Minimum Dominating Set Approximation in Graphs of Bounded Arboricity

Christoph Lenzen and Roger Wattenhofer

Computer Engineering and Networks Laboratory (TIK)
ETH Zurich
{lenzen,wattenhofer}@tik.ee.ethz.ch

Abstract. Since in general it is NP-hard to solve the minimum dominating set problem even approximatively, a lot of work has been dedicated to central and distributed approximation algorithms on restricted graph classes. In this paper, we compromise between generality and efficiency by considering the problem on graphs of small arboricity a. These family includes, but is not limited to, graphs excluding fixed minors, such as planar graphs, graphs of (locally) bounded treewidth, or bounded genus. We give two viable distributed algorithms. Our first algorithm employs a forest decomposition, achieving a factor $\mathcal{O}(a^2)$ approximation in randomized time $\mathcal{O}(\log n)$. This algorithm can be transformed into a deterministic central routine computing a linear-time constant approximation on a graph of bounded arboricity, without a priori knowledge on a. The second algorithm exhibits an approximation ratio of $\mathcal{O}(a \log \Delta)$, where Δ is the maximum degree, but in turn is uniform and deterministic, and terminates after $\mathcal{O}(\log \Delta)$ rounds. A simple modification offers a trade-off between running time and approximation ratio, that is, for any parameter $\alpha \geq 2$, we can obtain an $\mathcal{O}(a\alpha \log_\alpha \Delta)$-approximation within $\mathcal{O}(\log_\alpha \Delta)$ rounds.

1 Introduction

We are interested in the distributed complexity of the minimum dominating set (MDS) problem, a classic both in graph theory and distributed computing. Given a graph, a dominating set is a subset D of nodes such that each node in the graph is either in D, or has a direct neighbor in D. There are various applications where it is beneficial to find dominating sets of small cardinality, for instance in routing. We want to find a minimum dominating set (MDS)—or a dominating set that is not much larger than an MDS—fast, if possible even in constant time.

Regrettably, it has been shown that in general graphs, small dominating sets cannot be computed in constant time [15]. In some special graphs, however, this is simple. In a tree, for instance, we can get a constant approximation of the MDS problem by choosing all inner nodes, because a third of the inner nodes must be in any MDS. In fact, one can approximate an MDS quickly and up to a constant for more general graphs, for instance planar graphs [18]. What

N.A. Lynch and A.A. Shvartsman (Eds.): DISC 2010, LNCS 6343, pp. 510–524, 2010.

about graph classes between planar and general? One property of planar graphs is *sparsity*, i.e., the number of edges is at most linear in the number of nodes. Hence it seems natural to raise the question on which side of the boundary between "easy" and "hard" sparse graphs reside.

Unfortunately, a constant MDS approximation cannot be computed quickly in sparse graphs, as the hardness of the general MDS problem can be enclosed into a small part of the graph whose contribution to the global solution is decisive. Just take $n - \sqrt{n}$ nodes, and connect them as a star. The remaining $\sqrt{n}$ nodes can be connected arbitrarily, yielding a sparse graph with less than $n+\sqrt{n}^2 = 2n$ edges. Whereas the star is dominated by its center alone, we can just apply the lower bound [15] to the remaining $\sqrt{n}$ nodes.

Consequently, we need a different definition of sparseness, one that applies "everywhere" in the graph. In this work we consider graphs of bounded *arboricity*, subsuming planar graphs, graphs of bounded genus or treewidth, and, more generally, graphs excluding any fixed minor. The arboricity of a graph can be defined in two equivalent ways: (*i*) the minimum number of forests into which the edge set can be partitioned and (*ii*) the maximum ratio of edges to nodes in any subgraph. We present two distributed algorithms, each of which exploits one of these properties. Generally speaking, both algorithms run in logarithmic time and feature an approximation guarantee that depends on the arboricity of the graph. This can be seen as the result of a balancing act between generality of feasible inputs on the one hand and approximation quality and running time on the other hand: On graphs of small arboricity, we outperform the so far best algorithm designed for general graphs; faster routines achieving similar approximations are currently known only for severely constrained inputs like planar graphs or graphs of bounded growth.

As a corollary, we observe that based on a forest decomposition, a central algorithm can compute a constant-factor approximation in a linear number of operations in any graph of bounded arboricity. In contrast, asymptotically optimum solutions are intractable in general graphs (and thus also on sparse graphs), where it is known to be NP-hard to obtain any sublogarithmic approximation ratio [23]. To the best of our knowledge, graphs of bounded arboricity represent the so far most general family of graphs for which a constant MDS approximation can be computed efficiently. Finally, given that all our algorithms are simple and the distributed routines require only small messages, we believe them to be suitable for use in practice.

2 Related Work

It is safe to say that computing small dominating sets is one of the most classical and fundamental graph problems. The task of finding a minimum dominating set was among the first to be recognized as NP-complete [11] and—dominating set being a quite general special case of set cover—Raz and Safra proved that it is NP-hard to achieve a $c \log \Delta$-approximation, where Δ is the maximum node degree and $c > 0$ a constant [23]. Ironically, a $((1 - o(1)) \log \Delta)$-approximation

can be obtained by a naive greedy algorithm, a strategy which was shown to be optimal for polynomial-time algorithms unless NP is contained in the class of problems that can be solved deterministically in time $n^{\mathcal{O}(\log\log n)}$ (where n is the number of nodes) by Feige [10].

Arguably, for distributed algorithms things are even worse. Even if message size is unbounded and nodes may perform arbitrary local computations (in particular solve NP-hard problems!), $\Omega(\log \Delta / \log\log \Delta)$ and $\Omega(\log n / \log\log n)$ communication rounds are required to guarantee a polylogarithmic approximation [15]. Currently, the best known randomized algorithm yields an expected[1] $\mathcal{O}(\log \Delta)$-approximation in $\mathcal{O}(\log n)$ time [17]. Considering that a better approximation ratio is (supposing P≠NP, of course) intractable, this is optimal up to a factor of $\mathcal{O}(\log n \log\log \Delta / \log \Delta)$ in time complexity. However, for this algorithm no non-trivial bound on the message size holds. The authors give a second algorithm whose messages are of size at most $\mathcal{O}(\log \Delta)$, trading in for a time complexity of $\mathcal{O}(\log^2 \Delta)$. To the best of our knowledge, the deterministic distributed complexity of MDS approximations on general graphs is more or less a blind spot, as so far neither fast (polylogarithmic time) algorithms nor stronger lower bounds are known.

In light of the strong negative results, it is natural to consider restricted families of graphs. Here we get a colorful picture. The $((1-o(1))\log \Delta)$-hardness result of Feige has been extended to bipartite and split graphs by Chlebík and Chlebíková [4], and thus also to chordal graphs and their complements. The case of trees is trivial for centralized algorithms. Distributed algorithms need time in the order of the depth of the tree for an optimal solution, while a constant approximation is also trivial. For series-parallel graphs a linear-time centralized algorithm was devised by Takamizawa et al. [25]. This algorithm generalizes to a polynomial-time one in graphs of bounded treewidth.

More interesting are less extreme cases. For instance, both in planar and unit disk graphs, it remains NP-complete to solve MDS exactly [5,11], but PTAS have been given [2,13]. These graph classes are also comparatively well understood in the distributed setting. On unit disk graphs, or more generally the family of graphs of bounded growth,[2] a constant approximation can be found in $\mathcal{O}(\log^* n)$ rounds [24]. The respective algorithm outputs a maximal independent set (which must also be a dominating set); because the number of independent neighbors of any node in an optimal solution is bounded by a constant, so is the approximation ratio of the algorithm. Using the same argument, a constant approximation can be found on graphs of bounded independence, however, the so far best known maximal independent set algorithms on this graph class have randomized running time $\mathcal{O}(\log n)$ with high probability [1,21]. For deterministic algorithms on unit disk graphs, the product of approximation ratio and running time cannot be in $o(\log^* n)$ [19], implying that the upper bound from [24] is

[1] This can be improved to hold with high probability [14].

[2] "Bounded independence" means that the number of independent nodes in neighborhoods is constant. "Bounded growth" refers to the stronger property that the number of independent nodes in an r-neighborhood is polynomially bounded in r.

asympotically optimal. The same result also implies a constant lower bound on the approximation ratio of deterministic distributed algorithms running on planar graphs in $o(\log^* n)$ rounds, which was proved independently in [6]. On the other hand, a deterministic constant-time algorithm that outputs a 74-approximation on planar graphs was given in [18]. Shortly thereafter, Czygrinow et al. devised a $(1+\varepsilon)$-approximation on planar graphs [6] (for any constant $\varepsilon > 0$) in $\mathcal{O}(\log^* n)$ time. Their algorithm utilizes a different constant-time approximation for planar graphs with a constant, but much larger approximation ratio than in [18]; intriguingly, a closer look reveals that this routine does not require planarity at all, but works for any graph free of $K_{3,3}$-minors. Earlier, two of the authors of this work gave slower (polylogarithmic time with large exponent) algorithms yielding $(1+o(1))$-approximations in graphs excluding any fixed minor [7,8]. It should be mentioned, though, that the algorithms from [7,8,6] have in common that they compute optimal solutions to subproblems on subgraphs of bounded diameter, implying that NP-hard problems are solved. Therefore, the respective results are mainly theoretic statements on distributed time complexity and probably infeasible in practice.

Returning to centralized algorithms, Baker's PTAS for planar graphs [2] is based on the fact that planar graphs admit an $\mathcal{O}(D)$ tree decomposition in time $\mathcal{O}(Dn)$, where D denotes the diameter of the graph. Building on Baker's ideas, Eppstein extended the approach to minor-closed families excluding a specific apex graph [9]. Finally Grohe generalized the technique further, giving PTAS for any minor-closed family that does not contain all minors [12].

3 Contribution

In this work, we give practical approximation algorithms whose approximation ratios depend on the arboricity of the underlying graph, i.e., the minimum number of forests into which the edge set can be decomposed. The family of graphs of bounded arboricity contains all graphs excluding fixed minors, thus including planar graphs, graphs of bounded genus, and graphs of bounded tree-width. It is a proper superset of the family of graphs excluding some minor, as already graphs of arboricity 2 may have $K_{\sqrt{n}}$-minors.[3]

Our first distributed algorithm computes, given a forest decomposition into f forests, an f^2-approximation in randomized time $\mathcal{O}(\log n)$ with high probability. The algorithms of Czygrinow et al. [6,7,8] also involve forest decompositions, but they rely on additional properties of the underlying graph and are considerably slower. Barenboim and Elkin applied essentially the same technique as Czygrinow et al. to obtain forest decompositions [3]. However, they stated slightly different and more general results, better fitting our needs. More precisely, their

[3] In contrast, graphs of bounded independence are fundamentally different. A clique has maximum arboricity, but independent sets are of size one, while a star has arboricity one, but the neighborhood of the center consists of $n-1$ independent nodes. The combination of bounded independence *and* arboricity implies bounded degree and vice versa.

algorithms compute $\Theta(a)$-forest decompositions of graphs of arboricity a in time $\mathcal{O}(\log n)$ provided that an upper bound in $\mathcal{O}(a)$ on a or an upper bound in $\mathcal{O}(n^c)$ on n (for a constant $c \geq 1$) is known to the nodes. Employing their algorithm, we thus get an $\mathcal{O}(a^2)$-approximation in $\mathcal{O}(\log n)$ time on any graph of arboricity a. In particular, the resulting routine guarantees constant approximation ratios on graphs of bounded arboricity, using messages of size $\mathcal{O}(\log n)$. From a coloring lower bound by Linial [20], Barenboim and Elkin inferred a lower bound of $\Omega(\log n / \log f)$ on the time to compute a forest decomposition into f forests distributedly. Thus, no algorithm utilizing a forest decomposition can yield substantially better results. Using standard techniques, our algorithm can also be transformed into an efficient central routine. In this case, we can remove the logarithmic overhead in running time and the need for randomization, and we need no a-priori knowledge on the arboricity of the graph. Hence, we achieve a uniform, central $\mathcal{O}(a)$-approximation within $\mathcal{O}(an)$ steps, i.e., a linear time constant approximation on graphs of bounded arboricity.

We proceed by presenting a second, deterministic distributed algorithm that features a smaller running time of $\mathcal{O}(\log \Delta)$, smaller messages of size $\mathcal{O}(\log \log \Delta)$, and is uniform, i.e., does not require any bounds on a or n as input. Interestingly, this algorithm requires less symmetry breaking than the first one (which apart from the forest decomposition computes a maximal independent set). Indeed, a port numbering suffices, and if (an upper bound on) a is known to the algorithm, also this condition can be dropped, i.e., the modified algorithm does not rely on any non-topological symmetry breaking information at all. These advantages come not for free, as the approximation ratio of the second algorithm now depends on the maximum degree, being $\mathcal{O}(a \log \Delta)$. A simple modification of the algorithm permits to reduce the running time to $\mathcal{O}(\log \Delta / \log \alpha)$, but at the expense weakening the approximation guarantee to $\mathcal{O}(a\alpha \log \Delta / \log \alpha)$ (for any $\alpha \geq 2$). We give an example where the approximation ratio of the second algorithm is matched, i.e., better approximation guarantees require different techniques.

4 Constant-Factor Approximation

In this section, we present an algorithm that computes a dominating set at most $\mathcal{O}(a^2)$ larger than optimum. After introducing the employed standard model of distributed computation and some preliminary definitions, we proceed by presenting the algorithm. After proving its approximation ratio, we review how to obtain a $(2a)$-forest decomposition of a graph of arboricity a in $\mathcal{O}(an)$ central operations. From these results, we conclude that on graphs of bounded arboricity constant MDS approximations can be computed in $\mathcal{O}(\log n)$ distributed rounds and $\mathcal{O}(n)$ central steps.

4.1 Model

We model a distributed system as a simple, undirected graph $G = (V, E)$, where edges represent bidirectional communication links. Algorithms proceed in

synchronous rounds. In each round, nodes (*i*) send messages to their neighbors, (*ii*) receive messages sent by their neighbors, and (*iii*) perform arbitrary (but finite) local computations. Furthermore, we permit randomization, i.e., nodes have access to an unlimited source of unbiased random bits. Initially, any node v knows its *inclusive neighborhood* $\mathcal{N}_v^+ := \{v\} \cup \{w \in V \mid \{v, w\} \in E\}$. Similarly, by $\mathcal{N}_S^+ := \bigcup_{s \in S} \mathcal{N}_s^+$ we denote the inclusive neighborhood of a set $S \subseteq V$.

Before we describe our first algorithm, we need to formalize some well-known graph theoretic concepts.

Definition 1 (Dominating Sets). *A* dominating set *is a subset of the nodes $D \subseteq V$ such that $\mathcal{N}_D^+ = V$. A* minimum dominating set (MDS) *is a dominating set of minimum cardinality. An α-approximation to an MDS is a dominating set of size at most $\alpha|M|$, where M is an MDS.*

Definition 2 (Independent Sets). *An* independent set *is a subset of the nodes $I \subseteq V$ containing no neighbors. A* maximal independent set (MIS) *is an independent set for which adding any node destroys independence, i.e., $\mathcal{N}_I^+ = V$ and I is also a dominating set.*

Definition 3 (Forests and Forest Decompositions). *A* forest *is a graph containing no cycles. An* oriented forest *is a forest where edges have been oriented such that outdegrees are at most one. If $(v, w) \in E$ for two nodes in the forest, w is called* parent *of v and v is a* child *of w. An f-*forest decomposition *of a graph G is a decomposition of the edge set $E = E_1 \dot\cup E_2 \dot\cup \ldots \dot\cup E_f$ together with an appropriate orientation of the edges such that the subgraphs induced by each E_i, $i \in \{1, \ldots, f\}$, are oriented forests. Given a forest decomposition, we denote by $P(v) := \{w \in V \mid (v, w) \in E\}$ and $P(S) := \bigcup_{s \in S} P(s)$ the set of parents of the node $v \in V$ and the set $S \subseteq V$, respectively. The* arboricity *$a(G)$ of G is the minimal number of forests in a forest decomposition of G. In the forthcoming, we will write a instead of $a(G)$ whenever the argument is clear from the context.*

4.2 Algorithm

Our first algorithm is based on the following observations. Given an f-forest decomposition and an MDS M, the nodes can be partitioned into two sets. One set contains the nodes which are dominated by a parent, the other contains the remaining nodes, which thus are either themselves in M or have a child in M. Since dominating set nodes can cover only f parents, the latter are at most $(f + 1)|M|$ many nodes. If each such node elects all its parents into the dominating set, we have chosen at most $f(f + 1)|M|$ nodes.

For the first set, we can exploit the fact that each node has at most f parents in a more subtle manner. Covering the nodes in this set by parents only, we need to solve a special case of set cover where each element is part of at most f sets. Such instances can be approximated well by a simple sequential greedy algorithm: Pick any element that is not yet covered and add *all* sets containing it; repeat this until no element remains. Since in each step we add at least one new set from an optimum solution, we get an f-approximation. This strategy

can be parallelized by computing a maximal independent set in the graph where two nodes are adjacent exactly if they share a parent, as adding the parents of the nodes in an independent set in any order would be a feasible execution of the sequential greedy algorithm.

Putting these two observations together, first all parents of nodes from a maximal independent set in a helper graph are elected into the dominating set. In this helper graph, two nodes are adjacent if they share a parent. Afterwards, the remaining uncovered nodes have no parents, therefore it is uncritical to select them all. This approach is summarized in Algorithm 1.

Algorithm 1. Parent Dominating Set

input : f-forest decomposition of G
output: dominating set D
1 $H := (V, F)$, where $\{v, w\} \in F \Leftrightarrow P(v) \cap P(w) \neq \emptyset$ // neighbors in H share a parent in G
2 Compute a maximal independent set I on H
3 $D := P(I)$ // all parents of nodes in the independent set join D
4 $D := D \cup (V \setminus \mathcal{N}_D^+)$ // all still uncovered nodes may safely join D

4.3 Analysis

Lemma 1. *In Line 1 of Algorithm 1, at most $f(f+2)|M|$ many nodes enter D, where M denotes an MDS of G.*

Proof. Denote by $V_c \subseteq V$ the set of nodes that have a child in M or are themselves in M. We have that $|V_c| \leq (f+1)|M|$, since no node has more than f parents. Each such node adds at most f parents to D in Line 1 of the algorithm, i.e., in total at most $f(f+1)|M|$ many nodes join D because they are elected by children in $I \cap V_c$.

Now consider the set of nodes $V_p \subset V$ that have at least one parent in M, i.e., in particular the nodes in $I \cap V_p$ have at least one parent in M. By the definition of F and the fact that I is an independent set, no node in M can have two children in I. Thus, $|I \cap V_p| \leq |M|$. Since no node has more than f parents, we conclude that at most $f|M|$ many nodes join $|D|$ after being marked as candidate by a child in $I \cap V_p$.

Finally, observe that since M is a dominating set, we have that $V_c \cup V_p = V$ and thus

$$|D| \leq f|I \cap V_c| + f|I \cap V_p| \leq f(f+1)|M| + f|M| = f(f+2)|M| \ ,$$

concluding the proof. □

Theorem 1. *Algorithm 1 outputs a dominating set D containing at most $(f^2 + 3f + 1)|M|$ many nodes, where M is an optimum solution.*

Proof. By Lemma 1, at most $f(f+2)|M|$ nodes enter D in Line 1 of the algorithm. Since I is a MIS in H, all nodes that have a parent are adjacent to at least one node in D after Line 1. Hence, the nodes selected in Line 1 must either be covered by a child or themselves be in M. As no node has more than f parents, thus in Line 1 at most $(f+1)|M|$ many nodes join D. Altogether, at most $(f^2+3f+1)|M|$ many nodes may end up in D as claimed. □

Corollary 1. *In any graph G, a factor $\mathcal{O}(a(G)^2)$ approximation to the MDS problem can be computed distributedly in $\mathcal{O}(\log n)$ rounds with high probability.*[4] *In particular, on graphs of bounded arboricity a constant-factor approximation can be obtained in $\mathcal{O}(\log n)$ rounds with high probability. This can be accomplished with messages of size $\mathcal{O}(\log n)$.*

Proof. We run Algorithm 1 in a distributed fashion. To see that this is possible, observe that (*i*) nodes need only to know whether a neighbor is a parent or a child, (*ii*) that H can be constructed locally in 2 rounds and (*iii*) a synchronous round in H can be simulated by two rounds on G. Thus, we simply may pick distributed algorithms to compute a forest decomposition of G and a maximal independent set and plug them together to obtain a distributed variant of Algorithm 1.

For the forest decomposition, we employ the algorithm from [3], yielding a decomposition into $\mathcal{O}(a)$ forests in $\mathcal{O}(\log n)$ rounds. A maximal independent set can be computed in $\mathcal{O}(\log n)$ rounds with high probability by well-known algorithms [1,21], or a more recent similar technique [22]. In total the algorithm requires $\mathcal{O}(\log n)$ rounds with high probability and according to Theorem 1 the approximation guarantee is $\mathcal{O}(a)$.

Regarding message size, we need to check that we do not require large messages because we compute a MIS on H. Formulated abstractly, the algorithm from [22] breaks symmetry by making each node still eligible for the independent set choosing a random value in each round and permitting it to join the independent set if its value is a local minimum. This concept can for instance be realized by taking $\mathcal{O}(\log n)$ random bits as encoding of some number and comparing it to neighbors. The respective values will with high probability differ. This approach can be emulated using messages of size $\mathcal{O}(\log n)$ in G: Nodes send their random values to all parents in the forest decomposition, which then forward only the smallest values to all children.[5] □

4.4 Linear Time Central Algorithm

Employing well-known techniques, a central algorithm can compute a suitable forest decomposition with linear complexity.

[4] I.e., with probability at least $1 - 1/n^c$ for an arbitrary, but fixed constant $c > 0$.

[5] If (an upper bound on) n is not known, one can start with constantly many bits and double the number of used bits in each round where two nodes pick the same value. This will not slow down the algorithm significantly and bound message size by $\mathcal{O}(\log n)$ with high probability.

Lemma 2. *A $2a(G)$-forest decomposition of G can be computed in $\mathcal{O}(|E|+n) \subseteq \mathcal{O}(an)$ computational steps.*

Proof. For each node, we compute and store its degree ($\mathcal{O}(|E|)$ steps). Now we place the nodes into buckets according to their degree. We pick a node with smallest degree, we orient its edges, delete them, and update the assignment of the nodes to the buckets. This is repeated until no more nodes remain. Assuming appropriate data structures, the number of operations will be bounded by $\mathcal{O}(|E| + n) \subseteq \mathcal{O}(an)$, as each edge and node is accessed a constant number of times. Since a graph of arboricity a and n' nodes has less than an' edges, the smallest degree of any subgraph of G is at most $2a(G)$. Hence we obtain a forest decomposition into $2a(G)$ forests. □

Hence, for any graph of arboricity $a \in \mathcal{O}(1)$, a deterministic, central algorithm can compute an $\mathcal{O}(1)$-approximation to the MDS problem with linear complexity.

Corollary 2. *Deterministically, an $\mathcal{O}(a)$-approximation to an MDS can be computed in $\mathcal{O}(|E| + n) \subseteq \mathcal{O}(an)$ central steps.*

Proof. Using Lemma 2, we can compute a forest decomposition within the stated complexity bounds. In a central setting, Algorithm 1 can easily be implemented using $\mathcal{O}(|E| + n)$ steps. The approximation guarantee follows from Theorem 1. □

5 A Solution in the Port Numbering Model

Algorithm 1 might be unsatisfactory with regard to several aspects. Its running time is logarithmic in n even if the maximum degree Δ is small. This cannot be improved upon by any approach that utilizes a forest decomposition, as a lower bound of $\Omega(\log n / \log f)$ is known on the time to compute a forest decomposition into f forests [3]. The algorithm is not uniform, as it necessitates global knowledge of a bound on $a(G)$ or n.

Moreover, the algorithm requires randomization in order to compute a MIS quickly. Considering deterministic algorithms, one might pose the question how much initial symmetry breaking information needs to be provided to the nodes. While randomized algorithms may randomly generate unique identifiers of size $\mathcal{O}(\log n)$ in constant time with high probability, many deterministic algorithms assume them to be given as input. Milder assumptions are the ability to distinguish neighbors by means of a port numbering and/or an initial orientation of the edges.

In this section, we show that an uniform, deterministic algorithm exists that requires a port numbering only, yet achieves a running time of $\mathcal{O}(\log \Delta)$ and a good approximation ratio. The size of the computed dominating set is bounded linearly in the product of the arboricity $a(G)$ of the graph and the logarithm of the maximum degree Δ. Interestingly, we will observe later that if (an upper bound on) the arboricity is known to the algorithm, one can even drop the assumption of a port numbering.

5.1 The Port Numbering Model

In this section, we consider the so-called *port numbering model*. Again an undirected and simple graph $G = (V, E)$ is given. Communication is synchronous and computation is deterministic. Nodes refer to their neighbors by means of a port numbering, i.e., each node v uniquely maps the numbers $\{1, \ldots, \delta\}$ to its edges (where δ is the degree of v). Nodes can distinguish from which of their neighbors they received a specific message. However, in contrast to the previous setting where nodes had unique identifiers, different nodes now may refer to the same destination by different port numbers.

Note that this model is quite harsh. For instance, it is impossible to reliably discover cycles, compute a MIS, or determine the diameter of the graph.

5.2 Algorithm

The basic idea of Algorithm Greedy-by-Degree (Algorithm 2) is that it is always feasible to choose nodes of high *residual degree* (by which we mean the number of uncovered nodes in the inclusive neighborhood) simultaneously, i.e., all the nodes that cover up to a constant factor as many nodes as the one covering the most uncovered nodes. This permits to obtain strong approximation guarantees without the structural information provided by knowledge of $a(G)$ or a forest decomposition; the mere fact that the graph must be "locally sparse" enforces that if many nodes are elected into the set, also the dominating set must be large. A difficulty arising from this approach is that nodes are not aware of the current maximum residual degree in the graph. Hence, every node checks whether there is a node in its 2-hop neighborhood having a residual degree larger by a factor 2. If not, nodes may join the dominating set (even if their degree is not large from a global perspective), implying that the maximum residual degree drops by a factor of 2 in a constant number of rounds.

A second problem occurs once residual degrees become small. In fact, it may happen that a huge number of already covered nodes can each dominate the same small set of $a(G) - 1$ nodes. For this reason, it is mandatory to ensure that not more nodes may join the dominating set than actually need to be covered. To this end, nodes that still need to be covered elect one of their neighbors (if any) that are feasible according to the criterion of (locally) large residual degree. This scheme is described in Algorithm 2.

Note that nodes may never leave D once they entered it. Thus, nodes may terminate based on local knowledge only when executing the algorithm, as they can cease executing the algorithm as soon as $\delta_v = 0$, i.e., their complete neighborhood is covered by D. Moreover, it can easily be verified that one iteration of the loop can be executed by a local algorithm in the port numbering model using 6 rounds.

5.3 Analysis

In the sequel, when we talk of a *phase* of Algorithm 2, we refer to a complete execution of the while loop. We start by proving that not too many nodes with small residual degrees enter D.

Algorithm 2. Greedy-by-Degree.

```
   output: dominating set D
 1 D := ∅
 2 while V ≠ N_D^+ do
 3     C := ∅ // candidate set
 4     for v ∈ V in parallel do
 5         δ_v := |N_v^+ \ N_D^+| // residual degree
 6         Δ_v := max_{w∈N_v^+} {δ_w} // maximum residual degree within one hop
 7         Δ_v := max_{w∈N_v^+} {Δ_w} // maximum residual degree within two hops
 8         if ⌈log δ_v⌉ ≥ ⌈log Δ_v⌉ then
 9             C := C ∪ {v}
10         end
11         if v ∈ N_C^+ \ N_D^+ then
12             w := any node from C ∩ N_v^+ (break symmetry by port numbers)
13             D := D ∪ {w} // uncovered nodes select a candidate joining D
14         end
15     end
16 end
```

Lemma 3. *Denote by M an MDS. During the execution of Algorithm 2, in total at most $16a|M|$ nodes join D in Line 2 of the algorithm after computing $\delta_v \leq 8a$ in Line 2 of the same phase.*

Proof. Fix a phase of the algorithm. Consider the set S consisting of all nodes $v \in V$ that become covered in this phase by some node $w \in \mathcal{N}_v^+$ that computes $\delta_w \leq 8a$ and joins D. As according to Line 2 nodes join D subject to the condition that residual degrees throughout their 2-hop neighborhoods are less than twice as large as their own, no node $m \in M$ can cover more than $16a$ many nodes in S. Hence, $|S| \leq 16a|M|$. Because of the rule that a node needs to be elected by a covered node in order to enter D, this is also a bound on the number of nodes joining D in a phase when they have residual degree at most $8a$. □

Next, we show that in each phase, at most a constant factor more nodes of large residual degree are chosen than are in an MDS.

Lemma 4. *If M is an MDS, in each phase of Algorithm 2 at most $16a|M|$ nodes v that compute $\delta_v > 8a$ in Line 2 join D in Line 2.*

Proof. Denote by D' the nodes $v \in V$ joining D in Line 2 of a phase in which they computed $\delta_v > 8a$ and by V' the set of nodes that had not been covered at the beginning of this phase. Define for $i \in \{0, \dots, \lceil \log n \rceil\}$ that

$$M_i := \{v \in M \mid \delta_v \in (2^{i-1}, 2^i]\}$$
$$V_i := \left\{ v \in V' \;\middle|\; \max_{w \in \mathcal{N}_v^+} \{\delta_w\} \in (2^{i-1}, 2^i] \right\}$$
$$D_i := \{v \in D' \mid \delta_v \in (2^{i-1}, 2^i]\} \ .$$

Note that $\bigcup_{i=\lceil \log 8a \rceil}^{\lceil \log n \rceil} D_i = D'$.

Consider any $j \in \{\lceil \log 8a \rceil, \ldots, \lceil \log n \rceil\}$. By definition, nodes in V_j may only be covered by nodes from M_i for $i \leq j$. Thus, $\sum_{i=0}^{j} 2^i |M_i| \geq |V_j|$.

Nodes $v \in D_j$ cover at least $2^{j-1}+1$ nodes from the set $\bigcup_{i \in \{j,\ldots,\lceil \log n \rceil\}} V_i$, as by definition they have no neighbors in V_i for $i < j$. On the other hand, Lines 2 to 2 of the algorithm impose that these nodes must not have any neighbors of residual degree larger than $2^{\lceil \log \delta_v \rceil} = 2^j$, i.e., these nodes cannot be in a set V_i for $i > j$. Hence, each node $v \in D_j$ has at least $2^{j-1}+1$ neighbors in V_j. This observation implies that the subgraph induced by $D_j \cup V_j$ has at least $2^{j-2}|D_j| \geq 2a|D_j|$ edges. On the other hand, by definition of the arboricity, this subgraph has less than $a(|D_j| + |V_j|)$ edges. It follows that

$$|D_j| \leq \frac{a|V_j|}{2^{j-2} - a} \leq 2^{3-j} a |V_j| \leq 2^{3-j} a \sum_{i=0}^{j} 2^i |M_i| \ .$$

We conclude that

$$\begin{aligned} \sum_{j=\lceil \log 8a \rceil}^{\lceil \log n \rceil} |D_j| &\leq \sum_{j=\lceil \log 8a \rceil}^{\lceil \log n \rceil} 2^{3-j} a \sum_{i=0}^{j} 2^i |M_i| \leq 8a \sum_{j=0}^{\lceil \log n \rceil} \sum_{i=0}^{j} 2^{i-j} |M_i| \\ &< 8a \sum_{i=0}^{\lceil \log n \rceil} \sum_{j=i}^{\infty} 2^{i-j} |M_i| = 16a \sum_{i=0}^{\lceil \log n \rceil} |M_i| = 16a|M| \ , \end{aligned}$$

as claimed. □

We now can bound the approximation quality of the algorithm.

Theorem 2. *Assume that G has maximum degree Δ. Then an execution of Algorithm 2 on G terminates within $6\lceil \log(\Delta+1) \rceil$ rounds and outputs a dominating set at most a factor $16a(G) \log \Delta$ larger than optimum. The worst-case approximation ratio of the algorithm is $\Theta(a(G) \log \Delta)$. Message size can be bounded by $\mathcal{O}(\log \log \Delta)$.*

Proof. We first examine the running time of the algorithm. Denote by $\Delta(i)$ the maximum residual degree after the i^{th} phase, i.e., $\Delta(0) = \Delta + 1$ (as a node also covers itself). As observed earlier, each phase of Algorithm 2 takes six rounds. Because all nodes v computing a δ_v satisfying $\lceil \log \delta_v \rceil = \lceil \log \Delta(i) \rceil$ join C in phase i and any node in $\mathcal{N}_C^+$ becomes covered, we have that $\lceil \log \Delta(i+1) \rceil \leq \lceil \log \Delta(i) \rceil - 1$ for all phases i. Since the algorithm terminates at the end of the subsequent phase once $\Delta(i) \leq 2$, in total at most $\lceil \log \Delta(0) \rceil = \lceil \log(\Delta+1) \rceil$ phases are required.

Having established the bound on the running time of the algorithm, its approximation ratio directly follows[6] by applying Lemmas 3 and 4. The bound on the message size follows from the observation that in each phase nodes need to exchange residual degrees rounded to powers of 2 and a constant number of binary values only.

[6] Note that in the last three phases the maximum degree is at most $8 \leq 8a$.

For the lower bound of $\Omega(a \log \Delta)$, we briefly sketch appropriate input graphs. We start with $a(G)$ being constant. For $k \in \mathbb{N}$, $k \geq 2$ an optimal solution consists of 4 nodes of degree 2^{k-1}. Half of their inclusive neighborhood is covered by a node having degree 2^k, another quarter by a node of degree 2^{k-1}, one eighth by a node of degree 2^{k-2}, and so on. In each phase, the algorithm will choose one of the latter nodes, namely the one of highest degree. Thus it will choose $k \in \Theta(\log \Delta)$ nodes, whereas an optimal solution contains $4 \in \mathcal{O}(1)$ nodes. This bound extends to arbitrarily large values of n by replicating this graph sufficiently often. The described graph has constant arboricity, as each node is covered at most constantly often. Similarly, we could decide to cover each node multiple times by nodes in the suboptimal solution computed by the algorithm (i.e., half of the neighborhood of a node in the optimal solution is covered by $\Theta(a)$ nodes, a quarter by another $\Theta(a)$ nodes, etc.). Hence, for any value of a and arbitrarily large n, we can construct a graph of n nodes where Algorithm 2 computes a solution by factor $\Omega(a \log \Delta)$ worse than the optimum. □

Like with the algorithms by Kuhn et al. [16,17], we can sacrifice accuracy in order to speed up the computation.

Corollary 3. *For any $\alpha \geq 2$, Algorithm 2 can be modified such that it has a running time of $\mathcal{O}(\log_\alpha \Delta)$ and approximation ratio $\mathcal{O}(a(G)\alpha \log_\alpha \Delta)$.*

Proof. We simply change the base of the logarithms in Line 2 of the algorithm, i.e., instead of rounding residual degrees to integer powers of two, we round to integer powers of α. Naturally, this linearly affects the approximation guarantees. In the proof of Lemma 4, we just replace the respective powers of 2 by α as well, yielding a bound of $\mathcal{O}(a(G) + \alpha \log_\alpha \Delta)$ on the approximation ratio by the same reasoning as in Theorem 2. □

If it was not for the computation of a MIS, we could speed up Algorithm 1 in almost the same manner (accepting a forest decomposition into a larger number of forests). However, the constructed helper graph is of bounded independence, but not arboricity or growth. For this graph class currently no distributed algorithm computing a MIS in time $o(\log n)$ is known.

We conclude this section with some final remarks. If nodes know $a(G)$ (or a reasonable upper bound), a port numbering is not required anymore. In this case, nodes will join D without the necessity of being elected by a neighbor, however only if the prerequisite $\delta_v > 8a$ is satisfied. To complete the dominating set, uncovered nodes may join D independently of δ_v once their neighborhood contains no more nodes of residual degree larger than $8a$. It is not hard to see that with this modification, essentially the same analysis as for Algorithm 2 applies, both with regard to time complexity and approximation ratio.

6 Conclusion

We presented efficient distributed minimum dominating set approximation algorithms for graphs of bounded arboricity. Compared to the best known solution for general graphs using reasonably sized messages, we can either (i) reduce the

running time by a factor of $\Theta(\log \Delta)$ (Algorithm 2) or (*ii*) change the running time from $\Theta(\log^2 \Delta)$ to $\Theta(\log n)$ and improve on the approximation quality by a factor of $\Theta(\log \Delta)$ (Algorithm 1). Moreover, our algorithms provide deterministic approximation guarantees. The Algorithms by Kuhn et al. [16] utilize randomized rounding, thus yielding probabilistic bounds. The faster of the two given algorithms is entirely deterministic. In contrast, Algorithm 1 necessitates the computation of a maximal independent set on a graph of bounded independence; deterministic solutions of running time linear in Δ are known, but a respective variant of the algorithm would be slow in comparison.

Of independent interest might be that on graphs of bounded arboricity, a central version of Algorithm 1 obtains a constant-factor approximation within a linear number of operations, without the need for randomization. In summary, minimum dominating sets are substantially easier to approximate on graphs of small arboricity, for distributed as well as for central algorithms.

Acknowledgements

We would like to thank Jukka Suomela for inspiring discussions and Topi Musto and the anonymous reviewers for many valuable comments that helped to improve this paper.

References

1. Alon, N., Babai, L., Itai, A.: A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. J. Algorithms 7(4), 567–583 (1986)
2. Baker, B.S.: Approximation Algorithms for NP-Complete Problems on Planar Graphs. J. ACM 41(1), 153–180 (1994)
3. Barenboim, L., Elkin, M.: Sublogarithmic Distributed MIS algorithm for Sparse Graphs using Nash-Williams Decomposition. Distributed Computing, 1–17 (2009)
4. Chlebk, M., Chlebkov, J.: Approximation Hardness of Dominating Set Problems in Bounded Degree Graphs. Information and Computation 206(11), 1264–1275 (2008)
5. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit Disk Graphs. Discrete Math. 86(1-3), 165–177 (1990)
6. Czygrinow, A., Hańćkowiak, M., Wawrzyniak, W.: Fast Distributed Approximations in Planar Graphs. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 78–92. Springer, Heidelberg (2008)
7. Czygrinow, A., Hańćkowiak, M.: Distributed Almost Exact Approximations for Minor-Closed Families. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 244–255. Springer, Heidelberg (2006)
8. Czygrinow, A., Hanckowiak, M.: Distributed Approximation Algorithms for Weighted Problems in Minor-Closed Families. In: Lin, G. (ed.) COCOON 2007. LNCS, vol. 4598, pp. 515–525. Springer, Heidelberg (2007)
9. Eppstein, D.: Diameter and Treewidth in Minor-Closed Graph Families. Algorithmica 27(3), 275–291 (2000)
10. Feige, U.: A Threshold of *lnn* for Approximating Set Cover. J. ACM 45(4), 634–652 (1998)

11. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
12. Grohe, M.: Local Tree-Width, Excluded Minors, and Approximation Algorithms. Combinatorica 23(4), 613–632 (2003)
13. Hunt, H.B., Marathe, M.V., Radhakrishnan, V., Ravi, S.S., Rosenkrantz, D.J., Stearns, R.E.: NC-Approximation Schemes for NP- and PSPACE-Hard Problems for Geometric Graphs. Journal of Algorithms 26(2), 238–274 (1998)
14. Kuhn, F.: Personal Communication (2010)
15. Kuhn, F., Moscibroda, T., Wattenhofer, R.: What Cannot Be Computed Locally! In: Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing, PODC (2004)
16. Kuhn, F., Moscibroda, T., Wattenhofer, R.: The Price of Being Near-Sighted. In: Proc. 17th ACM-SIAM Symposium on Discrete Algorithms, SODA (2006)
17. Kuhn, F., Wattenhofer, R.: Constant-Time Distributed Dominating Set Approximation. Distrib. Comput. 17(4), 303–310 (2005)
18. Lenzen, C., Oswald, Y.A., Wattenhofer, R.: What can be Approximated Locally? In: 20th ACM Symposium on Parallelism in Algorithms and Architecture, SPAA (June 2008)
19. Lenzen, C., Wattenhofer, R.: Leveraging Linial's Locality Limit. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 394–407. Springer, Heidelberg (2008)
20. Linial, N.: Locality in Distributed Graph Algorithms. SIAM Journal on Computing 21(1), 193–201 (1992)
21. Luby, M.: A Simple Parallel Algorithm for the Maximal Independent Set Problem. SIAM J. Comput. 15(4), 1036–1055 (1986)
22. Métivier, Y., Robson, J.M., Saheb Djahromi, N.,, Z.: An Optimal Bit Complexity Randomised Distributed MIS Algorithm. In: Kutten, S., Žerovnik, J. (eds.) SIROCCO 2009. LNCS, vol. 5869, pp. 1–15. Springer, Heidelberg (2010)
23. Raz, R., Safra, S.: A Sub-Constant Error-Probability Low-Degree Test, and a Sub-Constant Error-Probability PCP Characterization of NP. In: Proc. of the 29th Annual ACM Symposium on Theory of Computing (STOC), pp. 475–484. ACM, New York (1997)
24. Schneider, J., Wattenhofer, R.: A Log-Star Distributed Maximal Independent Set Algorithm for Growth-Bounded Graphs. In: Proc. of the 27th Annual ACM Symposium on Principles of Distributed Computing, PODC (August 2008)
25. Takamizawa, K., Nishizeki, T., Saito, N.: Linear-Time Computability of Combinatorial Problems on Series-Parallel Graphs. J. ACM 29(3), 623–641 (1982)