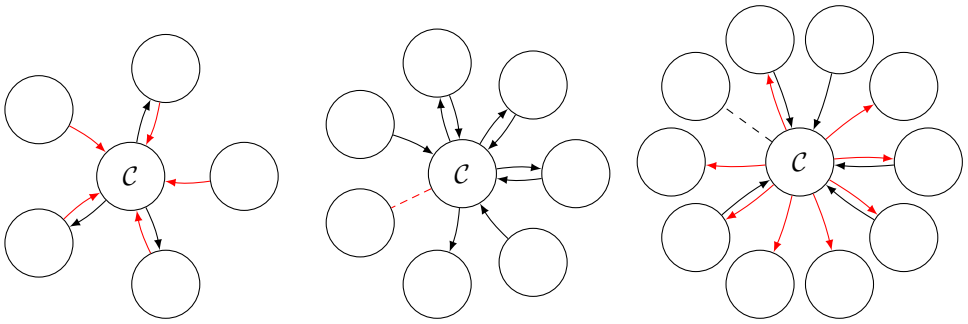
- **(A)** We have at most $n/d^i$ clusters.
- **(B)** The radius of each cluster is at most $h_i = (O(1))^i \leq 2^{O(\sqrt{\log N})}$ in $G^k$, which means its radius is at most $k \cdot 2^{O(\sqrt{\log N})}$ in $G$.
- **(C)** Each edge $e$ is part of at most $i \cdot 13d^3$ spanning trees, which is always at most $2^{O(\sqrt{\log N})}$.

Note that for $i = 0$, these invariants are trivially fulfilled. The first invariant ensures that after $\sqrt{\log N}$ phases, there is at most one cluster left, which we can then color with one color and finish the algorithm. The second invariant means that cluster radii remain small enough. Invariants (B) and (C) together imply that we can perform $2^{O(\sqrt{\log N})} \cdot \log^* S$ iterations of broadcast and convergecast in each cluster within each phase. This is because we can simulate a round of communication along an edge in the spanning tree of $G^k$ within $k \cdot 2^{O(\sqrt{\log N})}$ rounds of communication on $G$.

**Informal Outline of each phase.** We start out with a set of (old) clusters and will merge some of them into new clusters, while we color the remaining ones and add them to the resulting decomposition. In a first step, each cluster $\mathcal{C}$ will try to learn its neighboring clusters. If $\mathcal{C}$ has more than $4d^2$ neighbors, we call $\mathcal{C}$ *marked*. If we now consider the graph $\mathcal{G}$ induced by all non-marked clusters, it has maximum degree $\Delta_{\mathcal{G}} \leq 4d^2$. This allows us to simulate communication within $\mathcal{G}$ in the underlying network, with about a $4d^2$ overhead in the round complexity (ignoring cluster diameters). We will use this fact to find a well-separated set $\mathcal{C}^*$ in $\mathcal{G}$. All clusters from $\mathcal{C}^*$, together with the marked clusters will now form the centers of new clusters. Then all old clusters that have a neighboring center join this center to form a new cluster. Intuitively, as all new cluster centers have high degree, there cannot be a lot of them. What we are now left with is a set of low-degree clusters that are not part of any newly formed cluster. We can now just color these remaining clusters, using standard coloring techniques, and add them to the final output. As the degrees are low, the number of required colors is also low.

**Building a small in-degree virtual graph $H$.** Call two clusters $\mathcal{C}$ and $\mathcal{C}'$ neighboring if they contain vertices $v \in \mathcal{C}$ and $v' \in \mathcal{C}'$ such that $v$ and $v'$ are at distance at most $k$ in $G$. This means that $v$ and $v'$ are neighbors in $G^k$. Similarly, a node $v$ and a cluster $\mathcal{C}$ are called neighboring if there is some $u \in \mathcal{C}$ such that $u$ is at distance at most $k$ from $v$.

Now we want every cluster to learn about up to $2d$ many neighboring clusters. More precisely, a cluster that has less than $2d$ neighbors should learn about all of its neighbors. If it has more than $2d$ neighbors, it learns about some $2d$ of them. We can do so in $O(k \cdot d)$ rounds: Every node starts a broadcast, sending the identifier of its current cluster to all neighbors. Then, over $(2d+1) \cdot (k-1)$ rounds, every node $v$ forwards up to $2d+1$ different such messages about clusters of distance up to $(k-1)$ from $v$. This way, if node has at most $2d$ neighboring clusters it learns about all of them and if there are more, it learns about at least $2d$ many, of which it picks some $2d$ many arbitrarily. Within clusters, the nodes convergecast at most $2d$ identifiers to the center node. This is possible in $O((d + k \cdot h_i) \cdot 2^{O(\sqrt{\log N})}) = k \cdot 2^{O(\sqrt{\log N})}$ rounds, as invariant (C) states that at most $2^{O(\sqrt{\log N})}$ clusters overlap. Thus, a $2^{O(\sqrt{\log n})}$ round overhead is enough to allow all clusters to perform a convergecast at the same time.




**(a)** Low-Degree. **(b)** High-Degree. **(c)** Marked.

**Figure 1** Different states of a cluster $\mathcal{C}$ in the virtual graph $H$, where clusters are vertices and an edge $\mathcal{C} \to \mathcal{C}'$ means that the center of $\mathcal{C}'$ received the identifier of $\mathcal{C}$. Dashed lines indicate that clusters are neighboring, but neither center received the ID of the other center.

This process creates a directed virtual graph $H$ among the clusters, where an edge $\mathcal{C} \to \mathcal{C}'$ indicates that the center of $\mathcal{C}'$ received the identifier of $\mathcal{C}$. We call a cluster *high-degree* if it has at least $2d$ neighboring clusters, and *low-degree* otherwise, see Figure 1. Notice that a low-degree clusters has all neighboring clusters as incoming edges in $H$. Also, a high-degree cluster has at least $2d$ incoming edges.

**Making $H$ undirected with small degrees.** One problem is that $H$ is a directed graph with possibly large out-degrees, while we would like to have an undirected graph with small degrees. Additionally we would like to keep the fact that all low degree clusters are adjacent to all their neighboring clusters in this virtual graph. For that, we first *mark* clusters of extremely high out-degree as follows: we reverse the communication direction of the previous paragraph, but instead of sending just one message per round along each edge, we send up to $4d^2$ messages. This increases the number of rounds by at most a $4d^2$ factor. This way, if a message from some cluster $\mathcal{C}$ was sent along an edge in the previous phase, we send up to $4d^2$ messages in the opposite direction, all from clusters that received the identifier of $\mathcal{C}$. If more clusters received the identifier of $\mathcal{C}$, we just inform $\mathcal{C}$ that it will be marked. This can be done within $O(k \cdot d^3) = k \cdot 2^{O(\sqrt{\log N})}$ rounds, as every round from the previous paragraph now takes $4d^2$ as long. Also, at most $(2d+1) \cdot 4d^2 \leq 12d^3$ many messages are sent along each edge. As in the previous paragraph, we can now convergecast the identifiers of at most $4d^2$ outgoing neighbors in $O((d^2 + k \cdot h_i) \cdot 2^{O(\sqrt{\log N})}) = k \cdot 2^{O(\sqrt{\log N})}$ rounds to the cluster centers, marking them the same way as before.

Now, we temporarily remove marked clusters from $H$; we later discuss how to deal with them. Note that there are at most $\frac{n}{2d^{i+1}}$ many marked clusters. This is because each cluster has in-degree at most $2d$, which means at most a $1/(2d)$ fraction of clusters can have out-degree exceeding $4d^2$. This is at most $\frac{n}{d^i} \cdot \frac{1}{2d}$ many clusters, by invariant (A). We now have an undirected virtual graph on the clusters, which has degree at most $4d^2$.

**Computing a Maximal 2-Independent Set in $H$.** $H$ has now degree at most $4d^2$, but we need an additional fact to ensure that we can simulate the communication along $H$ in $G$: Every edge is part of at most $12d^3 = 2^{O(\sqrt{\log N})}$ edges of $H$. This is because we can think of every message in the previous phase as trying to establish an edge between two clusters $\mathcal{C}$, $\mathcal{C}'$ in $H$. Such an edge is only established if a message from $\mathcal{C}$ actually reaches $\mathcal{C}'$. As every edge in $G$ only forwarded $12d^3$ many such messages, it can only be part of as many edges in $H$.

This means that we can now simulate one round of CONGEST model on $H$ in $O(k \cdot d^3 + (d^2 + k \cdot h_i) \cdot 2^{O(\sqrt{\log n})}) = k \cdot d^3 \cdot 2^{O(\sqrt{\log N})}$ rounds of the base graph. This is because every edge is additionally only part of at most $2^{O(\sqrt{\log N})}$ clusters, by our invariant (C). Using that, we first compute a coloring of $H^2$, hence ensuring that any two clusters that are within 2 hops in $H$ have different colors. That can be done in $O(d^8 \log^* S \cdot (k \cdot d^3 \cdot 2^{O(\sqrt{\log n})}) = O(d^{11} \log^* S \cdot 2^{O(\sqrt{\log N})})$ rounds, using Linial's algorithm [22], which runs in $O(\Delta_H^2 \log^* S)$ rounds, as $H^2$ has maximum degree $\Delta_H = O(d^4)$. Then, we compute a maximal 2-independent set $C^*$ of high-degree clusters (this is the definition of high degree mentioned above, which is with respect to the neighborhood of clusters in $G^k$). Here, 2-independent set means that no two clusters in $C^*$ should share a common neighboring cluster in $H$. We can do so by going through all colors one by one, adding clusters to $C^*$ that do not already have a cluster from $C^*$ within distance two in $H$. Any $\mathcal{C}$ that has a $\mathcal{C}' \in C^*$ within its 2-cluster-hops joins the new cluster being formed at the center of $\mathcal{C}'$. As all high-degree clusters not in $C^*$ must have a neighbor in $C^*$ within 2-cluster hops, all high-degree clusters are part of a newly formed cluster.

**Forming new clusters.** Each high-degree cluster $\mathcal{C}' \in C^*$ has two cases: (I) either none of the neighboring clusters of $\mathcal{C}'$ were marked, in which case all of them will join the new cluster being formed by $\mathcal{C}'$. This means that the new cluster contains at least $2d$ many old clusters. Thus, there are at most $\frac{n}{d^i} \cdot \frac{1}{2d}$ many such new clusters. (II) at least one of the neighboring clusters of $\mathcal{C}'$ was marked. In this case, after $\mathcal{C}'$ accepts the clusters that want to join with it, $\mathcal{C}'$ picks one of its marked neighbors and joins a new cluster centered at that marked cluster. To make clusters learn about neighbors, use $k$ rounds of flooding, initiated at all nodes of marked clusters. This way, we have to send the identifier of at most one marked cluster along each edge, to ensure that all clusters know if they have a marked neighbor. Since there are at most $\frac{n}{2d^{i+1}}$ many marked clusters, the number of the new clusters of this kind is also at most $\frac{n}{2d^{i+1}}$.

**Proving the inductive invariants.** By the previous paragraph, we have at most $\frac{n}{d^{i+1}}$ many new clusters, proving invariant (A). Regarding invariant (B), first notice that each new cluster that we form is made of some of the previous clusters, all of which were within $O(1)$ cluster hops (w.r.t. distances in $G^k$) of the center of the merge (either in $C^*$ or a marked cluster). Hence, the maximum cluster radius grows by at most a factor of $O(1)$, which shows that each cluster radius in phase $i$ is at most $(O(1))^{i+1}$ in $G^k$.

For invariant (C), we have already argued that due to merges between non marked clusters, every edge is used by at most $12d^3$ many additional clusters, as these merges only happen along edges of $H$. For the merging centered at a marked cluster $\mathcal{C}$, we have that if an edge $e$ is part of a path that informed some other clusters about $\mathcal{C}$, they might merge with $\mathcal{C}$ or some other marked cluster. In either case, $e$ is included in at most one additional cluster. As all of those will merge to the same cluster, we have that $e$ is used by at most $12d^3 + 1 \leq 13d^3$ additional clusters. By induction, there are at most $i \cdot 13d^3 + 13d^3 = (i+1) \cdot 13d^3$ many spanning trees that include a given edge.

**Coloring low-degree old clusters that remain.** Finally, we are left with only low-degree clusters, as we have included all high-degree clusters in a new cluster. This means that all remaining clusters have at most $2d$ neighboring clusters. We can color these cluster using $O(d^2)$ colors by applying Linial's algorithm [22] which runs in $O(\log^* S)$ rounds of CONGEST on top of the cluster graph, that is, in $2^{O(\sqrt{\log N})} \log^* S$ rounds of the base graph. As we use different colors for each phase, we get a total of $\sqrt{\log N} \cdot O(d)^2 = 2^{O(\sqrt{\log N})}$ colors. ◀

▶ Remark 6. Even though edges can be part of up to $2^{O(\sqrt{\log N})}$ many clusters, per color class they can be included in at most one cluster. This is because otherwise we would have two clusters with the same color that are at distance less than $k$.

## 3 Implications on MIS

In this section we present our improved algorithm for computing maximal independent set in the CONGEST model. In particular, we prove the following:

▶ **Theorem 7.** *There is a randomized distributed algorithm, with $O(\log n)$-bit messages, that computes an MIS in $O(\log \Delta \cdot \sqrt{\log \log n}) + 2^{O(\sqrt{\log \log n})}$ rounds, w.h.p.*

We will use the following results about Ghaffari's algorithm for computing a (maximal) independent set [19].

▶ **Theorem 8** ([19]). *For each node $v$, the probability that $v$ has not made its decision within the first $O(\log \deg(v) + \log 1/\epsilon)$ rounds, where $\deg(v)$ denotes $v$'s degree at the start of the algorithm, is at most $\epsilon$.*

▶ **Lemma 9** ([19]). *Let $B$ be the set of nodes remaining undecided after $\Theta(\log \Delta)$ rounds. Then, with high probability, we have the following properties:*

**(P1)** *There is no $(G^4)$-independent $(G^9)$-connected subset $S \subseteq B$ s.t. $|S| \geq \log_\Delta n$. This means that $S$ is an independent set in $G^4$ and induces a connected subgraph in $G^9$.*

**(P2)** *All connected components of $G[B]$, that is the subgraph of $G$ induced by nodes in $B$, have each at most $O(\log_\Delta n \cdot \Delta^4)$ nodes.*

The statement of Lemma 9 is known as a *shattering* guarantee, which is used in various (distributed) algorithms, see e.g. [3, 12, 13]. Intuitively, this means that after $O(\log \Delta)$ rounds of the algorithm, the components induced by undecided nodes are "small", or more precisely in this case: they do not contain a large 5-independent set. If we allowed for messages of unbounded size, we could just think of the remaining components as graphs of size $O(\log n)$, and use traditional algorithms to solve the problem. However, as we restrict messages to $O(\log n)$-bits, we will need some additional ideas.

We will also use the following ruling set algorithm of Ghaffari [20]:

▶ **Lemma 10** ([20]). *There is a randomized distributed algorithm in the CONGEST model that, in any network $H = (V, E)$ with at most $n$ vertices, and for any $B' \subset V$ and for any integer $k \geq 1$, with high probability, computes a $(k, 10k^2 \log \log n)$ ruling set $B^* \subseteq B'$, with respect to distances in $H$, in $O(k^2 \log \log n)$ rounds.*

**Algorithm Outline.** Combining these results, we can obtain the following: First, we run the algorithm of Ghaffari [19] for $O(\log \Delta)$ rounds, which results in a state described by Lemma 9. Then, we compute a $(5, O(\log \log n)$ ruling set of each remaining component, using Lemma 10. This ruling set induces a clustering, where each node is vertex is clustered to its closest node from the ruling set. By property (P1) of Lemma 9, this yields $N = O(\log n)$ clusters per component of the remaining graph. Let us call one such cluster a *meta-node*, and note that it has diameter $r = O(\log \log n)$. For the remainder of this section, let $H$ be the graph, where the vertex set are all meta-nodes and where two clusters are connected if they contain two adjacent nodes. Then, we compute a network decomposition of $H$ into *super-clusters*. Going through the color classes of this decomposition, one by one, we compute an MIS of each super-cluster. We use the fact that these are graphs of low-diameter to amplify the success probability of a randomized algorithm.

The main challenge will be to compute a suitable network decomposition of $H$. In particular, we aim to compute a network decomposition using as few colors as possible. In Lemma 12 we obtain such a decomposition, enabling us to prove Theorem 11. We strengthen this result in Lemma 13, using a randomized approach, to get a decomposition that will be sufficient to prove Theorem 7.

## 3.1 First Approach: Slower but Simpler

In this section, we prove the following Theorem. While it give a slower runtime than Theorem 7, it is still faster than previous algorithms.

▶ **Theorem 11.** *There is a randomized distributed algorithm, with $O(\log n)$-bit messages, that computes an MIS in $O(\log \Delta \cdot \log\log n) + 2^{O(\sqrt{\log\log n})}$ rounds, w.h.p.*

To prove Theorem 11, we use the following algorithm for network decomposition:

▶ **Lemma 12.** *Let $H$ be the $N = O(\log n)$-node graph as described in the outline. There is a deterministic distributed algorithm, with $O(\log n)$-bit messages, that computes a $(O(\log\log n), 2^{O(\sqrt{\log\log n})})$ network decomposition of $H$ into* super-clusters *in $2^{O(\sqrt{\log\log n})}$ rounds of communication on $G$.*

**Proof.** We will first argue that we can compute a network decomposition of $H^k$. Then we refine this decomposition into a decomposition of $H$, while reducing the number of colors used. We will do so by using a *ball growing* process, inspired by [4, 5, 23]: Here, a ball is just a set of vertices with low diameter. Starting from balls being clusters of one color, we grow each of them hop by hop in $H$ until it contains enough meta-nodes. We use the fact that we have a decomposition of $H^K$ (for $K$ large enough) to argue that different clusters can operate independently.

**Intermediate Network Decomposition.** First, we will compute a network decomposition of $H^K$ for $K = \Theta(\log\log n)$. In $G$, every meta-node of $H$ is a cluster of diameter $O(\log\log n)$, so we compute such a network decomposition of $H^K$ by computing a network decomposition of $G^k$ for $k = \Theta((\log\log n)^2)$: Using the initial partition as a starting point, we get a $(2^{O(\sqrt{\log\log n})}, 2^{O(\sqrt{\log\log n})})$ network decomposition of $G^k$ by Remark 5. Note that by design of the algorithm, all nodes of such an initial cluster will end up in the same cluster as well. As these initial clusters have diameter $O(\log\log n)$ and we set $k = \Theta((\log\log n)^2)$, two clusters are at distance $\Theta(\log\log n)$ in $H$.

Now we have an intermediate $(2^{O(\sqrt{\log N})}, 2^{O(\sqrt{\log N})})$ network decomposition of $H^K$. That is, every two meta-nodes from different clusters of the same color have distance at least $K = O(\log N)$ in $H$. We can simulate one round of communication within clusters of $H$ in $2^{O(\sqrt{\log n})}$ rounds in $G$.

**One Step of Ball Growing.** The next step is to refine this intermediate decomposition to compute a new decomposition of $H$ with the properties from Lemma 12. To do so we use the following ball growing process: In each step, we add some meta-nodes to a new super-cluster, while deactivating another set of meta-nodes. Initially, all meta-nodes are active.

More precisely, the $i^{th}$ step is as follows: Starting from all clusters of color $i$, we initiate a ball growing process. Note that we only consider meta-nodes that are still active and not yet part of a super cluster. We call a meta-node of $H$ a *boundary* for this ball if at least one of its neighbors is in a different ball. We call a ball *good* if there are less boundary than non-boundary noes.

Initially, a ball is a cluster of color $i$ (or rather its remaining meta-nodes). If the ball is not good, we grow it by one hop in $H$. We can do this along edges of $H$, which are also edges in $G$. In that case, by definition of a good ball, this ball grows by at least a 2 factor in terms of its number of meta-nodes. We repeat this until we reach a good ball. That happens within $\log N$ steps of growth as otherwise the ball would have more than $2^{\log N} = N$ meta-nodes of $G$, which is not possible. Notice that each step of growth can be performed in $2^{O(\sqrt{\log N})}$ rounds: We aggregate the number of boundary and non-boundary meta-nodes at the center, which then decides whether to stop or continue the process. Once a ball is good, we deactivate its boundary meta-nodes for this phase. The non-boundary meta-nodes of each ball are joined together as one super-cluster of the output-decomposition. In each step of the ball growing, the radius of these super-clusters increases by at most one and thus stays $2^{O(\sqrt{\log N})}$ (which is also true in $G$, as every meta-node has radius $O(\log N)$). Additionally, balls can grow along each edge at most once, meaning that every edge gets included in at most one additional super-cluster on top of the previous clusters it was included in. Together with the diameter staying $2^{O(\sqrt{\log N})}$, this ensures property (B). We note that the balls that start from different clusters of color $i$ in the intermediate network decomposition can grow simultaneously. They will never reach each other, as originally they were separated by at least $\Omega(\log N)$ hops in $H$ and each ball grows at most $\log N$ hops.

**The Full Algorithm.** We perform $\log N$ phases: In each phase, we perform $2^{O(\sqrt{\log N})}$ steps of ball growing, one step for each color class of the intermediate network decomposition. Once a phase is finished, we reactive all unclustered meta-nodes and move on to the next phase. Notice that in each phase, at least half of the remaining meta-nodes join a new super-cluster: we only deactivate boundary-nodes and further only do so, whenever we add at least as many nodes to a new super cluster. Thus after $\log N$ phases, the graph must be empty. In total, we spend $\log N \cdot 2^{O(\sqrt{\log N})} \cdot 2^{O(\sqrt{\log N})} = 2^{O(\sqrt{\log N})}$ rounds. As we always deactivate the boundary nodes, super clusters are non-adjacent, which shows property (A). For the number of colors, we use only one color per phase, and as there are $\log N = O(\log \log n)$ phases, we use as many colors. ◀

We can now use this decomposition of $G$, to compute a maximal independent set:

**Proof of Theorem 11.** As a first step, we compute $H$ as described before, by running Ghaffari's algorithm [19] for $O(\log \Delta)$ rounds, and computing a clustering in the remaining parts of the graph. Then, we find a $(O(\log \log n), 2^{O(\sqrt{\log \log n})})$ decomposition of $H$, using Lemma 12.

For computing the MIS we proceed as follows: We work through each of the $O(\log \log n)$ color classes, spending $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$ rounds per color class. In one color class, we can find one MIS per super cluster, as super clusters of the same color are non-adjacent.

In every step, all nodes of the active super clusters execute $O(\log n)$ parallel executions of the algorithm of Ghaffari [19], as reviewed in Theorem 8. This can be done without any overhead, as every single execution only uses one-bit messages. This super cluster contains $O(\Delta^4 \log n)$ regular nodes, by property (P1) of Lemma 9. Running this algorithm for $O(\log(\Delta^4 \log n)) = O(\log \Delta + \log \log n)$ rounds, we find an MIS with probability at least $1 - 1/\text{poly}(\Delta^4 \log n)$. Since all $O(\log n)$ parallel executions are independent, the probability that none of them succeeds is at most $1/\text{poly}(n)$.

Now we just need to find a run that was successful. For this we use the network decomposition we obtained from Lemma 12. First, each node $v$ performs a local check for all runs, by making sure that either $v$ is in the MIS and none of its neighbors is, or that $v$ is not in the MIS, but at least one of its neighbors is. This can again be done with just

one-bit messages. Then, we can convergecast these local checks towards the cluster centers in $2^{O(\sqrt{\log\log n})}$ rounds. These centers can pick the first successful run and inform all nodes of their cluster in $2^{O(\sqrt{\log\log n})}$ rounds. We remove all nodes that are in the MIS of the super cluster, together with all their neighbors (in the base graph). After this, we move on to the next color.

In total, we spend $O(\log\log n)\cdot(O(\log\Delta+\log\log n)+2^{O(\sqrt{\log\log n})} = O(\log\Delta\cdot\log\log n)+2^{O(\sqrt{\log\log n})})$ rounds and find an MIS with high probability. ◀

## 3.2 Second Approach: Faster

To improve the runtime compared to Theorem 11 we need to obtain a network decomposition using fewer colors. Instead of a sequential ball growing process, we will perform a randomized *ball carving*, similar to Elkin and Neiman [18]. As this decomposition will be used on small components of the graph, we need to ensure that we still succeed with probability $1-1/\text{poly}(n)$ even on components with much less than $n$ vertices. We will use a similar idea as in the proof of Theorem 11, namely that we run many random processes in parallel and use a previously computed network decomposition to find a run that was successful. However, defining the right measure of success and identifying a successful run both require much more care than in algorithm for MIS. The resulting algorithm is formalized in the following Lemma:

▶ **Lemma 13.** *Let $H$ be the $N=O(\log n)$-node graph as described in the algorithm outline. There is a randomized distributed algorithm, with $O(\log n)$ bit messages, that computes a strong diameter $(O(\sqrt{\log\log n}), 2^{O(\sqrt{\log\log n})})$ network decomposition of $H$ in $2^{O(\sqrt{\log\log n})}$ rounds, with probability $1-1/\text{poly}(n)$.*

Below, we provide a proof outline, for the complete proof, see the full version of this paper.

**Proof Sketch.** The general idea is the same as in the proof of Lemma 12: As we can compute a network decomposition of $H^k$ by Theorem 4, we want to use the fact that clusters are separated by $k$ hops, to get a decomposition of $H$ which uses fewer colors. Instead of starting from the initial network decomposition, we restart from scratch, only using the initial network decomposition for amplifying success probabilities.

Let us first quickly recap what *ball carving* is and how it is used in constructing network decompositions: We randomly select a number balls, where each ball $B$ is a set of nodes with low diameter. We call a node a *boundary* of a ball $B$ if it has at least one neighbor that is not in $B$. For every ball $B$, we define one cluster $C$ containing all non-boundary nodes of $B$ and remove all such clusters $C$ from the graph. This concludes one step of ball carving.

To create a network decomposition we apply this process recursively on the remaining graph, until it is empty. As we ignore boundary-nodes, the clusters formed in every step are non-adjacent, which means we can use a single color for each recursive step. In order to obtain good bounds for the resulting network decomposition, we need the selected balls to have the following two properties: (A) their diameter is low, and (B) the number of boundary nodes is be small. Intuitively, property (A) means that the resulting decomposition has low diameter, while property (B) means that it uses a small number of colors. We call a ball carving *successful* if properties (A) and (B) are fulfilled.

Given a suitable algorithm for selecting balls, see e.g. [18, 23], this process can be used to compute a network decomposition with much better bounds than what we can compute deterministically. However, the success probability of such selection algorithms is too low in our setting: We only have $N=O(\log n)$ nodes, which means that the success probability is just $1-1/\text{poly}(N) = 1-1/\text{poly}(\log n)$.

We use the same idea as in the proof of Theorem 11 to amplify this probability: Instead of executing a ball carving on all nodes, we only consider one color class of the initial network decomposition at a time. Next, we execute $O(\log n)$ many parallel attempts of ball carving. This means that with probability $1 - 1/\text{poly}(n)$ at least one attempt was successful. To find such a successful attempt, clusters of the same color can operate independently, as they are separated by $k$ hops. To decide if an attempt was successful, we compute the number of boundary and non-boundary nodes for each ball and aggregate all these numbers at the center node of each cluster. Additionally, we aggregate the maximum diameter amongst all balls. Based on this information, the centers can decide if a attempt was successful. By performing this check for all executions in parallel, we can thus find a successful attempt. Then, we move on to the next color class. ◀

As in the proof of Theorem 11, we can now use this network decomposition to compute a maximal independent set. Since the two proofs are identical, we provide a brief outline:

**Proof Sketch of Theorem 7.** We can use the Algorithm of [19] to compute an independent set, leaving only components of small diameter in the remaining graph. On these remaining components we compute a $(O(\sqrt{\log\log n}), 2^{O(\sqrt{\log\log n})})$ network decomposition by applying Lemma 13. Then we go through all colors of this network decomposition one by one. Per color class we spend $O(\log\Delta + 2^{O(\sqrt{\log\log n})})$ rounds: We run $O(\log n)$ parallel randomized MIS algorithms for $O(\log\Delta)$ rounds, and use the fact that clusters have radius $2^{O(\sqrt{\log\log n})}$ to pick a successful run in time proportional to this radius. Then we move on to the next color class, removing all nodes that have a neighbor in the computed independent set. In total, this takes $O(\log\Delta) + \sqrt{\log\log n} \cdot 2^{O(\sqrt{\log\log n})}$ rounds to compute an MIS. ◀

## References


1 Yehuda Afek and Moty Ricklin. Sparser: A paradigm for running distributed algorithms. *Journal of Algorithms*, 14(2):316–328, 1993.

2 Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of algorithms*, 7(4):567–583, 1986.

3 Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1132–1139. Society for Industrial and Applied Mathematics, 2012.

4 B Awerbuch and D Peleg. Sparse partitions. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 503–513. IEEE, 1990.

5 Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM (JACM)*, 32(4):804–823, 1985.

6 Baruch Awerbuch. Efficient broadcast and light-weighted spanners. *manuscript*, 1992.

7 Baruch Awerbuch, Shay Kutten, and David Peleg. Online load balancing in a distributed network. In *Proc. 24th ACM Symp. on Theory of Comput*, pages 571–580, 1992.

8 Baruch Awerbuch, Michael Luby, Andrew V Goldberg, and Serge A Plotkin. Network decomposition and locality in distributed computation. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 364–369. IEEE, 1989.

9 Baruch Awerbuch and David Peleg. Routing with polynomial communication-space srade-ff. *SIAM Journal on Discrete Mathematics*, 5(2):151–162, 1992.

10 Leonid Barenboim. On the locality of some NP-complete problems. In *International Colloquium on Automata, Languages, and Programming*, pages 403–415. Springer, 2012.

11 Leonid Barenboim, Michael Elkin, and Cyril Gavoille. A fast network-decomposition algorithm and its applications to constant-time distributed computation. *Theoretical Computer Science*, 751:2–23, 2018.

12 Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *Journal of the ACM (JACM)*, 63(3):20, 2016.

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

14 Guy E Blelloch, Anupam Gupta, Ioannis Koutis, Gary L Miller, Richard Peng, and Kanat Tangwongsan. Nearly-linear work parallel sdd solvers, low-diameter decomposition, and low-stretch subgraphs. *Theory of Computing Systems*, 55(3):521–554, 2014.

15 Janosch Deurer, Fabian Kuhn, and Yannic Maus. Deterministic Distributed Dominating Set Approximation in the CONGEST Model. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, page to appear. ACM, 2019.

16 Devdatt Dubhashi, Alessandro Mei, Alessandro Panconesi, Jaikumar Radhakrishnan, and Aravind Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. *Journal of Computer and System Sciences*, 71(4):467–479, 2005.

17 Michael Elkin. A faster distributed protocol for constructing a minimum spanning tree. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 359–368. Society for Industrial and Applied Mathematics, 2004.

18 Michael Elkin and Ofer Neiman. Distributed strong diameter network decomposition. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 211–216. ACM, 2016.

19 Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 270–277. Society for Industrial and Applied Mathematics, 2016.

20 Mohsen Ghaffari. Distributed Maximal Independent Set using Small Messages. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 805–820. SIAM, 2019.

21 Mohsen Ghaffari and Fabian Kuhn. Derandomizing distributed algorithms with small messages: Spanners and dominating set. In *32nd International Symposium on Distributed Computing (DISC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

22 Nathan Linial. Distributive graph algorithms global solutions from local data. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 331–335. IEEE, 1987.

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

24 Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986.

25 Gary L Miller, Richard Peng, and Shen Chen Xu. Parallel graph decompositions using random shifts. In *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures*, pages 196–203. ACM, 2013.

26 Alessandro Panconesi and Aravind Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 581–592. ACM, 1992.

27 David Peleg. Distributed computing. *SIAM Monographs on discrete mathematics and applications*, 5, 2000.