Algorithms for the Minimum Dominating Set Problem in Bounded Arboricity Graphs: Simpler, Faster, and Combinatorial

Adir Morgan ✉
Tel Aviv University, Israel

Shay Solomon ✉
Tel Aviv University, Israel

Nicole Wein ✉
MIT, Cambridge, MA, USA

Abstract

We revisit the minimum dominating set problem on graphs with arboricity bounded by α. In the (standard) centralized setting, Bansal and Umboh [6] gave an $O(\alpha)$-approximation LP rounding algorithm, which also translates into a near-linear time algorithm using general-purpose approximation results for explicit mixed packing and covering or pure covering LPs [39, 57, 1, 50]. Moreover, [6] showed that it is NP-hard to achieve an asymptotic improvement for the approximation factor. On the other hand, the previous two *non*-LP-based algorithms, by Lenzen and Wattenhofer [43], and Jones et al. [36], achieve an approximation factor of $O(\alpha^2)$ in *linear* time.

There is a similar situation in the distributed setting: While there is an $O(\log^2 n)$-round LP-based $O(\alpha)$-approximation algorithm implied in [40], the best non-LP-based algorithm by Lenzen and Wattenhofer [43] is an implementation of their centralized algorithm, providing an $O(\alpha^2)$-approximation within $O(\log n)$ rounds.

We address the questions of whether one can achieve an $O(\alpha)$-approximation algorithm that is *elementary*, i.e., not based on any LP-based methods, either in the centralized setting or in the distributed setting. We resolve both questions in the affirmative, and en route achieve algorithms that are faster than the state-of-the-art LP-based algorithms. Our contribution is two-fold:

1. In the centralized setting, we provide a surprisingly simple combinatorial algorithm that is asymptotically optimal in terms of both approximation factor and running time: an $O(\alpha)$-approximation in *linear* time. The previous state-of-the-art $O(\alpha)$-approximation algorithms are (1) LP-based, (2) more complicated, and (3) have super-linear running time.
2. Based on our centralized algorithm, we design a distributed combinatorial $O(\alpha)$-approximation algorithm in the CONGEST model that runs in $O(\alpha \log n)$ rounds with high probability. Not only does this result provide the first nontrivial *non*-LP-based distributed $o(\alpha^2)$-approximation algorithm for this problem, it also outperforms the best LP-based distributed algorithm for a wide range of parameters.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Graph Algorithms, Dominating Set, Bounded Arboricity, Linear time algorithms

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.33

Related Version This paper is an abbreviated version of the following full paper:
Full Version: https://arxiv.org/pdf/2102.10077.pdf

Funding *Shay Solomon*: Partially supported by the Israel Science Foundation grant No.1991/19.
Nicole Wein: Supported by NSF Grant CCF-1514339.

Acknowledgements We thank Yosi Hezi and Quanquan Liu for fruitful discussions. We would also like to thank Neal E. Young and Kent Quanrud for correspondence about their prior work.

1 Introduction

1.1 Background

The minimum dominating set (MDS) problem is a classic combinatorial optimization problem. Given a graph G we want to find a minimum cardinality set D of vertices, such that every vertex of the graph is either in D or has a neighbor in D. Besides its theoretical implications, solving this basic problem efficiently has many practical applications in domains ranging from wireless networks to text summarizing (see, e.g., [56, 46, 52]). The MDS problem was one of the first problems recognized as NP-complete [27]. It was also one of the first problems for which an approximation algorithm was analyzed: a simple greedy algorithm achieves a $\ln n$-approximation in general graphs [35]. This approximation factor is optimal up to lower order terms unless $\mathsf{P} = \mathsf{NP}$ [19].

Distributed MDS in general graphs. The first efficient distributed approximation algorithm for MDS was given by Jia, Rajaraman, and Suel [34], who gave a randomized $O(\log \Delta)$-approximation in $O(\log^2 n)$ rounds in the CONGEST model. This was improved by Kuhn, Moscibroda and Wattenhofer [40], who gave a randomized $(1+\varepsilon)(1+\ln(\Delta+1))$-approximation in $O(\log^2 \Delta/\varepsilon^4)$ rounds in the CONGEST model and in $O(\log n/\varepsilon^2)$ rounds in the LOCAL model. Ghaffari, Kuhn, and Maus [30] showed that by allowing exponential-time local computation, one can get a randomized $(1 + o(1))$-approximation in a polylogarithmic number of rounds in the LOCAL model. This result was derandomized by the network decomposition result of Rozhoň and Ghaffari [51]. From the lower bounds side, Kuhn, Moscibroda, and Wattenhofer [41] showed that getting a polylogarithmic approximation ratio requires $\Omega\big(\sqrt{\frac{\log n}{\log\log n}}\big)$ and $\Omega\big(\frac{\log \Delta}{\log\log \Delta}\big)$ rounds in the LOCAL model.

For *deterministic* distributed algorithms, improving over previous work, Deurer, Kuhn, and Maus [17] recently gave two algorithms in the CONGEST model with approximation factor $(1+\varepsilon)\ln(\Delta+1)$ for $\varepsilon > 1/\mathrm{polylog}\Delta$, running in $2^{O(\sqrt{\log n \log\log n})}$ and $O((\Delta+\log^* n)\mathrm{polylog}\Delta)$ rounds, respectively; the running time of the former CONGEST algorithm [29], achieving approximation factor $O(\log^2 n)$, is dominated by the time needed for deterministically computing a network decomposition in the CONGEST model, which, due to [28], is thus reduced to $O(\mathrm{poly}\log n)$.

Graphs of bounded arboricity. The MDS problem has been studied on a variety of restricted classes of graphs, such as graphs with bounded degree (e.g., [14]), planar and bounded genus graphs (e.g., [5, 16, 3]), and graphs of bounded arboricity – which is the focus of this paper. The class of bounded *arboricity* graphs is a wide family of *uniformly sparse* graphs, defined as follows:

▶ **Definition 1.** *Graph G has* arboricity *bounded by α if $\frac{m_s}{n_s-1} \leq \alpha$, for every $S \subseteq V$, where m_s and n_s are the number of edges and vertices in the subgraph induced by S, respectively.*

The class of bounded arboricity graphs contains the other graph classes mentioned above as well as bounded treewidth graphs, and in general all graphs excluding a fixed minor. Moreover, many natural and real world graphs, such as the world wide web graph, social networks and transaction networks, are believed to have bounded arboricity. Consequently, this class of graphs has been subject to extensive research, which led to many algorithms for bounded arboricity graphs in both the (classic) centralized setting (e.g. [23, 32, 13]) and in the distributed setting (e.g. [15, 7, 31, 54]); there are also many algorithms in other settings, such as dynamic graph algorithms, sublinear algorithms and streaming algorithms (see [11, 33, 49, 48, 47, 53, 37, 20, 21, 22, 9, 44, 10, 8], and the references therein).

In distributed settings, one cannot always assume that all processors know the arboricity of the graph, so it is important to devise robust algorithms, which can perform correctly also when the arboricity is unknown to the processors (see e.g. [7, 43]).

1.2 Approximating MDS on graphs of arboricity α

Centralized setting. In the centralized setting, there are two non-LP-based algorithms for MDS for graphs of arboricity (at most) α (for brevity, in what follows we may write graphs of "arboricity α" instead of arboricity *at most* α). One is by Lenzen and Wattenhofer [43], the other is by Jones, Lokshtanov, Ramanujan, Saurabh, and Suchý [36], and both achieve an $O(\alpha^2)$-approximation in deterministic linear time[1]. There is also a very simple LP rounding algorithm by Bansal and Umboh that gives a 3α-approximation [6]. This algorithm is very simple, after the LP has been solved. To solve the LP, there are near-linear time general-purpose approximation algorithms for explicit mixed packing and covering or pure covering LPs [39, 57, 1, 50]. Combining such an algorithm with [6] yields an $O(\alpha)$-approximation for MDS, either deterministically within $O(m \log n)$ time [57] or randomly (with high probability) within $O(n \log n + m)$ time [39]. The latter bound is super-linear in the entire (non-degenerate) regime of arboricity $\alpha = o(\log n)$; the regime $\alpha = \Omega(\log n)$ is considered degenerate, since in that case one can use the greedy linear-time $\ln n$-approximation algorithm. Bansal and Umboh [6] also proved that achieving asymptotically better approximation is NP-hard.[2]

Distributed setting. In the distributed setting, there are two non-LP-based algorithms for MDS for graphs of arboricity α, both by Lenzen and Wattenhofer [43]. The first is a randomized $O(\alpha^2)$-approximation algorithm in the CONGEST model that runs in $O(\log n)$ rounds with high probability. This algorithm was made deterministic by Amiri [2], and uses an LP-based subroutine of Even, Ghaffari, and Medina [24]. The second algorithm of Lenzen and Wattenhofer is a deterministic $O(\alpha \log \Delta)$-approximation algorithm in the CONGEST model that runs in $O(\log \Delta)$ rounds, where Δ is the maximum degree.

Regarding LP-based algorithms, Kuhn, Moscibroda, and Wattenhofer [40] developed a general-purpose method for solving LPs of a particular structure in the distributed setting. It seems that by applying their method (specifically, Corollary 4.1 of [40]) to the LP approximation result of Bansal and Umboh in bounded arboricity graphs [6], one can get a deterministic $O(\alpha)$-approximation algorithm for MDS in the CONGEST model that runs in $O(\log^2 \Delta)$ rounds, but such a result has not been explicitly claimed in the literature.

A natural question. The aforementioned results demonstrate a significant gap for MDS algorithms in bounded arboricity graphs when comparing LP-based methods to elementary *combinatorial* approaches. It is natural to ask whether this gap can be bridged.

- In the centralized setting, is there any efficient non-LP-based $O(\alpha)$-approximation algorithm for MDS (even one that is slower than the aforementioned $O(m \log n)$ time deterministic and $O(n \log n + m)$ time randomized LP-based algorithms)? Further, can one achieve an $O(\alpha)$-approximation in *linear time* using *any* (even LP-based) algorithm?

[1] Note that the theorem statement of [43] has a typo suggesting that the approximation factor is $O(\alpha)$.
[2] Achieving $(\alpha - 1 - \varepsilon)$-approximation is NP-hard for any $\varepsilon > 0$ and any fixed α; achieving $(\lfloor \alpha/2 \rfloor - \varepsilon)$-approximation is NP-hard for any $\varepsilon > 0$ and any $\alpha = 1, \ldots, \log^{\delta} n$, for some constant δ [6, 18].

- In the distributed setting, is there any efficient non-LP-based distributed $O(\alpha)$-approximation algorithm for MDS? Further, can one achieve an $O(\alpha)$-approximation in the CONGEST model within $o(\log^2 \Delta)$ rounds using *any* (even LP-based) algorithm?

We note the caveat that there is no clear-cut distinction between combinatorial and non-combinatorial algorithms, but we operate under the premise that an algorithm is combinatorial if all its intermediate computations have a natural combinatorial interpretation in terms of the original problem. While all algorithms presented in this paper are certainly combinatorial under this premise, it is far less clear whether prior work is. In particular, the previous state-of-the-art LP-based approaches are based on general-purpose primal/dual methods; when restricted to the MDS problem, it is possible that these methods could reduce, after proper adaptations, into simpler combinatorial algorithms. Nonetheless, even if possible, it is unlikely that the resulting algorithm would be as simple and elementary as ours. In the distributed setting, [42] gives an LP-based algorithm specifically for MDS that is simpler than the subsequent general-purpose LP-based algorithm of Moscibroda, and Wattenhofer [40]; however, [42] is inferior to [40] in both approximation ratio and running time.

1.3 Our Contributions

We answer all parts of the above question in the affirmative. In particular, we give algorithms that achieve the asymptotically optimal approximation factor of $O(\alpha)$, and are not only simple and elementary, but also run faster than all known algorithms, including LP-based algorithms. We note that $O(\alpha)$ is the asymptotically optimal approximation factor for polynomial time algorithms in the centralized setting and also in distributed settings where processors are assumed to have polynomially-bounded processing power.

Centralized Setting. Our core contribution is an asymptotically optimal algorithm in the centralized setting.

▶ **Theorem 2.** *For graphs of arboricity α, there is an $O(m)$ time $O(\alpha)$-approximation algorithm for MDS.*

We note that our algorithm works even when α is not known a priori, since there is a linear time 2-approximation algorithm for computing the arboricity of a graph [4].

Our algorithm is asymptotically optimal in both running time and approximation factor: it runs in linear time, and asymptotically improving the approximation factor it gets is proved to be NP-hard [6]. (The constant in the approximation ratio is not tight; our algorithm gives an 8α-approxmation.) While the quantitative improvement in running time over prior work is admittedly minor (a logarithmic factor over the deterministic algorithm, and $\log n/\alpha$ over the randomized algorithm), still getting a truly linear time algorithm is qualitatively very different than an almost-linear time. Indeed, the study of linear time algorithms has received much attention over the years, even when it comes to shaving factors that grow as slowly as inverse-Ackermann type functions. This line of work includes celebrated breakthroughs in computer science: For example, for the Union-Find data structure, efforts to achieve a linear time algorithm led to a lower bound showing that inverse-Ackermann function dependence is necessary [25], matching the upper bound [55], which is a cornerstone result in the field. Another example is MST, where the inverse-Ackermann function was shaved from the upper bound of [12] to achieve a linear time algorithm either using randomization [38] or when the edge weights are integers represented in binary [26], but it remains a major open problem whether or not there exists a linear time deterministic comparison-based MST algorithm.

Distributed Setting. We demonstrate the applicability of our centralized algorithm, by using its core ideas to develop a distributed algorithm.

▶ **Theorem 3.** *For graphs of arboricity α, there is a randomized distributed algorithm in the* CONGEST *model that gives an $O(\alpha)$-approximation for MDS and runs in $O(\alpha \log n)$ rounds. The bound on the number of rounds holds with high probability (and in expectation). The algorithm works even when either α or n is unknown to each processor.*

For the "interesting" parameter regime where Δ is polynomial in n, and $\alpha = o(\log n)$, the number of rounds in our algorithm beats the prior work obtained by combining [40] and [6] which appears to run in $O(\log^2 \Delta)$ rounds; as noted already, such an algorithm has not been claimed explicitly before. We note the caveat that our algorithm is randomized while their algorithm appears to be deterministic.

In the process of obtaining our distributed algorithm, we also obtain a *deterministic* algorithm in the LOCAL model (with polynomial message sizes) in a polylogarithmic number of rounds, via reduction to the maximal independent set (MIS) problem:

▶ **Theorem 4.** *Suppose there is a deterministic (resp., randomized) distributed algorithm in the* LOCAL *model for computing an MIS on a general graph in $R(n)$ rounds. Then, for graphs of arboricity α, there is a deterministic (resp., randomized) distributed algorithm in the* LOCAL *model that gives an $O(\alpha)$-approximation for MDS in $O(R(n) \cdot \alpha^2 \log n)$ rounds. The algorithm works even when either α or n is unknown to each processor.*

While Theorem 4 is the first deterministic non-LP-based algorithm to achieve an $O(\alpha)$-approximation, we note that the LP-based approach obtained by combining [40] and [6] appears to achieve fewer rounds and work in the CONGEST model. Theorem 4 is not our main result and is used as a stepping stone towards our $O(\alpha \log n)$ round algorithm in the CONGEST model, which is deferred to the full version [45] due to space constraints.

We finally note that unlike in the centralized setting, handling unknown α in the distributed setting it is not trivial and requires special treatment; in the full version [45] we demonstrate that all of our distributed algorithms can cope with unknown α without increasing the approximation factor and running time beyond constant factors.

Wider applicability. We have demonstrated the applicability of our centralized algorithm to the distributed setting. We anticipate that the core idea behind our centralized algorithm could be applied more broadly, to other settings that involve locality. Perhaps the prime example in this context is the standard (centralized) setting of dynamic graph algorithms, where the graph undergoes a sequence of edge updates (a single edge update per step), and the algorithm should maintain the graph structure of interest ($O(\alpha)$-approximate MDS in our case) with a small *update time* – preferably $\operatorname{poly}\log(n)$ and ideally $O(1)$.

1.4 Technical overview

Centralized algorithm. As a starting point, we consider the algorithm of Jones, Lokshtanov, Ramanujan, Saurabh, and Suchý [36], which achieves an $O(\alpha^2)$-approximation in linear time. Their algorithm is as follows. They iteratively build a dominating set D and maintain a partition of the remaining vertices into the dominated vertices B (the vertices that have a neighbor in D), and the undominated vertices W. This partition of the vertices, as well as further partitioning described later, is shown in Figure 1. The basic property of arboricity α graphs used by their algorithm is that every subgraph contains a vertex of degree $O(\alpha)$. They begin by choosing a vertex v with degree $O(\alpha)$ and adding v along with

v's entire neighborhood $N(v)$ to D. The intuition behind this is that at least one vertex in $\{v\} \cup N(v)$ must be in OPT (an optimal dominating set), since OPT must dominate v. Hence, they add at least one vertex in OPT and use that to pay for adding $O(\alpha)$ vertices not in OPT. We say that a vertex w *witnesses* v and the vertices in $N(v)$ that are added to D, if $w \in OPT \cap (\{v\} \cup N(v))$. Now, the goal of the algorithm is to iteratively choose vertices v to add to D along with $O(\alpha)$ many of v's neighbors so that each vertex in OPT witnesses $O(\alpha)$ vertices v along with $O(\alpha)$ neighbors for each such vertex v. That is, each vertex in OPT witnesses $O(\alpha^2)$ vertices in D, which yields an $O(\alpha^2)$-approximation.

To choose which vertices v and which $O(\alpha)$ of v's neighbors to add to D, they partition the set B into two subsets B_{low} and B_{high}, which are the sets of vertices in B with low and high degree to W, respectively, where the degree threshold is $\delta\alpha$ for some constant δ. We also define $W_{low} \subseteq W$ (differently from the notation of [36]) as the subset of vertices with degree at most $\delta\alpha$ in the subgraph induced by $W \cup B_{high}$. They add a vertex $w \in W_{low}$ to D along with w's $O(\alpha)$ neighbors that are in $W \cup B_{high}$. In the interest of brevity, we will not motivate why this scheme achieves the desired outcome that each vertex in OPT witnesses $O(\alpha^2)$ vertices in D.

The key innovation in our algorithm that allows us to reduce the approximation factor from $O(\alpha^2)$ to $O(\alpha)$ is a simple but powerful idea. After choosing a vertex w to add to D, we do not *immediately* add $O(\alpha)$ of w's neighbors to D. Instead w casts a "vote" for these $O(\alpha)$ neighbors, and only once a vertex gets $\delta\alpha$ many votes is it added to D. With this modification, we can argue that each vertex in OPT still witnesses $O(\alpha)$ such vertices w as in the previous approach, but the catch here is that each such vertex w contributes only $O(1)$ neighbors to D on average, so each vertex in OPT only witnesses a *total* of $O(\alpha)$ vertices in D, rather than $O(\alpha^2)$. Moreover, it is trivial to implement this algorithm in linear time.

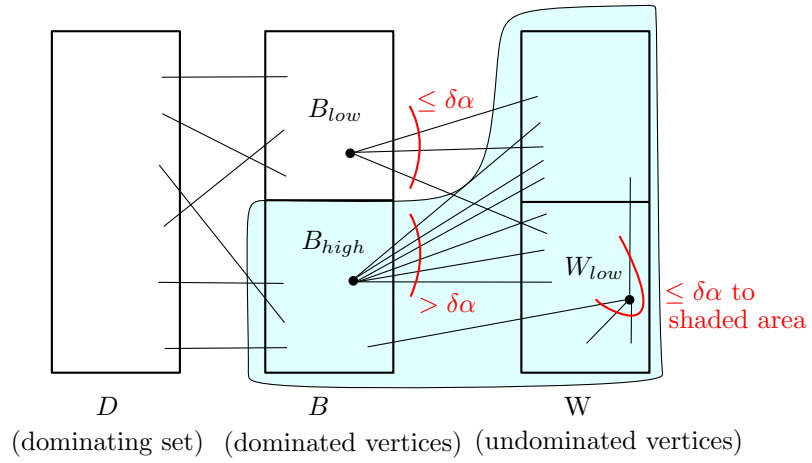

Figure 1 The partition of V into D, B and W, and further partitions of B and W.

Distributed algorithms using MIS. This section concerns the proof of Theorem 4: our reduction from MDS to MIS in the LOCAL model. This section also concerns a modification of this reduction that gives an $O(\alpha^2 \log^2 n)$ round algorithm in the CONGEST model. We use this algorithm as a stepping stone towards obtaining our main distributed algorithm (Theorem 3) which runs in $O(\alpha \log n)$ rounds in the CONGEST model.

We adapt our centralized algorithm to the distributed setting as follows. Recall that in our centralized algorithm, we repeatedly choose a vertex $w \in W_{low}$, add w to D, and cast a vote for each vertex in $N(w) \cap (W \cup B_{high})$. For our distributed algorithms, we would like

to choose *many* such vertices w and process them in *parallel*. In fact, a constant fraction of the vertices in $W \cup B_{high}$ could be chosen as our vertex w since a constant fraction of vertices in a graph of arboricity α have degree $O(\alpha)$. However, we cannot simply process all of these vertices in parallel. In particular, if a vertex v has many neighbors being processed in parallel, v might accumulate many votes during a single round. This would invalidate the analysis of the algorithm, which relies on the fact that once a vertex v receives $\delta\alpha$ votes, v enters D.

To overcome this issue, we compute an MIS with respect to a 2-hop graph built from a subgraph of "candidate" vertices, and only process the vertices in this MIS in parallel. This MIS has two useful properties: 1. Its maximality implies that in any 2-hop neighborhood of a candidate vertex there is a vertex in the MIS; this helps to bound the number of rounds, and 2. Its independence implies that every vertex has at most one neighbor in the MIS, which ensures that any vertex can only receive one vote per round. To conclude, this approach gives a reduction from distributed MDS to distributed MIS in the LOCAL model. This approach can be made to work in the CONGEST model by replacing the black-box MIS algorithm with a 2-hop version of Luby's algorithm. This approach of running the 2-hop version of Luby's algorithm was also used in [43] for their distributed (α^2)-approximation for MDS.

Faster randomized distributed algorithm. In the CONGEST model, our distributed algorithm using MIS runs in $O(\alpha^2 \log^2 n)$ rounds with high probability. We devise a new, more nuanced algorithm that decreases the number of rounds to $O(\alpha \log n)$ with high probability. Our new algorithm is based on our previous algorithm, but with two key modifications, which save factors of $\log n$ and α, respectively.

Our first key modification, which shaves a $\log n$ factor from the number of rounds, is that we do not run an MIS algorithm as a black box. Instead, we run only a single phase of a Luby-like MIS algorithm before updating the data structures. Intuitively, this saves a $\log n$ factor because we are running just one phase of a $O(\log n)$-phase algorithm, but it is not clear a priori if we achieve the same progress as Luby's algorithm in a single phase. We show that this is indeed the case via more refined treatment of the behavior of each edge.

Our second key modification, which shaves an α factor from the number of rounds, concerns the Luby-like algorithm. Recall that in Luby's algorithm, each vertex v picks a random value $p(v)$ and then joins the MIS if $p(v)$ is the local minimum. In our algorithm, a vertex v instead joins the dominating set if $p(v)$ is an α-minimum, which roughly means that $p(v)$ is among the α smallest values that it is compared to. We show that with this relaxed definition, we still have the desired property that no vertex receives more than $\delta\alpha$ votes in a single round.

The main technical challenge is the analysis of the number of rounds. It is tempting to use an analysis similar to that of Luby's algorithm, where we count the expected number of "removed edges" over time. However, our above modifications introduce several complications that preclude such an analysis. Instead, we use a carefully chosen function to measure our progress. Throughout the algorithm, we add "weight" to particular edges, and our function measures the "total available weight". Specifically, whenever a vertex v is added to the dominating set, v adds *weight* to a particular set of edges in its 2-hop neighborhood. We show that the total amount of weight added in a single iteration of the algorithm decreases the total available weight substantially, which allows us to bound the total number of iterations.

All of our distributed algorithms so far have assumed that α is known to each processor but that n is unknown. We additionally show that all of them can be made to work in the setting where α is unknown but n is known. The idea of this modification is to guess

$\log n$ values of α and run a truncated version of the algorithm for each guess. However, it is impossible for an individual processor to know which guess of α is the most accurate without knowing the whole graph, so the processors cannot coordinate their guesses globally. We end up with different processors using different guesses of α, but we show that we can nonetheless obtain an algorithm whose approximation factor and running time are in accordance with the correct α.

1.5 Organization

Section 2 is for preliminaries. In Section 3, we present our centralized algorithm (Theorem 2). In Section 4, we present our distributed algorithms using MIS: in the LOCAL model we prove Theorem 4, and in the CONGEST model we give a randomized algorithm with $O(\alpha^2 \log^2 n)$ rounds, as a warm-up for the faster algorithm of Theorem 3. In section 5 of the full version [45] we prove Theorem 3 and that all our distributed algorithms can cope with unknown α.

2 Preliminaries

Let $G = (V, E)$ be an unweighted undirected graph. For any $S \subseteq V$, let $G[S]$ be denote the subgraph induced by S. For any $v \in V$, $N_G(v)$ denotes the neighborhood of v, and $\deg_G(v) = |N_G(v)|$ denotes the degree of v. When the graph G is clear from context, we omit the subscript.

Our distributed algorithm apply to the LOCAL and CONGEST models of distributed computing; definitions can be found in section 2 of the full version [45]. For the problem of MDS in both models, the requirement is that at the end of the computation, every vertex knows whether or not it belongs to the dominating set.

The following two simple claims about graphs of bounded arboricity will be useful.

▷ Claim 5. In a graph of arboricity α, every subgraph contains a vertex of degree $\leq 2\alpha$.

▷ Claim 6. In a graph G with arboricity α, at least half of the vertices in any subgraph have degree at most 4α.

3 Linear time $O(\alpha)$-approximation for MDS

In this section we will prove Theorem 2.

3.1 Algorithm

A description of our algorithm is as follows. See Algorithm 1 for the pseudocode.

We first introduce some notation. Since our algorithm builds off of [36], we stick to their notation for the most part. See Figure 1. We define a constant δ and let $\delta\alpha$ be our *degree threshold*. We will set $\delta = 2$, but we use the variable δ so that our analysis also applies to our distributed algorithms, where δ is a different constant. We maintain a partition of the vertices into three sets: D, B, and W, where initially $D = \emptyset$, $B = \emptyset$, and $W = V$. The set D is our current dominating set, the set B is the vertices not in D with at least one neighbor in D, and the set W is the remaining vertices, i.e. the undominated vertices. The set B is further partitioned into two sets based on the degree of each vertex to W. Let $B_{low} = \{v \in B : |N(v) \cap W| \leq \delta\alpha\}$ and let $B_{high} = B \setminus B_{low}$. Let $W_{low} = \{v \in W : |N(v) \cap (W \cup B_{high})| \leq \delta\alpha\}$ Also, each vertex v has a *counter* c_v initialized to 0. (The counter c_v counts the number of "votes" that v receives, for the notion of "votes" introduced in the technical overview.)

First we claim that while W is nonempty, W_{low} is also nonempty. By Claim 5, $G[W \cup B_{high}]$ contains a vertex v of degree at most 2α. Since $\delta = 2$, v cannot be in B_{high} by the definition of B_{high}, so v must be in W, and hence in W_{low}.

The algorithm proceeds as follows. While there still exists an undominated vertex (i.e. while $W \neq \emptyset$), we do the following. First, we pick an arbitrary vertex $w \in W_{low}$ (we showed that W_{low} is nonempty). Then, for all $v \in N(w) \cap (W \cup B_{high})$, we increment c_v, and if $c_v = \delta\alpha$, we add v to D. Then, we add w to D. Lastly, we update the sets B, $_{low}$, B_{high}, W, and W_{low} according to their definitions. This concludes the description of the algorithm.

Algorithm 1 Linear time $O(\alpha)$-approximation for MDS.

Initialize partition: $D \leftarrow \emptyset$, $B = \emptyset$, $B_{high} \leftarrow \emptyset$, $B_{low} \leftarrow \emptyset$, $W \leftarrow V$, $W_{low} = \{v \in V : \deg(v) \leq \delta\alpha\}$
Initialize counters: $\forall v \in V : c_v \leftarrow 0$
while $W \neq \phi$ **do**
    $w \leftarrow$ a vertex in W_{low}
    for all $v \in N(w) \cap (W \cup B_{high})$ **do**
        $c_v \leftarrow c_v + 1$
        if $c_v = \delta\alpha$ **then**
            $D \leftarrow D \cup v$
    $D \leftarrow D \cup w$
    // **Bookkeeping to update partition:**
    $B = \{v : N(v) \cap D \neq \emptyset\}$
    $B_{low} = \{v \in B : |N(v) \cap W| \leq \delta\alpha\}$
    $B_{high} = B \setminus B_{low}$
    $W = V \setminus (D \cup B)$
    $W_{low} = \{v \in W : |N(v) \cap (W \cup B_{high})| \leq \delta\alpha\}$
Return D

3.2 Analysis

First, we note that D is indeed a dominating set because the algorithm only terminates once the set W of vertices that are not dominated, is empty.

3.2.1 Approximation ratio analysis

Let OPT be an optimal MDS. We will prove that the set D returned by Algorithm 1 is of size at most $4\delta\alpha \cdot |OPT|$.

We first make the next claim about the behavior of the partition of vertices over time.

▷ Claim 7.
(1) No vertex can ever leave D.
(2) No vertex can ever enter W from another set.
(3) No vertex can ever leave B_{low}.

Proof. Item 1 is by definition. Item 2 follows from item 1 combined with the fact that W is defined as the set of vertices with no neighbors in D. Now we prove item 3. A vertex from B_{low} cannot enter W by item 2. A vertex from B_{low} cannot enter B_{high} since the degree partition of B is based on degree to W, and by item 2 the degree of any vertex to W can only decrease over time. A vertex from B_{low} cannot enter D because there are two ways a vertex can enter D: on Algorithm 1 a vertex can only enter D from $W \cup B_{high}$, and on Algorithm 1 a vertex can only enter D from W. ◁

To show that $|D| \leq 4\delta\alpha \cdot |OPT|$, we partition D into two sets, D_{active} and $D_{passive}$, and bound each of these sets separately. The set D_{active} consists of the vertices added to D due to being chosen as the vertex w; that is, the vertices added to D in Algorithm 1 of Algorithm 1. The set $D_{passive}$ consists of the vertices added to D as a result of their counters reaching $\delta\alpha$; that is, the vertices added to D in Algorithm 1 of Algorithm 1. We first bound $|D_{active}|$.

▷ Claim 8. $|D_{active}| \leq 2\delta\alpha \cdot |OPT|$.

Proof. For each vertex $v \in D_{active}$, we assign v to an arbitrary vertex $u \in N(v) \cap OPT$, and we say that u *witnesses* v. Such a vertex u exists since OPT is a dominating set. For each vertex $u \in OPT$, let $D_u \subseteq D_{active}$ be the set of vertices that u witnesses. Our goal is to show that for each $u \in OPT$, $|D_u| \leq 2\delta\alpha$.

Fix a vertex $u \in OPT$. We partition the vertices $v \in D_u$ into two sets $D_u[B_{low}]$ and $D_u[B_{high} \cup W]$. Let $D_u[B_{low}] \subseteq D_u$ be the vertices that enter D while u is in B_{low}. Let $D_u[B_{high} \cup W] \subseteq D_u$ be vertices that enter D while u is in $B_{high} \cup W$. We note that no vertex in D_u can enter D while u is in D, because by definition, every vertex in $D_{active} \supseteq D_u$ moves directly from W to D. Therefore, $D_u = D_u[B_{low}] \cup D_u[B_{high} \cup W]$.

We first bound $\left|D_u[B_{low}]\right|$. By definition, while u is in B_{low}, u has at most $\delta\alpha$ neighbors in W. Since no vertex can ever enter W by Claim 7, no vertex can ever enter $N(u) \cap W$. Therefore, starting from the time that u first enters B_{low}, the total number of vertices ever in $N(u) \cap W$ is at most $\delta\alpha$. Every vertex $v \in D_u[B_{low}]$ is in $N(u) \cap W$ right before moving to D, so $\left|D_u[B_{low}]\right| \leq \delta\alpha$. Next, we bound $D_u[B_{high} \cup W]$. By the specification of the algorithm, whenever a vertex $v \in D_u[B_{high} \cup W]$ enters D, the counter c_u is incremented. Once c_u reaches $\delta\alpha$, u is added to D. Therefore, $\left|D_u[B_{high} \cup W]\right| \leq \delta\alpha$.

Putting everything together, we have $|D_u| = \left|D_u[B_{low}]\right| + \left|D_u[B_{high} \cup W]\right| \leq 2\delta\alpha$. ◁

Now we bound $D_{passive}$.

▷ Claim 9. $|D_{passive}| \leq |D_{active}|$.

Proof. We will show that every vertex in $D_{passive}$ has at least $\delta\alpha$ neighbors in D_{active}, while every vertex in D_{active} has at most $\delta\alpha$ neighbors in $D_{passive}$. Then, by the pigeonhole principle, it follows that $|D_{passive}| \leq |D_{active}|$.

First, we will show that every vertex in $D_{passive}$ has at least $\delta\alpha$ neighbors in D_{active}. By definition, every vertex $v \in D_{passive}$ has had its counter c_v incremented $\delta\alpha$ times. Every time c_v is incremented, one of v's neighbors (the vertex w from Algorithm 1) is added to D, joining D_{active}. Each such neighbor of v that joins D_{active} is distinct since every vertex can be added to D at most once by Claim 7. Therefore, every vertex in $D_{passive}$ has at least $\delta\alpha$ neighbors in D_{active}.

Now we will show that every vertex in D_{active} has at most $\delta\alpha$ neighbors in $D_{passive}$. Fix a vertex $w \in D_{active}$. By definition, when w enters D, w is moved straight from W to D. Thus, by Claim 7, w is never in B. Therefore, w is added to D before any of its neighbors are added to D, as otherwise w would enter B. Therefore, when w enters D, all of w's neighbors that will enter $D_{passive}$ are in $B \cup W$. By Claim 7, no vertex in B_{low} can ever enter D, so actually, when w enters D all of w's neighbors that will enter $D_{passive}$ are in $B_{high} \cup W$. By definition, when w enters D, w has at most $\delta\alpha$ neighbors in $B_{high} \cup W$. Therefore, w has at most $\delta\alpha$ neighbors in $D_{passive}$. ◁

Combining Claim 8 and Claim 9, we have that $|D| = |D_{active}| + |D_{passive}| \leq 4\delta\alpha \cdot |OPT|$.

3.2.2 Running time analysis

Our goal is to prove that Algorithm 1 runs in $O(m)$ time.

Throughout the execution of the algorithm, we maintain a data structure that consists of the following:

- The partition of V into D, B, W; with subsets B_{low}, B_{high}, W_{low}
- The induced graph $G[W \cup B_{high}]$ represented as an adjacency list
- For each vertex $v \in W \cup B_{high}$, the quantities $|N(v) \cap W|$ and $|N(v) \cap (W \cup B_{high})|$

We can bound the time needed for maintaining the data structure using the following observations:

- The data structure can be initialized in $O(m)$ time
- To maintain this data structure, it suffices to scan the neighborhood of a vertex every time it move between subsets
- Every vertex moves between subsets a constant number times during the run of the algorithm
- Maintaining the data structure allows the algorithm to run in time $O(m)$

The first three observations implies that maintaining the data structure takes time $O(m)$. Together with the last observation, we have that the entire algorithm takes time $O(m)$. Full analysis and proof can be found in subsection 3.2.2 of the full version [45].

4 Distributed $O(\alpha)$-approximation for MDS using MIS

In this section we will prove Theorem 4. We also show how to modify of the proof of Theorem 4 to get a bound in the CONGEST model:

▶ **Theorem 10.** *For graphs of arboricity α, there is a randomized distributed algorithm in the* CONGEST *model that gives an $O(\alpha)$-approximation for MDS that runs in $O(\alpha^2 \log^2 n)$ rounds with high probability. The algorithm works even when either α or n is unknown to each processor.*

In the full version [45], we use the algorithm of Theorem 10 as a starting point to get an improved algorithm with $O(\alpha \log n)$ rounds.

The algorithms presented in this section assume that α is known to each processor but n is unknown. We defer discussion of handling unknown α to the full version [45].

4.1 Algorithm

4.1.1 Overview

Our algorithm is an adaptation of our centralized algorithm from Theorem 2 to the distributed setting. Recall that in our centralized algorithm, we repeatedly choose a vertex $w \in W_{low}$, add w to the dominating set, and increment the *counter* of w's neighbors that are in $W \cup B_{high}$. For our distributed algorithms, we would like process *many* vertices in W_{low} in parallel. There are in fact many vertices in W_{low} (if $\delta \geq 4$) since Claim 6 implies that at least half of the vertices in any subgraph has degree at most 4α. However, we cannot simply process all of W_{low} at once. In particular, if a vertex v has many neighbors being processed in parallel, v might have its counter incremented once for each of these neighbors. This is undesirable because the analysis of our centralized algorithm relies on the fact that once a vertex has its counter incremented to $\delta\alpha$, it is added to the dominating set. Therefore, we would like to guarantee that only a limited number of v's neighbors are processed in parallel.

This is where the MIS problem becomes relevant: we ensure that no vertex has more than one neighbor being processed in parallel by taking an MIS I with respect to the graph G_{low} defined as follows: the vertex set of G_{low} is W_{low}. There is an edge (u, v) in G_{low} if there is a path of length 2 between u and v in $G[W \cup B_{high}]$. Note that because no vertex has more than one neighbor in I, we can process all vertices in I in parallel and only increase the counter of each vertex by at most one.

The algorithms for Theorem 4 and Theorem 10 are identical except for the MIS subroutine. Theorem 4 is for the LOCAL model so we can simply run any distributed MIS algorithm that works in the LOCAL model on G_{low} as a black box. On the other hand, Theorem 10 is for the CONGEST model and because G_{low} can have higher degree than G, running an MIS algorithm directly on G_{low} could result in messages that become too large after translating the algorithm to run on G. To bypass this issue, we use a simple modification of Luby's algorithm that computes I using only small messages, without increasing the number of rounds.

4.1.2 Algorithm description

We provide a description of the algorithms here, and include the pseudocode in Algorithm 2. The only difference between the algorithms for Theorem 4 and Theorem 10 is the MIS subroutine, which we will handle separately later.

The sets D, B, W, B_{high}, B_{low}, and W_{low} are defined exactly the same as in our centralized algorithm, except we set $\delta = 4$ instead of $\delta = 2$ so that we can apply Claim 6 instead of Claim 5. We repeat the definitions here for completeness. The set D is our current dominating set, the set B is the vertices not in D with at least one neighbor in D, and the set W is the remaining vertices, i.e. the undominated vertices. The set B is further partitioned into two sets based on the degree of each vertex to W. Let $B_{low} = \{v \in B : |N(v) \cap W| \leq \delta\alpha\}$ and let $B_{high} = B \setminus B_{low}$. Also, let $W_{low} = \{v \in W : |N(v) \cap (W \cup B_{high})| \leq \delta\alpha\}$. Lastly, each vertex v has a *counter* c_v.

Each vertex v maintains the following information:

- The set(s) among D, B, W, B_{high}, B_{low}, and W_{low} that v is a member of.
- The quantity $|N(v) \cap W|$.
- The quantity $|N(v) \cap (W \cup B_{high})|$.
- The counter c_v.

At initialization, every vertex v is in W (so D and B are empty). Consequently, the quantities $|N(v) \cap W|$ and $|N(v) \cap (W \cup B_{high})|$ are both equal to $\deg(v)$. For each vertex v, if $\deg(v) \leq \delta\alpha$, then $v \in W_{low}$. Each counter c_v is initialized to 0.

It will be useful to define the graph G_{low}, which changes over the execution of the algorithm:

▶ **Definition 11.** *Let G_{low} be the graph with vertex set W_{low} such that there is an edge (u, v) in G_{low} if there is a path of length 2 between u and v in $G[W \cup B_{high}]$.*

The algorithm proceeds as follows. Repeat the following until W is empty. Compute an MIS I with respect to G_{low}. This step is implemented differently for Theorem 4 and Theorem 10, and we describe the details of this step later.

Then, each vertex in I adds itself to D and tells its neighbors to increment their counters. Whenever the counter of a vertex reaches $\delta\alpha$, it enters D (and does *not* tell its neighbors to increment their counters).

Whenever a vertex moves from one set of the partition to another, it notifies each of its neighbors v so that v can update the quantities $|N(v) \cap W|$ and $|N(v) \cap (W \cup B_{high})|$, and move to the appropriate set. When no more vertices are left in W, B_{high} is also empty, and all processors terminate. This concludes the description of the algorithm. See Algorithm 2 for the precise ways that vertices react to the messages that they receive.

4.1.3 MIS subroutine

Theorem 4 is a reduction from MDS to MIS, while Theorem 10 is not, so we need to describe the MIS subroutine (in the CONGEST model) only for Theorem 10. Recall that we cannot use a reduction to MIS in the CONGEST model because running an MIS algorithm directly on G_{low} could result in messages that become too large after translating the algorithm to run on G.

Our goal is to compute an MIS with respect to G_{low}, using small messages sent over G. We use a simple adaptation of Luby's algorithm. Recall that Luby's algorithm builds an MIS I as follows. While the graph is non-empty, do the following: Add all singletons to I. Then, each vertex v picks a random value $p(v) \in [0, 1]$. Then, all vertices whose value is less than that of all of their neighbors are added to I. Then, all vertices that are in I or have a neighbor in I are removed from the graph for the next iteration of the loop.

We use the following adaptation of Luby's algorithm. See Algorithm 3 for the pseudocode. Initially, the set L of *live* vertices is the set W_{low}. While $L \neq \emptyset$, do the following: Each vertex $v \in L$ picks a random value $p(v) \in [0, 1]$. In the first round each $v \in L$ sends $p(v)$ to its neighbors. In the second round, each vertex that receives one or more values $p(v)$, forwards to its neighbors the minimum value that it received. Then, for each vertex $v \in W_{low}$, if $p(v)$ is equal to the minimum value that v receives in the second round, v is added to I. When v is added to I, v notifies its neighbors, and each neighbor of v that is in $W \cup B_{high}$ forwards this notification to their neighbors. Note that each vertex has at most one neighbor in I, so forwarding this notification only takes one round. Now, every vertex knows whether it has a neighbor with respect to G_{low} that is in I, and every vertex that does is removed from L for the next iteration of the loop.

The proof that this algorithm runs in $O(\log n)$ rounds with high probability and produces an MIS with respect to G_{low} is the same as the analysis of Luby's algorithm and we will not include it here.

4.2 Analysis

The proof that Algorithm 2 achieves an $O(\alpha)$-approximation is precisely the same as that of the centralized algorithm (see Section 3.2.1) given that no counter c_v ever exceeds $\delta\alpha$. This is true because in a single iteration of the **while** loop each vertex can only have its counter incremented once since only vertices in the MIS I send INCREMENT COUNTER messages, and each vertex in $W \cup B_{high}$ only has at most one neighbor in I. This bound on the number of neighbors in I holds, since otherwise there is a path of length 2 between two vertices in $G[W \cup B_{high}]$, making I not an independent set in G_{low}. Once c_v reaches $\delta\alpha$, the vertex v enters D, which prevents c_v from increasing in the future.

Our goal in this section is to prove that if the MIS subroutine takes $R(n)$ rounds, then Algorithm 2 takes $O(R(n) \cdot \alpha^2 \log n)$ rounds. First, we note that the body of the **while** loop besides the MIS subroutine takes a constant number of rounds. Thus, our goal is to show that the number of iterations of the **while** loop is $O(\alpha^2 \log n)$.

Algorithm 2 Distributed $O(\alpha)$-approximation for MDS using MIS.

Initialize partition: $D \leftarrow \emptyset$, $B_{high} \leftarrow \emptyset$, $B_{low} \leftarrow \emptyset$, $W \leftarrow V$, $W_{low} \leftarrow \{v \in V : \deg(v) \leq \delta\alpha\}$
Initialize counters: $\forall v \in V : c_v \leftarrow 0$
Initialize degrees: $\forall v \in V : |N(v) \cap W| = \deg(v)$, $|N(v) \cap (W \cup B_{high})| = \deg(v)$
while $W \neq \emptyset$ **do**
    Find an MIS I with respect to the graph G_{low}
    Each vertex v runs the following procedure:
    if $v \in I$ **then**
        Move v to D
        Send INCREMENT COUNTER message to neighbors
        Send MOVED FROM W TO D message to neighbors
    if $v \in W \cup B_{high}$ and v receives INCREMENT COUNTER **then**
        Increment c_v
        if $c_v = \delta\alpha$ **then**
            if $v \in W$ **then**
                Send MOVED FROM W TO D message to neighbors
            if $v \in B_{high}$ **then**
                Send MOVED FROM B_{high} TO D message to neighbors
            Move v to D
    // **The rest of the algorithm is bookkeeping**
    if v receives MOVED FROM W TO D **then**
        Decrement $|N(v) \cap W|$
        if $v \in B_{high}$ and $|N(v) \cap W| = \delta\alpha$ **then**
            Move v to B_{low}
    if v receives MOVED FROM W TO D or MOVED FROM B_{high} TO D **then**
        Decrement $|N(v) \cap (W \cup B_{high})|$
        if $v \in W$ and $|N(v) \cap W| \leq \delta\alpha$ **then**
            Move v to B_{low}
            Send MOVED FROM W TO B_{low} message to neighbors
        else if $v \in W$ and $|N(v) \cap W| > \delta\alpha$ **then**
            Move v to B_{high}
            Send MOVED FROM W TO B_{high} message to neighbors
    if v receives MOVED FROM W TO B_{low} or MOVED FROM W TO B_{high} **then**
        Decrement $|N(v) \cap W|$
        if $v \in B_{high}$ and $|N(v) \cap W| = \delta\alpha$ **then**
            Move v to B_{low}
    if v receives MOVED FROM W TO B_{low} **then**
        Decrement $|N(v) \cap (W \cup B_{high})|$
        if $v \in W$ and $|N(v) \cap (W \cup B_{high})| = \delta\alpha$ **then**
            Add v to W_{low}

Algorithm 3 Distributed MIS with respect to G_{low} in the CONGEST model.

```
1: L = W_low
2: while L ≠ ∅ do
3:     Each vertex v runs the following procedure:
4:     if v ∈ L then
5:         p(v) ← a value in [0, 1] chosen uniformly at random
6:         Send p(v) message to neighbors
7:     Send m_v = min_{y∈N(v)∩L} p(y) message to neighbors
8:     if p(v) = min_{y∈N(v)} m_y then
9:         Add v to I
10:        Send ADDED message to neighbors
11:    if v ∈ W ∪ B_high and v receives ADDED then
12:        Send NEIGHBOR ADDED message to neighbors
13:    if v receives NEIGHBOR ADDED and v ∈ L then
14:        Remove v from L
```

We begin with a simple claim about the behavior of the partition of vertices over time:

▷ Claim 12.
(1) No vertex can ever enter W from another set.
(2) No vertex can ever move from W_{low} to W_{high}.

Proof. The proof of item 1 is the same as in the proof of Claim 7. For item 2, it is impossible for a vertex to move from W_{low} to W_{high} since for all v the quantity $N(v) \cap (W \cup B_{high})$ that determines membership in W_{low} versus W_{high}, can only decrease over time (in Algorithm 2, this quantity is only decremented). ◁

We begin with the following claim, which when combined with Claim 12, implies that each vertex only spends a limited number of rounds in W_{low}.

▷ Claim 13. For every vertex v that is ever in W_{low}, within $(\delta\alpha)^2$ iterations of the **while** loop after v joins W_{low}, v leaves W.

Proof. First we note that by Claim 12 no vertex can ever move from W_{low} to W_{high}. Thus, if v is in W_{low}, v will remain in W_{low} until v leaves W. Suppose v is in W_{low} at the beginning of an iteration of the **while** loop. Because I is an MIS with respect to G_{low}, if v does not join I during this iteration, then v has a neighbor $y \in W \cup B_{high}$ such that a neighbor z of y joins I. As a result, z immediately joins D and c_y is incremented. Thus, during every iteration that v remains in W_{low}, a vertex in $N(v) \cap (W \cup B_{high})$ has its counter incremented. Recall that whenever a vertex has its counter incremented $\delta\alpha$ times, it joins D. Because $v \in W_{low}$, we have that $|N(v) \cap (W \cup B_{high})| \leq \delta\alpha$. Therefore, the event that a vertex in $N(v) \cap (W \cup B_{high})$ has its counter incremented can only happen at most $(\delta\alpha)^2$ times. Thus, v can only remain in W_{low} for $(\delta\alpha)^2$ iterations of the **while** loop. ◁

We will complete the analysis using the fact that enough vertices are in W_{low} at any given point in time. In particular, Claim 6 implies that at least half of the vertices in $W \cup B_{high}$ are in W_{low}. This implies that at least half of the vertices in W are in W_{low}. Formally, we divide the execution of the algorithm into phases where each phase consists of $(\delta\alpha)^2$ iterations of the **while** loop. At the beginning of any phase, at least half of the vertices

in W are in W_{low}. By the end of the phase, all of these vertices have left W by Claim 13. Therefore, each phase witnesses at least half of the vertices in W leaving W. By Claim 12, no vertex can re-enter W, so there can only be $O(\log n)$ phases.

Putting everything together, there are $O(\log n)$ phases, each consisting of $(\delta\alpha)^2$ iterations of the **while** loop, and one iteration of the **while** loop takes $O(R(n))$ rounds. Therefore, the total number of rounds is $O(R(n) \cdot \alpha^2 \log n)$.

For Theorem 10, $R(n) = O(\log n)$, so the number of rounds is $O(\alpha^2 \log^2 n)$.

References

1 Zeyuan Allen-Zhu and Lorenzo Orecchia. Nearly linear-time packing and covering lp solvers. *Mathematical Programming*, 175(1):307–353, 2019.

2 Saeed Akhoondian Amiri. Deterministic congest algorithm for mds on bounded arboricity graphs. *arXiv preprint*, 2021. `arXiv:2102.08076`.

3 Saeed Akhoondian Amiri, Stefan Schmid, and Sebastian Siebertz. Distributed dominating set approximations beyond planar graphs. *ACM Transactions on Algorithms (TALG)*, 15(3):1–18, 2019.

4 Srinivasa R Arikati, Anil Maheshwari, and Christos D Zaroliagis. Efficient computation of implicit representations of sparse graphs. *Discrete Applied Mathematics*, 78(1-3):1–16, 1997.

5 Brenda S Baker. Approximation algorithms for np-complete problems on planar graphs. *Journal of the ACM (JACM)*, 41(1):153–180, 1994.

6 Nikhil Bansal and Seeun William Umboh. Tight approximation bounds for dominating set on graphs of bounded arboricity. *Information Processing Letters*, 122:21–24, 2017.

7 Leonid Barenboim and Michael Elkin. Sublogarithmic distributed mis algorithm for sparse graphs using nash-williams decomposition. *Distributed Computing*, 22(5-6):363–379, 2010.

8 Suman K Bera, Amit Chakrabarti, and Prantar Ghosh. Graph coloring via degeneracy in streaming and other space-conscious models. *arXiv preprint*, 2019. `arXiv:1905.00566`.

9 Suman K. Bera, Noujan Pashanasangi, and C. Seshadhri. Linear time subgraph counting, graph degeneracy, and the chasm at size six. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 38:1–38:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.ITCS.2020.38`.

10 Suman K. Bera and C. Seshadhri. How the degeneracy helps for triangle counting in graph streams. In Dan Suciu, Yufei Tao, and Zhewei Wei, editors, *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, pages 457–467. ACM, 2020. `doi:10.1145/3375395.3387665`.

11 Gerth Stølting Brodal and Rolf Fagerberg. Dynamic representation of sparse graphs. In Frank K. H. A. Dehne, Arvind Gupta, Jörg-Rüdiger Sack, and Roberto Tamassia, editors, *Algorithms and Data Structures, 6th International Workshop, WADS '99, Vancouver, British Columbia, Canada, August 11-14, 1999, Proceedings*, volume 1663 of *Lecture Notes in Computer Science*, pages 342–351. Springer, 1999. `doi:10.1007/3-540-48447-7_34`.

12 Bernard Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *Journal of the ACM (JACM)*, 47(6):1028–1047, 2000.

13 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on computing*, 14(1):210–223, 1985.

14 Miroslav Chlebík and Janka Chlebíková. Approximation hardness of dominating set problems in bounded degree graphs. *Information and Computation*, 206(11):1264–1275, 2008.

15 Andrzej Czygrinow, Michał Hańćkowiak, and Edyta Szymańska. Fast distributed approximation algorithm for the maximum matching problem in bounded arboricity graphs. In *International Symposium on Algorithms and Computation*, pages 668–678. Springer, 2009.

16 Andrzej Czygrinow, Michal Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approximations in planar graphs. In *International Symposium on Distributed Computing*, pages 78–92. Springer, 2008.

17 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*, pages 94–103, 2019.

18 Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered PCP and the hardness of hypergraph vertex cover. *SIAM J. Comput.*, 34(5):1129–1146, 2005. doi:10.1137/S0097539704443057.

19 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 624–633, 2014.

20 Talya Eden, Reut Levi, and Dana Ron. Testing bounded arboricity. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2081–2092. SIAM, 2018. doi:10.1137/1.9781611975031.136.

21 Talya Eden, Dana Ron, and Will Rosenbaum. The arboricity captures the complexity of sampling edges. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 52:1–52:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.52.

22 Talya Eden, Dana Ron, and C. Seshadhri. Faster sublinear approximation of the number of k-cliques in low-arboricity graphs. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1467–1478. SIAM, 2020. doi:10.1137/1.9781611975994.89.

23 David Eppstein. Arboricity and bipartite subgraph listing algorithms. *Information processing letters*, 51(4):207–211, 1994.

24 Guy Even, Mohsen Ghaffari, and Moti Medina. Distributed set cover approximation: Primal-dual with optimal locality. In *32nd International Symposium on Distributed Computing (DISC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

25 Michael Fredman and Michael Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 345–354, 1989.

26 Michael L Fredman and Dan E Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 719–725. IEEE, 1990.

27 Michael R Garey. A guide to the theory of np-completeness. *Computers and intractability*, 1979.

28 Mohsen Ghaffari, Christoph Grunau, and Václav Rozhoň. Improved deterministic network decomposition. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2904–2923. SIAM, 2021.

29 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.

30 Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 784–797, 2017.

31 Mohsen Ghaffari and Hsin-Hao Su. Distributed degree splitting, edge coloring, and orientations. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2505–2523. SIAM, 2017. doi:10.1137/1.9781611974782.166.

32 Gaurav Goel and Jens Gustedt. Bounded arboricity to determine the local structure of sparse graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 159–167. Springer, 2006.

33 Meng He, Ganggui Tang, and Norbert Zeh. Orienting dynamic graphs, with applications to maximal matchings and adjacency queries. In Hee-Kap Ahn and Chan-Su Shin, editors, *Algorithms and Computation - 25th International Symposium, ISAAC 2014, Jeonju, Korea, December 15-17, 2014, Proceedings*, volume 8889 of *Lecture Notes in Computer Science*, pages 128–140. Springer, 2014. doi:10.1007/978-3-319-13075-0_11.

34 Lujun Jia, Rajmohan Rajaraman, and Torsten Suel. An efficient distributed algorithm for constructing small dominating sets. *Distributed Computing*, 15(4):193–205, 2002.

35 David S Johnson. Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 9(3):256–278, 1974.

36 Mark Jones, Daniel Lokshtanov, MS Ramanujan, Saket Saurabh, and Ondřej Suchỳ. Parameterized complexity of directed steiner tree on sparse graphs. In *European Symposium on Algorithms*, pages 671–682. Springer, 2013.

37 Haim Kaplan and Shay Solomon. Dynamic representations of sparse distributed networks: A locality-sensitive approach. *ACM Transactions on Parallel Computing (TOPC)*, 8(1):1–26, 2021.

38 David R Karger, Philip N Klein, and Robert E Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM (JACM)*, 42(2):321–328, 1995.

39 Christos Koufogiannakis and Neal E Young. A nearly linear-time ptas for explicit fractional packing and covering linear programs. *Algorithmica*, 70(4):648–674, 2014.

40 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22–26, 2006*, pages 980–989. ACM Press, 2006. doi:10.5555/1109557.1109666.

41 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *Journal of the ACM (JACM)*, 63(2):1–44, 2016.

42 Fabian Kuhn and Roger Wattenhofer. Constant-time distributed dominating set approximation. *Distributed Computing*, 17(4):303–310, 2005.

43 Christoph Lenzen and Roger Wattenhofer. Minimum dominating set approximation in graphs of bounded arboricity. In Nancy A. Lynch and Alexander A. Shvartsman, editors, *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings*, volume 6343 of *Lecture Notes in Computer Science*, pages 510–524. Springer, 2010. doi:10.1007/978-3-642-15763-9_48.

44 Andrew McGregor and Sofya Vorotnikova. A simple, space-efficient, streaming algorithm for matchings in low arboricity graphs. In Raimund Seidel, editor, *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, volume 61 of *OASICS*, pages 14:1–14:4. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/OASIcs.SOSA.2018.14.

45 Adir Morgan, Shay Solomon, and Nicole Wein. Algorithms for the minimum dominating set problem in bounded arboricity graphs: Simpler, faster, and combinatorial. *arXiv preprint*, 2021. arXiv:2102.10077.

46 Jose C Nacher and Tatsuya Akutsu. Minimum dominating set-based methods for analyzing biological networks. *Methods*, 102:57–63, 2016.

47 Krzysztof Onak, Baruch Schieber, Shay Solomon, and Nicole Wein. Fully dynamic MIS in uniformly sparse graphs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 92:1–92:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.92.

48 Merav Parter, David Peleg, and Shay Solomon. Local-on-average distributed tasks. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 220–239. SIAM, 2016.

49 David Peleg and Shay Solomon. Dynamic $(1+\epsilon)$-approximate matchings: A density-sensitive approach. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 712–729. SIAM, 2016. doi:10.1137/1.9781611974331.ch51.

50 Kent Quanrud. Nearly linear time approximations for mixed packing and covering problems without data structures or randomization. In *Symposium on Simplicity in Algorithms*, pages 69–80. SIAM, 2020.

51 Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 350–363, 2020.

52 Chao Shen and Tao Li. Multi-document summarization via the minimum dominating set. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 984–992, 2010.

53 Shay Solomon and Nicole Wein. Improved dynamic graph coloring. In *26th Annual European Symposium on Algorithms (ESA 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

54 Hsin-Hao Su and Hoa T. Vu. Distributed dense subgraph detection and low outdegree orientation. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPIcs*, pages 15:1–15:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.DISC.2020.15.

55 Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM (JACM)*, 22(2):215–225, 1975.

56 Peng-Jun Wan, Khaled M Alzoubi, and Ophir Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. In *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1597–1604. IEEE, 2002.

57 Neal E Young. Nearly linear-work algorithms for mixed packing/covering and facility-location linear programs. *arXiv preprint*, 2014. arXiv:1407.3015.