Distributed Computing manuscript No.
(will be inserted by the editor)

Distributed Minimum Dominating Set Approximations in Restricted Families of Graphs

Christoph Lenzen · Yvonne-Anne Pignolet · Roger Wattenhofer

Received: date / Accepted: date

Abstract A *dominating set* is a subset of the nodes of a graph such that all nodes are in the set or adjacent to a node in the set. A *minimum dominating set approximation* is a dominating set that is not much larger than a dominating set with the fewest possible number of nodes. This article summarizes the state-of-the-art with respect to finding minimum dominating set approximations in distributed systems, where each node locally executes a protocol on its own, communicating with its neighbors in order to achieve a solution with good global properties.

Moreover, we present a number of recent results for specific families of graphs in detail. A *unit disk graph* is given by an embedding of the nodes in the Euclidean plane, where two nodes are joined by an edge exactly if they are in distance at most one. For this family of graphs, we prove an asymptotically tight lower bound on the trade-off between time complexity and approximation ratio of deterministic algorithms. Next, we consider graphs of small *arboricity*, whose edge sets can be decomposed into a small number of forests. We give two algorithms, a randomized one excelling in its approximation ratio and a uniform deterministic one which is faster and simpler. Finally, we show that in *planar graphs*, which can be drawn in the Euclidean plane without intersecting edges, a constant approximation factor can be ensured within a constant number of communication rounds.

Keywords upper bound · lower bound · unit disk · bounded arboricity · planar

Christoph Lenzen has been partly supported by the Swiss National Science Foundation (SNSF) and the Swiss Society of Friends of the Weizmann Institute.

Christoph Lenzen
Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel
phone: 00972 2 658 5770
fax: 00972 2 658 5727
E-mail: clenzen@cs.huji.ac.il

Yvonne-Anne Pignolet
ABB Corporate Research, Dättwil-Baden, Switzerland
E-mail: yvonne-anne.pignolet@ch.abb.com

Roger Wattenhofer
Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland
E-mail: wattenhofer@tik.ee.ethz.ch

1 Introduction and Notation

For the efficient operation of a network its participants (nodes) need to coordinate their actions. In a decentralized setting, i.e., without a central authority assigning tasks, the nodes have to organize themselves. In many cases partitioning the nodes into clusters with a designated cluster head can help to solve a problem efficiently (e.g., routing messages to distant nodes [46, 47, 49], communication among adjacent nodes (MAC protocols) [11], localization [6], saving energy [19]). Often, these clusterings are based on graph theoretic constructs such as dominating sets. Nodes in a dominating set satisfy the condition that every node has at least one neighbor in the dominating set.

Definition 1.1 (Dominating Sets) Given a graph $G = (V, E)$, a node $v \in V$ *covers* its *inclusive neighborhood* $\mathcal{N}^+(v) := \{w \in V \mid \{v, w\} \in E\}$. A set $A \subseteq V$ covers its *inclusive neighborhood* $\cup_{a \in A} \mathcal{N}^+(a)$. The set $D \subseteq V$ is a *dominating set (DS)* if it covers V. A dominating set of minimal cardinality is a *minimum dominating set (MDS)*.

While it is easy to state the problem of finding a minimum dominating set, solving it is notoriously hard. In

fact, finding a minimum dominating set was one of the first tasks known to be NP-hard [16]. Consequently, one typically is satisfied with an approximate solution.

Definition 1.2 (MDS Approximations) Given $f \in \mathbb{R}^+$, a DS D is an *f-approximation to MDS*, if $|D| \leq f|M|$ for any MDS M. For $f : \mathbb{N} \to \mathbb{R}^+$, an *$f$-approximation algorithm* to the MDS problem returns for any feasible graph of n nodes a DS that is an $f(n)$-approximation. For randomized algorithms this might happen only with at least a certain probability; this probability bound however must not depend on the particular instance.[1]

In general, it is also NP-hard to compute a $(C \log \Delta)$-approximation [41], where $C > 0$ is some constant and Δ denotes the maximum degree of the graph. Indeed, unless NP is a subset of the problems that can be solved within $n^{\mathcal{O}(\log \log n)}$ steps, a $((1 - o(1)) \ln \Delta)$-approximation is intractable in general graphs. This bound is easily matched by the centralized algorithm that in each step picks the node covering the most yet uncovered nodes, until all nodes are covered [22,44].

Note that any hardness result for centralized algorithms also applies to distributed algorithms, i.e., one should not expect to find computationally efficient algorithms with approximation ratio $o(\Delta)$ for general graphs. However, in the distributed setting we have to cope with the additional issues of communication and concurrency. In particular, the simple greedy algorithm mentioned above is inherently sequential and therefore of little use in distributed systems.

It has been shown that a combination of Linear Programming and randomized rounding can asymptotically match the $\Omega(\log \Delta)$ lower bound on the approximation ratio within $\mathcal{O}(\log n)$ rounds [27]. Moreover, the time complexity of any distributed algorithm achieving a polylogarithmic approximation is in $\Omega(\log \Delta)$ and $\Omega(\sqrt{\log n})$ [26]. These lower bounds are based on graphs that have large girth, yet many edges. Although such graphs exist and demonstrate that there is no algorithm solving the problem more efficiently in *any* graph of n nodes or maximum degree Δ, it is highly unlikely that such a graph is ever encountered in a practical setting.

[1] For central or parallel algorithms, one is typically satisfied with a guarantee that holds in expectation. In the distributed setting, this approach has two severe shortcomings. Firstly, one may not want to run multiple instances of the algorithm and take the best result by counting the number of nodes in the obtained MDS: while this will boost the probability of a good approximation ratio, it also incurs a large overhead in the running time if the graph has a large diameter. Secondly, the running times of distributed algorithms are typically small (all presented algorithms run in $\mathcal{O}(\log n)$ time), hence obtaining strong probability bounds at a comparably low running time can be particularly challenging.

Thus, we argue that it is reasonable to examine graph families which occur in realistic settings. Of course, this approach suffers from the drawback that it is not trivial to find appropriate families of graphs—supposing they even exist—offering both sufficiently efficient solutions as well as wide practical applicability. We do not claim to give a satisfying answer to this problem in this article. Instead, we confine ourselves to studying the distributed complexity of the MDS problem in restricted families of graphs.

We will present recent findings on the complexity of the problem in *unit disk* graphs, *planar* graphs, and graphs of small *arboricity* in detail. Before we define these families of graphs, let us mention that each of them is adequate for modelling some distributed systems of interest. Unit disk graphs, which can be embedded in the Euclidean plane such that two nodes share an edge if their distance is below a threshold, are frequently used as a simplified model of communication and interference graphs for wireless networks [25,35]. The study of planar graphs, which can be drawn in the plane such that edges intersect only at their endpoints, is motivated by applications such as VLSI design and vehicle routing. The arboricity of a graph is the maximal density of all its subgraphs. Equivalently, it can be defined by the minimum number of forests into which the edge set can be partitioned. Graphs of bounded arboricity cover a wide range of graph classes, including planar graphs, graphs of bounded genus or treewidth, and, more generally, graphs excluding any fixed minor.

Let us introduce the above families of graphs now, starting with unit disk graphs.

Definition 1.3 (Unit Disk Graphs) A *unit disk graph* $G = (V, E)$ is any graph for which a mapping $\iota : V \to \mathbb{R}^2$ satisfying the property $E = E(\iota) := \left\{\{v, w\} \in \binom{V}{2} \mid \|v - w\|_{\mathbb{R}^2} \leq 1\right\}$ exists.

For this family of graphs (for which the MDS problem remains NP-hard [7]), we generalize the lower bound from [32] in Section 4 to show that no deterministic algorithm can find a constant-factor MDS approximation in $o(\log^* n)$ rounds.

Unit disk graphs feature the property that the number of independent nodes in the r-hop neighborhood of each node is bounded as a function depending on r only.

Definition 1.4 (Independent Sets) Given a graph $G = (V, E)$, a subset of the nodes $I \subseteq V$ is an *independent set (IS)*, if for any $v, w \in I$, $v \neq w$, we have that $\{v, w\} \notin E$. An IS is a *maximal independent set (MIS)*, if no nodes can be added without destroying independence, i.e., for all $v \in V \setminus I$, the set $I \cup \{v\}$ is not independent.

Definition 1.5 (Neighborhoods) For a graph $G = (V, E)$, the (exclusive) *neighborhood* of a node $v \in V$ is

$$\mathcal{N}(v) := \{w \in V \mid \{v, w\} \in E\}$$

and $\delta_v := |\mathcal{N}(v)|$ its degree. Recall that the inclusive neighborhoods of $v \in V$ and $A \subseteq V$, respectively, are

$$\mathcal{N}^+(v) := \{v\} \cup \mathcal{N}_v$$
$$\mathcal{N}^+(A) := \bigcup_{a \in A} \mathcal{N}^+(a),$$

i.e., the sets of nodes covered by v and A.

Moreover, for $r \in \mathbb{N}$,

$$\mathcal{N}^r(v) := \begin{cases} \mathcal{N}^+(v) & \text{if } r = 1 \\ \mathcal{N}^+(\mathcal{N}^{r-1}(v)) & \text{otherwise} \end{cases}$$

denotes the (inclusive) r-neighborhood of v.

Given these definitions, the aforementioned property of unit disk graphs can be formalized as follows.

Definition 1.6 (Bounded Growth) A graph family exhibits *bounded growth*, if there is a polynomial p such that for each instance $G = (V, E)$ it holds that

$$\forall v \in V, r \in \mathbb{N} : \max\{|I| \mid I \subseteq \mathcal{N}^r(v) \text{ is an IS}\} \leq p(r).$$

Note that any MIS is a DS. Thus, in unit disk graphs and any other graph where the number of independent nodes in each (single-hop) neighborhood is bounded by a constant, an MIS is also a constant MDS approximation. Exploiting this property, our lower bound has been matched asymptotically by an algorithm that computes within $\mathcal{O}(\log^* n)$ rounds an MIS in graphs of bounded growth [42].

In contrast, in graphs of small arboricity, which are considered in Section 5, the entire neighborhood of a node with large degree might be independent.

Definition 1.7 (Forest Decompositions and Arboricity) For $f \in \mathbb{N}_0$, an *f-forest decomposition* of a graph $G = (V, E)$ is a partition of the edge set into f rooted forests. The *arboricity* $A(G)$ is the minimum number of forests in a forest decomposition of G.

The graph class of (constantly) bounded arboricity is quite general, as any family of graphs excluding a fixed minor has bounded arboricity [12].

Definition 1.8 (Contractions and Minors) Given a simple graph G, a *minor* of G can be obtained by any sequence of the following operations.

- Deleting an edge.
- Deleting a node.
- *Contracting* an edge $\{v, w\}$, i.e., replacing v and w by a new node u such that $\mathcal{N}_u := \mathcal{N}_v \cup \mathcal{N}_w$.

Note, however, that graphs of bounded arboricity may contain arbitrary minors. In particular, if we take the complete graph $K_{\sqrt{n}}$ of $\sqrt{n}$ nodes and replace its edges by edge-disjoint paths of length two, we obtain a graph of fewer than n nodes and arboricity two that has $K_{\sqrt{n}}$ as minor. Therefore, demanding bounded arboricity is considerably less restrictive than excluding the existence of certain minors.

Both of the algorithms we present for graphs of bounded arboricity improve on the results from [27] that apply to unrestricted graphs. However, although the lower bound from [26] does not hold for graphs of bounded arboricity, these algorithms have logarithmic running times. Hence, in Section 6 we present a different approach that on planar graphs (for which the MDS problem remains NP-hard [16]) achieves an $\mathcal{O}(1)$-approximation in a few number of rounds.

Definition 1.9 (Planarity) A graph G is *planar* if and only if it can be drawn in the two-dimensional plane such that no two nodes are mapped to the same point and edges intersect at their endpoints only.

Equivalently, G is planar if and only if it does neither contain $K_{3,3}$ nor K_5 as a minor [45], where $K_{k,k}$, $k \in \mathbb{N}$, is the complete bipartite graph on k nodes on each side and K_k is the complete graph on k nodes.

Despite the fact that the algorithm we give for planar graphs can be considered impractical due to the use of large messages, we deem this result interesting because it shows that a fast solution exists in a graph family where an $\mathcal{O}(1)$-approximation is not immediately evident. In contrast, in trees the set of inner nodes forms a 3-approximation, for instance, and in graphs of bounded maximum degree Δ taking all nodes yields a $(\Delta + 1)$-approximation.

The remainder of this article is organized as follows. We next motivate and introduce our distributed model of computation. Subsequently, in Section 3, we detail our findings and compare them to related results. Section 4 presents our lower bound on the trade-off between running time and approximation ratio on unit disk graphs. Finally, Sections 5 and 6 present our algorithms for graphs of bounded arboricity and planar graphs, respectively, as well as the proofs of their running time bounds and approximation guarantees.

2 System Model

We make use of a very simple network model. We assume a fault-free distributed system. A simple graph $G = (V, E)$ describes the MDS problem instance as well as the communication infrastructure. In each synchronous round, each node $v \in V$ may send a (different)

message to each of its neighbors $w \in \mathcal{N}(v)$, receives all messages from its neighbors, and may perform arbitrary finite local computations. Initially, node v knows its neighbors $\mathcal{N}(v)$ and possibly a unique identifier of size $\mathcal{O}(\log n)$.

For some algorithms a *port numbering* is sufficient, i.e., the node v has a bijective mapping $p(v,\cdot) : \mathcal{N}(v) \to \{1,\dots,|\mathcal{N}(v)|\}$ at hand. When sending messages, the node specifies which neighbor receives which message by means of the respective port numbers. When receiving, it can tell apart which neighbor sent which message, also in terms of its port numbering. At termination, the node must output whether it is part of the DS or not, and these outputs must define a valid solution of the problem with regard to G.

In the context of distributed graph algorithms, this abstract model can be motivated as follows.

- Asynchronicity can be dealt with by a synchronizer (cf. [2]).
- Recovery from transient faults can be ensured by making an algorithm *self-stabilizing* (cf. [14]). There is a simple transformation from algorithms obeying the given model to self-stabilizing ones [3].
- Changing topology due to joining and leaving nodes, crash failures, etc. also changes the input, i.e., we need to rerun the algorithm on the new topology.
- With respect to lower bounds, we typically restrict neither the number of messages nor their size. For algorithms, these values should of course be small. This is not enforced by the model, but considered as quality measure like the running time.
- Local computation and memory are typically not critical resources, as the most efficient algorithms usually are not highly complicated (within a small number of rounds, only little information is available that can be processed).[2]

It is common practice to focus on the communication complexity and not on the local computational complexity (we refer to [51] for an introduction of the models and complexity measures of distributed computing). In fact, it holds for the algorithms presented in this article that in all but one case the number of local steps per round is linear in the respective node's degree. For the single exception occurring in our algorithm for planar graphs, we discuss how the computations can be polynomially bounded in a note.

Observe that if the algorithm terminates within T communication rounds, recovery from faults or adaption to new topology require *local* operations up to distance at most T from the event only. In particular, if T is sublogarithmic and degrees are bounded, we get a non-trivial bound on the size of the subgraph that may affect the outcome of a node's computations. This underlines the importance of both upper and lower bounds on the time complexity of algorithms in this model. For small time complexities, this might even be the most significant impact of such bounds, since the difference between 5 and 10 communication rounds might be negligible for many applications. However, whether a small fraction or the majority of the nodes has to re-execute the algorithm and change its state in face of e.g. a single node joining the network could change the system performance to a large extent.

[2] Exceptions are algorithms where nodes learn about the entire neighborhood up to a certain distance and then solve a hard problem on this neighborhood. Note, however, that this approach is also in conflict with the goal of few, small messages.

3 Distributed MDS Approximations and our Contributions

In Section 4, we study the problem of approximating an MDS in unit disk graphs. Leveraging Linial's lower bound [32] of $(\log^* n - 1)/2$ on the number of rounds required to compute a 3-coloring or maximal independent set on the ring, we can show that no deterministic algorithm can compute an $f(n)$-approximation in $g(n)$ rounds if $f(n)g(n) \in o(\log^* n)$.

Definition 3.1 (Node Coloring) Given a graph $G = (V,E)$, a *node coloring with $k \in \mathbb{N}$ colors* is a mapping $\mathcal{C} : V \to \{1,\dots,k\}$ such that no two neighbors have the same color, i.e., $\{v,w\} \in E \Rightarrow \mathcal{C}(v) \neq \mathcal{C}(w)$.

Independently, Czygrinow et al. showed by a related, but different approach that a constant approximation is impossible within $o(\log^* n)$ rounds [10].

On the other hand, we already mentioned that unit disk graphs feature *bounded growth*, permitting to compute an MIS in $\mathcal{O}(\log^* n)$ deterministic rounds [42]. Note that graphs of bounded growth (and thus in particular unit disk graphs) also exhibit the weaker property of *bounded independence*, i.e., the neighborhood of each node contains a constantly bounded number of independent nodes only. This implies that an MIS (which is always a DS) is an $\mathcal{O}(1)$-approximation to MDS, as the size of any IS is bounded by the size of an MDS times the respective constant. Therefore, our lower bound is matched asymptotically for the class of graphs of bounded growth. In contrast, the fastest known MIS algorithms on graphs of bounded independence are just the ones for general graphs [1,21,33,36], leaving a gap of $\mathcal{O}(\sqrt{\log n})$ to the lower bound from [26], which in case of MIS applies to line graphs which have bounded independence. Note, however, that for deterministic algorithms that are not based on computing

an MIS, our lower bound is the strongest known, and it remains open whether randomized algorithms that do not compute an MIS may perform better. If the nodes know their position, e.g., by using GPS, a deterministic $(1+\epsilon)$-approximation is feasible in a number of rounds that depends on the choice of $\epsilon > 0$ only [48]. For a communication model where concurrent transmissions interfere and prevent the correct reception of a message [43] provides an algorithm that computes a constant approximation in $\mathcal{O}(\log n)$ rounds with collision detection in graphs of bounded growth and a matching lower bound. Without collision detection, the bounds are $\mathcal{O}(\log^2 n)$ [37] and $\Omega(\log^2 n/\log\log n)$ [23].

In Section 5, we consider graphs of small arboricity. Such graphs can be very different from graphs with small independence. While in the latter case the number of edges may be large (in particular, in the complete graph there are no two independent nodes), graphs of constantly bounded arboricity have $\mathcal{O}(n)$ edges, but potentially large sets of independent neighbors (the star graph has arboricity one). We remark that it is not difficult to see that demanding both bounded growth and arboricity is equivalent to asking for bounded degree. In the family of graphs of maximum degree Δ simply taking all nodes yields a trivial $(\Delta+1)$-approximation to MDS.

We devise two algorithms for graphs of small arboricity A. The first employs a forest decomposition to obtain an $\mathcal{O}(A^2)$-approximation to MDS within time $\mathcal{O}(\log n)$ w.h.p. This algorithm utilizes the fact that not too many nodes may be covered by their children in a given forest decomposition, implying that a good approximation ratio can be maintained if we cover all nodes (that have one) by a parent. Solving this problem approximately can be reduced to computing an arbitrary MIS in some helper graph. Therefore, we get a randomized running time of $\mathcal{O}(\log n)$ w.h.p., whereas the approximation guarantee of $\mathcal{O}(A^2)$ is deterministic.

The second algorithm we propose is based on the property that subgraphs of graphs of bounded arboricity are sparse, i.e., if having n' nodes, they contain at most $A(n'-1)$ edges. For this reason, we can build on the very simple greedy strategy of adding all nodes of locally large degree to the output set simultaneously, until eventually, after $\mathcal{O}(\log\Delta)$ rounds, all nodes are covered. This straightforward approach yields an $\mathcal{O}(A\log\Delta)$-approximation if one makes sure that the number of covered nodes in each repetition is at least the number of selected nodes. The latter can easily be done by requiring that uncovered nodes choose one of their eligible neighbors to enter the set instead of just electing all possible candidates into the set. Playing with the factor up to which joining nodes are required to have largest degree within their two-neighborhood, the algorithm can be modified to an $\mathcal{O}(\alpha A\log_\alpha\Delta)$-approximation within $\mathcal{O}(\log_\alpha\Delta)$ time, for any integer $\alpha \geq 2$. This second algorithm appeals by its simplicity; unlike the first, it is uniform, deterministic, and merely requires port numbers.

Recall that the best algorithm with polylogarithmically sized messages for general graphs has running time $\mathcal{O}(\log^2\Delta)$ and approximation ratio $\mathcal{O}(\log\Delta)$ [27]. Thus, the algorithms given in Section 5 clearly improve on this result for graphs of bounded arboricity. However, although the lower bound from [26] does not hold for such graphs, our algorithms' running times are not below the thresholds of $\Omega(\sqrt{\log n})$ and $\Omega(\log\Delta)$, respectively. In contrast, we show in Section 6 that in planar graphs an $\mathcal{O}(1)$-approximation can be computed in $\mathcal{O}(1)$ rounds.[3] Our algorithm makes use of the facts that planar graphs and their minors are sparse, i.e., contain only $\mathcal{O}(n)$ edges, and that in planar graphs circles separate their interior (with respect to an embedding) from their outside.

A drawback of our algorithm for planar graphs is that it is rendered impractical because it relies on messages that in the worst case encode the whole graph. For the same reason, the technique by Czygrinow et al. [10] to obtain a $(1+\varepsilon)$-approximation in $\mathcal{O}(\log^* n)$ rounds is also of theoretical significance only. If the message size is required to be (poly)logarithmic in n, to the best of our knowledge the currently most efficient distributed algorithms are the ones from Section 5. More generally, the same is true for any graph family excluding fixed minors. Also here distributed $1+\varepsilon$ approximations are known [8,9], however of polylogarithmic running time with large exponent and again using large messages. Excluding particular minors is a rich source of graph families, apart from planar graphs including e.g. graphs of bounded genus or treewidth. Again to the best of our knowledge, currently no distributed algorithms tailored to these families of graphs exist. Moreover, as mentioned before, graphs of bounded (or non-constant, but slowly growing) arboricity extend beyond minor-free graph families. Therefore, the algorithms presented in Section 5 improve on the best known solutions for a wide range of inputs. See Table 3.1 for a comparison of distributed MDS approximations.

A number of works strive for fault-tolerant MDS approximation algorithms. One kind of fault-tolerance considered frequently is *self-stabilization* [14]. A self-stabilizing algorithm recovers from arbitrary transient

[3] This result was claimed previously, in [28] by us and independently in [10] by others. Sadly, our algorithm was wrong and the proof from [10] is incomplete. In this article, we present corrected versions of our algorithm and proof.

Table 3.1 Complexity bounds on distributed MDS approximation in various graph families. NP-hardness results imply impossibility of the stated approximation ratio if $P \neq NP$ and computations are polynomial in n. For upper bounds, the message size is stated to be "trivial" if the whole topology might be collected at individual nodes. The column labeled "det." indicates whether an algorithm is deterministic or whether a lower bound applies to deterministic algorithms only, respectively.

graph family	type	running time	approximation	det.	message size
general [27]	upper bound	$\mathcal{O}(\log n)$	$\mathcal{O}(\log \Delta)$	no	trivial
general [27]	upper bound	$\mathcal{O}(\log^2 \Delta)$	$\mathcal{O}(\log \Delta)$	no	$\mathcal{O}(\log \Delta)$
general [26]	lower bound	$\Omega\left(\min\left\{\sqrt{\log n}, \log \Delta\right\}\right)$	$\operatorname{polylog} n$	no	arbitrary
general [41]	NP-hardness	any (polynomial computations)	$\Omega(\log \Delta)$	no	arbitrary
bounded independence [36]	upper bound	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$	no	$\mathcal{O}(1)$
bounded independence [39]	upper bound	$2^{\mathcal{O}\left(\sqrt{\log n}\right)}$	$\mathcal{O}(1)$	yes	trivial
bounded growth [42]	upper bound	$\mathcal{O}(\log^* n)$	$\mathcal{O}(1)$	yes	$\mathcal{O}(\log n)$
unit disk (Section 4)	lower bound	$g(n) \in o(\log^* n)$	$\notin o\left(\frac{\log^* n}{g(n)}\right)$	yes	arbitrary
unit disk [7]	NP-hardness	any (polynomial computations)	1 (exact)	no	arbitrary
maximum degree Δ (trivial)	upper bound	0	$\Delta + 1$	yes	N/A
forests (trivial)	upper bound	1	3	yes	$\mathcal{O}(1)$
planar (Section 6)	upper bound	6	130	yes	trivial
planar [10]	upper bound	$\mathcal{O}(\log^* n)$	$1 + \varepsilon$	yes	trivial
planar [16]	NP-hardness	any (polynomial computations)	1 (exact)	no	arbitrary
ring [10]	lower bound	$\notin o(\log^* n)$	$5 - \varepsilon$	yes	arbitrary
excluded minor [8,9]	upper bound	$\operatorname{polylog} n$	$1 + \varepsilon$	yes	trivial
arboricity A (Section 5)	upper bound	$\mathcal{O}(A^2)$	$\mathcal{O}(\log n)$	no	$\mathcal{O}(\log n)$
arboricity A (Section 5)	upper bound	$\mathcal{O}(\log_\alpha \Delta)$	$\mathcal{O}(\alpha A \log_\alpha \Delta)$	yes	$\mathcal{O}(\log \log_\alpha \Delta)$

faults by recomputing a correct solution after faults cease. To the best of our knowledge, the most efficient algorithms in this model are obtained from deterministic distributed algorithms by means of a simple transformation [3]. Executing any deterministic distributed algorithm of time complexity T and maximum message size M in an infinite loop, one can construct a self-stabilizing algorithm of message size TM stabilizing in T rounds. By splitting messages, one can for any $F \leq TM$ reduce the message size to $\mathcal{O}(TM/F)$ at the expense of increasing the time the system requires to recover from faults to FT. Note that network dynamics can be treated in a similar manner, i.e., by simply rerunning the algorithm whenever a topology change occurs, and any self-stabilizing algorithm will implicitly do so. Other approaches seek to mask failures, i.e., maintain functional solutions in face of a limited number of crash failures. For instance, for the k-connected m-dominating set problem in UDG's constant approximation algorithms are presented in [50]. We are not aware of research concerning byzantine failures in the context of the MDS problem.

Finally, there are many tasks that are related to the MDS problem. These include for example the independent dominating set problem [5], the connected correlation-dominating set problem [18], the connected coverage problem [15], and computing a family of connected dominating sets minimizing how often nodes are part of a dominating set [20]. A capacitated version of the MDS problem, where each node v in the dominating set may cover at most $\operatorname{cap}(v)$ neighbors, has been considered in [24] for general graphs and bounded independence graphs. A $(1+\epsilon)$-approximation (for $\epsilon > 0$ arbitrary) to the connected MDS problem in UDG's can be computed deterministically in $\mathcal{O}(\epsilon^{-\mathcal{O}(1)} \log^* n)$ rounds [17]. If the nodes know their position, e.g., by using GPS, for any constant $\epsilon > 0$ a $(1+\epsilon)$-approximation to a connected MDS can be obtained in a constant number of rounds by a deterministic algorithm [48].

4 Lower Bound for Unit Disk Graphs

In this section, we will show that in unit disk graphs, no deterministic distributed algorithm can compute an $f(n)$-approximation to MDS in $g(n)$ rounds for any f, g with $f(n)g(n) \in o(\log^* n)$. This bound holds even if message size is unbounded, the nodes have unique identifiers, and the nodes know n. This section is based on [30].

4.1 Definitions and Preliminary Statements

The lower bound proof will reason about the following highly symmetric graphs.

Definition 4.1 (R_n^k) For integers $k, n \in \mathbb{N}$, we define the *k-ring with n nodes* $R_n^k := (V_n, E_n^k)$ by

$$V_n := \{1, \dots, n\}$$
$$E_n := \left\{ \{i,j\} \in \binom{V_n}{2} \;\middle|\; |(i-j) \bmod n| \leq k \right\}.$$

See Figure 4.1 for an illustration. By $R_n := R_n^1$ we denote the "simple" ring. Moreover, we will take numbers modulo n when designating nodes on the ring, e.g. we identify $3n + 5 \equiv 5 \in V_n$.

For any k and n this graph is a UDG (see Definition 1.3).

Lemma 4.2 *R_n^k can be realized as a UDG.*

Proof For $n > 2k + 1$, place all nodes equidistantly on a circle of radius $1/(2\sin(k\pi/n))$. Otherwise, use any circle of radius at most $1/2$. □

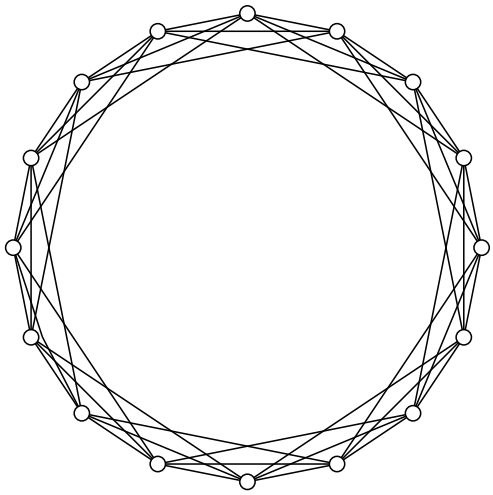

Fig. 4.1 R_{16}^3. Realized as UDG k is controlled by the scaling.

Our bound will be inferred from a classical result by Linial [32], which was later generalized to randomized algorithms by Naor [38].

Theorem 4.3 *There is no deterministic distributed algorithm 3-coloring the ring R_n in fewer than $\frac{1}{2}(\log^* n - 1)$ communication rounds, even if the nodes know n.*

Proof See e.g. [40]. □

We will use the following notion, which captures the amount of symmetry breaking information in the output of an algorithm.

Definition 4.4 ($\sigma(n)$-Alternating Algorithms) Suppose $\mathcal{A}$ is an algorithm operating on R_n which assigns each node $i \in V_n$ a binary output $b(i) \in \{0, 1\}$. We call $\mathcal{A}$ *$\sigma(n)$-alternating*, if the length ℓ of any monochromatic sequence $b(i) = b(i + 1) = \ldots = b(i + \ell)$ in the output of $\mathcal{A}$ is less than $\sigma(n)$.

If a $\sigma(n)$-alternating algorithm is given, one can easily obtain a 3-coloring of the ring R_n in $\mathcal{O}(\sigma(n))$ time.

Algorithm 1: 3-coloring the ring R_n based on σ-alternating inputs.

```
   input : σ-alternating values b(i) for each node i
   output: the color C(i) ∈ {1,2,3} of each node i
 1 for i ∈ V_n in parallel do
 2     find closest node j with b(j) = 1 and b(j+1) = 0
       or b(j−1) = 0
 3     d(i) := |i − j|
 4     C(i) = (d(i) mod 2) + 1 // "coloring with conflicts"
 5     if (C(i−1) = C(i) ∧ l(i) > l(i−1))
 6      ∨ (C(i+1) = C(i) ∧ l(i) > l(i+1)) then
 7         C(i) := C(i) + 2 // resolve conflict
 8     end
 9     // remove temporary color 4
10     if C(i) = 4 then
11         C(i) := min{1,2,3} \ {C(i−1), C(i+1)}
12     end
13 end
```

Lemma 4.5 *Given any $\sigma(n)$-alternating algorithm $\mathcal{A}$ running in $\mathcal{O}(\sigma(n))$ rounds, a 3-coloring of the ring can be computed in $\mathcal{O}(\sigma(n))$ rounds.*

Proof Essentially, nodes simply need to find the closest switch from 0 to 1 (or vice versa) in the output of $\mathcal{A}$. From there, nodes are colored alternatingly, while the third color is used to resolve conflicts where the alternating sequences meet. One can e.g. employ Algorithm 1 to this end.

Clearly, this algorithm can be executed in $\mathcal{O}(\sigma(n))$ rounds, as the definition of $\sigma(n)$ guarantees that the closest switch from 0 to 1 (or vice versa) in the inputs $b(j)$ is within distance $\sigma(n)$. The initially constructed 2-coloring can be interpreted as coloring all nodes with input 1 and some neighbor of input 0 with color 1 and than proceed with alternating colors along monochromatic input values. Since such nodes cannot be adjacent and we get conflicts only where the alternating sequences meet, we have that for each node, at least one of its neighbors has a different color. This is resolved by coloring exactly one of the two nodes of a conflicting pair by a different color; using color 3 for 1-pairs and color 4 for 2-pairs ensures the resulting colors to form a 4-coloring. Finally, color 4 is removed by choosing a free color; since no two neighbors share color 4, this results in a 3-coloring. □

4.2 Proof of the Lower Bound

To establish our lower bound, we construct a $\sigma(n)$-alternating algorithm using an MDS approximation algorithm.

Lemma 4.6 *Assume that a deterministic $f(n)$-approximation algorithm $\mathcal{A}$ for the MDS problem on UDG's*

is given that runs in at most $g(n) \geq 1$ rounds, where $f(n)g(n) \in o(\log^* n)$. Then an $o(\log^* n)$-alternating algorithm $\mathcal{A}'$ requiring $o(\log^* n)$ communication rounds exists.

Proof Assume w.l.o.g. that identifiers on R_n^k are from $\{1,\dots,n\}$. Consider R_n^k and label each node $i \in V_n$ with its input $l(i)$, i.e., its identifier. Set

$$\sigma_k(n) := \max\{f(n), k\}g(n)$$

and define

$$\begin{aligned}\mathcal{L}_n^k := \Big\{ (l_1,\dots,l_\alpha) \in \binom{\{1,\dots,n\}}{\alpha} \\ \mid \alpha \geq \sigma_k(n) + 2kg(n) \wedge (l(1) = l_1 \wedge \dots \wedge l(\alpha) = l_\alpha) \\ \Rightarrow b(kg(n)+1) = \dots = b(\alpha - kg(n)) = 1 \text{ on } R_n^k \Big\},\end{aligned}$$

i.e., the set of sequences of identifiers such that at least $\sigma_k(n)$ consecutive nodes will take the decision $b(v) = 1$ when $\mathcal{A}$ is executed on R_n^k, where the choices of the leading and trailing $kg(n)$ nodes may also depend on labels not in the considered sequence. As the decision of any node $i \in V_n$ depends on identifiers of nodes in $\mathcal{N}^{kg(n)}(i)$ only because it cannot learn from information further away in $g(n)$ rounds of communication on R_n^k, $\mathcal{L}_n^k$ is well-defined.

We distinguish two cases. The first case assumes that values $k_0, n_0 \in \mathbb{N}$ exist, such that for $n \geq n_0$ there is no unique assignment of identifiers $\{1,\dots,n\}$ to the n nodes such that at least $n/2$ nodes participate in non-overlapping sequences $(l(i),\dots,l(i+\alpha)) \in \mathcal{L}_n^{k_0}$ (for *any* possible division of $R_n^{k_0}$ into segments of varying lengths α). Hence, for any fixed n, we may choose a labeling maximizing the number of nodes/labels that participate in such sequences, discard the at most $n/2$ labels that participate in a sequence from $\mathcal{L}_n^{k_0}$, and conclude that the remaining labels, no matter how arranged, cannot form another sequence from $\mathcal{L}_n^{k_0}$. Thus, for $n' := \max\{n_0, 2n\}$, an injective mapping $\lambda_n : \{1,\dots,n\} \to \{1,\dots,n'\}$ exists such that no element of $\mathcal{L}_{n'}^{k_0}$ is completely contained in the image of λ_n. Now we can define $\mathcal{A}'$ (running on R_n) such that each node $i \in \{1,\dots,n\}$ simulates the behaviour it would exhibit when $\mathcal{A}$ was run on $R_{n'}^{k_0}$, with input label $\lambda_n(l(i))$, and return the computed result. Each simulated round of $\mathcal{A}$ will require k_0 communication rounds, thus the running time of $\mathcal{A}'$ is bounded by $k_0 g(n') \in o(\log^* n)$. As $\mathcal{A}$ determines a DS, at most $2k_0$ consecutive nodes will compute $b(i) = 0$, and, by the definitions of $\mathcal{L}_{k_0}^{n'}$ and λ_n, at most $\sigma_{k_0}(n') - 1 \in \mathcal{O}(f(n')g(n')) \subset o(\log^*(n')) = o(\log^* n)$ consecutive nodes take the decision $b(i) = 1$. Hence $\mathcal{A}'$ is $o(\log^* n)$-alternating. This completes the proof for the first case.

In the second case, no pair $k_0, n_0 \in \mathbb{N}$ as assumed in the first case exists. Thus, for any $k \in \mathbb{N}$ some $n \in \mathbb{N}$ exists for which we can construct a labeling of R_n^k with at least $n/2$ identifiers forming (disjoint) sequences in $\mathcal{L}_n^k$. We line up these sequences one after another and label the remaining nodes in a way resulting in a valid labeling of R_n^k. Running $\mathcal{A}$ on such an instance will yield at least

$$\frac{n\sigma_k(n)}{2(\sigma_k(n) + 2kg(n))} \geq \frac{n}{6} \in \Omega(n)$$

nodes choosing $b(i) = 1$.

On the other hand, a minimum dominating set of R_n^k has $\mathcal{O}(n/k)$ nodes. For $k \in \mathbb{N}$, define that n_k is the minimal value of n for which it is possible to construct a labeling of R_n^k with $n/2$ identifiers from sequences in $\mathcal{L}_n^k$. Thus, we have a lower bound of

$$f(n_k) \in \Omega(k) \tag{4.1}$$

on the approximation ratio of $\mathcal{A}$.

As the approximation quality f of $\mathcal{A}$ is sublinear, we conclude that $\lim_{k\to\infty} n_k = \infty$. Therefore, for each n, a minimum value $k(n)$ exists such that $n' := 2n < n_{k(n)}$. Consequently, similarly as in the first case, we can define an injective relabeling function $\lambda_n : \{1,\dots,n\} \to \{1,\dots,n'\}$, such that no element of $\mathcal{L}_{n'}^{k(n)}$ lies completely in the image of λ_n. We define $\mathcal{A}'$ to be the algorithm operating on R_n, but simulating at each node the behaviour of $\mathcal{A}$ on $R_{n'}^{k(n)}$, where we relabel all nodes $i \in \{1,\dots,n\}$ by $\lambda_n(l(i))$. By definition of $k(n)$ we have $n_{k(n)-1} \leq n'$. Together with (4.1) this yields

$$\begin{aligned}k(n) &= (k(n)-1) + 1 \\ &\in O(f(n_{k(n)-1}) + 1) \\ &\subseteq O(f(n')) = O(f(n)),\end{aligned}$$

where the last step exploits that f grows asymptotically sublinearly. Hence we can estimate the running time of $\mathcal{A}'$ by $k(n)g(n') \in O(f(n)g(n))$, using that g grows asymptotically sublinearly as well.

Since the simulated run of $\mathcal{A}$ yields a dominating set, at worst $2k(n) \in O(f(n)) \subseteq O(f(n)g(n))$ consecutive nodes may compute $b(v) = 0$. By the definitions of $\mathcal{L}_n^k$ and λ_n, at most

$$\sigma_{k(n)}(n') - 1 < \max\{f(n'), k(n)\}g(n') \in O(f(n)g(n))$$

consecutive nodes may take the decision $b(i) = 1$. Thus $\mathcal{A}'$ is $o(\log^* n)$-alternating, as claimed. This completes the second case and therefore the proof. □

This result implies the lower bound, as the assumption that a good approximation ratio is possible leads to the contradiction that the ring could be 3-colored quickly.

Theorem 4.7 *A deterministic $f(n)$-approximation to MDS on UDG's running in at most $g(n)$ rounds such that $f(n)g(n) \in o(\log^* n)$ is impossible.*

Proof Assuming the contrary, we may w.l.o.g. assume that $g(n) \geq 1$ for all $n \in \mathbb{N}$. Thus, we can combine Lemma 4.6 and Lemma 4.5 to construct an algorithm that 3-colors the ring in $o(\log^* n)$ rounds, contradicting Theorem 4.3. □

We remark that it is an open question whether a randomized algorithm can break this barrier for MDS approximations. For the MaxIS problem it is known that in planar graphs (and thus in particular on the ring), for any fixed constant $\varepsilon > 0$ a constant-time randomized algorithm can guarantee a $(1+\varepsilon)$-approximation w.h.p. [10].

5 Algorithms for Graphs of Bounded Arboricity

We present two MDS approximation algorithms published in [31] that are devised for graphs of small arboricity A. The first algorithm employs a forest decomposition, achieving a guaranteed approximation ratio of $\mathcal{O}(A^2)$ within $\mathcal{O}(\log n)$ rounds w.h.p. The second computes an $\mathcal{O}(A \log \Delta)$ approximation deterministically in $\mathcal{O}(\log \Delta)$ rounds. Both algorithms require small messages only.

5.1 Constant-Factor Approximation

In this section, we give an algorithm that computes a dominating set at most a factor of $\mathcal{O}(A^2)$ larger than optimum. After presenting the algorithm and its key ideas, we proceed with the formal proof of its properties.

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 covered by a parent, the other contains the remaining nodes, which thus are themselves in M or have a child in M. Since each dominating set node can cover at most f parents, the latter set contains in total 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 centralized 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 optimal 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 centralized 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 with respect to the approximation ratio to select them all. Denoting for $v \in V$ by $P(v)$ the set of parents of v in a given forest decomposition of G, this approach is summarized in Algorithm 2.

Algorithm 2: Parent Dominating Set

input : f-forest decomposition of G
output: dominating set D

1 $H := \left(V, \left\{\{v,w\} \in \binom{V}{2} \mid P(v) \cap P(w) \neq \emptyset\right\}\right)$
2 Compute a maximal independent set I on H
3 $D := \bigcup_{v \in I} P(v)$
4 $D := D \cup (V \setminus \mathcal{N}^+(D))$

Analysis

We need to bound the number of nodes that join the dominating set because they are elected by children.

Lemma 5.1 *In Line 3 of Algorithm 2, 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 3 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 $V_P \subseteq V$ that have at least one parent in M, in particular the nodes in $I \cap V_P$ that are also in the computed independent set. By the definition of H 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 $f|M|$ nodes join D because they are elected into the set 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

$$\begin{aligned}|D| &\leq f|I \cap V_C| + f|I \cap V_P| \\ &\leq f(f+1)|M| + f|M| \\ &= f(f+2)|M|,\end{aligned}$$

concluding the proof. □

The approximation ratio of the algorithm now follows easily.

Theorem 5.2 *Algorithm 2 outputs a dominating set D containing at most $(f^2+3f+1)|M|$ nodes, where M is an optimum solution.*

Proof By Lemma 5.1, at most $f(f+2)|M|$ nodes enter D in Line 3 of the algorithm. Since I is an MIS in H, all nodes that have a parent are adjacent to at least one node in D after Line 3. Hence, the nodes selected in Line 4 must be covered by a child in M or themselves be in M. As no node has more than f parents, thus in Line 4 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. □

Employing known distributed algorithms for computing an $\mathcal{O}(G(A))$-forest decomposition and an MIS, we can construct an MDS approximation algorithm.

Corollary 5.3 *In any graph G, a factor $\mathcal{O}(A(G)^2)$-approximation to an MDS can be computed distributedly in $\mathcal{O}(\log n)$ rounds w.h.p., provided that nodes know a polynomial upper bound on n or a linear upper bound on $A(G)$. In particular, on graphs of bounded arboricity a constant-factor approximation can be obtained in $\mathcal{O}(\log n)$ rounds w.h.p. This can be accomplished with messages of size $\mathcal{O}(\log n)$.*

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

For the forest decomposition, we employ the algorithm from [4], yielding a decomposition into $\mathcal{O}(A(G))$ forests in $\mathcal{O}(\log n)$ rounds; this algorithm is the one that requires the bound on n or $A(G)$, respectively, that is asked for in the preliminaries of the corollary. An MIS can be computed in $\mathcal{O}(\log n)$ rounds w.h.p. by well-known algorithms [1,21,33], or a more recent similar technique [36]. In total the algorithm requires $\mathcal{O}(\log n)$ rounds w.h.p. and according to Theorem 5.2 the approximation guarantee is $\mathcal{O}(A(G)^2)$.

Regarding the message size, the algorithm to compute a forest decomposition requires messages containing $\mathcal{O}(\log n)$ bits. Thus, we need to check that we do not require large messages because we compute an MIS on H. Formulated abstractly, the algorithm from [36] breaks symmetry by making each node still eligible for entering the IS choosing a random value in each round and permitting it to join the IS 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 differ w.h.p. 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 the smallest values only to their children.

If (an upper bound on) n is not known, one can start with constantly many bits and (locally) double the number of used bits in each round where two nodes pick the same value. Since for any edge in H there is some node that sees the random values of both its endpoints (the respective common parent), we can always inform a node if there is a conflict. In this case, the nodes (i) pick additional random bits for the round in question and (ii) increase the number of random bits for future rounds by a factor of two. There will be at most $\log\log n + \mathcal{O}(1)$ many such "failed" rounds for each node until in each round the chosen values will differ from all others w.h.p. Since nodes can complete any round for which they broke symmetry to all their neighbors and we can always locally propagate the current *maximal* number of random bits employed in each round, we can ensure that in total the running time of the algorithm increases by asymptotically negligible $\mathcal{O}(\log\log n)$ rounds w.h.p. due to the lack of knowledge on n. □

5.2 Linear-Time Central Algorithm

Algorithm 2 might also be of interest in a centralized setting. Employing well-known techniques, a central algorithm can compute a suitable forest decomposition with linear complexity, which gives rise to an efficient implementation of Algorithm 2.

Lemma 5.4 *A $2A(G)$-forest decomposition of G can be computed in $\mathcal{O}(|E|+n) \subseteq \mathcal{O}(nA(G))$ 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 $\mathcal{O}(|E| + n) \subseteq \mathcal{O}(nA(G))$, as each edge is accessed a constant number of times and each node is accessed no more than once without accessing an edge as well. Since any graph of arboricity $A(G)$ has average degree smaller than $A(G)$, the smallest degree of any subgraph of G is less than $2A(G)$. Hence we obtain a forest decomposition into less than $2A(G)$ forests. □

Hence, for any graph of arboricity $A(G) \in \mathcal{O}(1)$ we can compute an $\mathcal{O}(1)$-approximation to the MDS problem at linear complexity.

Corollary 5.5 *A deterministic, centralized $\mathcal{O}(A(G)^2)$-approximation algorithm for the MDS problem exists that runs for $\mathcal{O}(|E| + n) \subseteq \mathcal{O}(A(G)n)$ steps.*

Proof By Lemma 5.4 we can compute a forest decomposition within the stated complexity bounds. In a central setting, Algorithm 2 can easily be implemented using $\mathcal{O}(|E| + n)$ steps. The approximation guarantee follows from Theorem 5.2. □

5.3 Uniform Deterministic Algorithm

Algorithm 2 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 [4]. 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 an 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 generate unique identifiers of size $\mathcal{O}(\log n)$ in constant time w.h.p., 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 initially given orientation of the edges.

In this section, we show that a uniform, deterministic algorithm exists that requires a port numbering only, yet achieves a running time of $\mathcal{O}(\log \Delta)$ and a logarithmic 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.

Algorithm

The basic idea of Algorithm Greedy-by-Degree (Algorithm 3) is that it is always feasible to choose nodes of high *residual degree* simultaneously, i.e., all the nodes that cover up to a constant factor as many nodes as the one covering the most uncovered nodes.

Definition 5.6 (Residual Degree) Given a set $D \subseteq V$, the residual degree of node $v \in V$ with respect to D is $\bar{\delta}_v := |\mathcal{N}^+(v) \setminus \mathcal{N}^+(D)|$.

This permits to obtain strong approximation guarantees without the structural information provided by knowing $A(G)$ 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 two. If not, the respective 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 two 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 cover the same small set of $A(G) - 1$ nodes. For this reason, it is mandatory to ensure that not more nodes 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 is feasible according to the criterion of (locally) large residual degree explained above. This scheme is described in Algorithm 3.

Note that nodes 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 $\bar{\delta}_v = 0$, i.e., their entire inclusive neighborhood is covered by D.

Analysis

In the sequel, when we talk of a *phase* of Algorithm 3, we refer to a complete execution of the while loop. We start by verifying that one iteration of the loop can be executed within six rounds by a local algorithm that relies on port numbers only.

Algorithm 3: Greedy-by-Degree.

```
   output: dominating set D
1  D := ∅
2  while V ≠ N+(D) do
3  |  C := ∅
4  |  for v ∈ V in parallel do
5  |  |  δ̄_v := |N+(v) \ N+(D)|
6  |  |  δ̄_v^(1) := max_{w∈N+(v)} {δ̄_w}
7  |  |  δ̄_v^(2) := max_{w∈N+(v)} {δ̄_w^(1)}
8  |  |  if ⌈log δ̄_v⌉ ≥ ⌈log δ̄_v^(2)⌉ then
9  |  |  |  C := C ∪ {v}
10 |  |  end
11 |  |  if v ∈ N+(C) \ N+(D) then
12 |  |  |  choose any w ∈ C ∩ N+(v)
13 |  |  |  D := D ∪ {w}
14 |  |  end
15 |  end
16 end
```

Lemma 5.7 *Each phase (i.e., iteration of the while-loop) of Algorithm 3 can be executed within six rounds. It is sufficient if nodes have access to a port numbering.*

Proof The proof goes by induction on the phases. The induction hypothesis is that at the beginning of each phase, node v is aware of $\mathcal{N}^+(v) \cap \mathcal{N}^+(D)$. Clearly, this holds at the beginning of the first phase. Assuming that this hypothesis is true for some phase, v can compute $\bar{\delta}_v$ and send it to all neighbors within one round. In the next round, v computes $\bar{\delta}_v^{(1)}$ and sends it to all neighbors. Hence, at the beginning of round three, v can compute $\bar{\delta}_v^{(2)}$ and determine whether it must join C. This will be communicated in the same round, implying that in the fourth round, v can, if required, pick a node from $C \cap \mathcal{N}^+(v)$ and inform the chosen node. Finally, in round five and six, nodes transmit whether they joined D and whether they became covered by D, respectively. This completes the induction and shows the claimed bound on the running time of each phase. Observe that all the above steps can be executed based on a port numbering, implying the second statement of the lemma. □

Next, we prove that not too many nodes with small residual degrees enter D.

Lemma 5.8 *Denote by M an MDS of G. During the execution of Algorithm 3, in total at most $16A(G)|M|$ nodes v join D in Line 13 of the algorithm after computing $\bar{\delta}_v \leq 8A(G)$ in Line 5 of the same phase.*

Proof Consider the set S consisting of all nodes $v \in V$ that become covered in some phase by some node $w \in \mathcal{N}_v^+$ that computes $\bar{\delta}_w \leq 8A(G)$ and joins D. As according to Line 8 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 $16A(G)$ nodes from S. Hence, $|S| \leq 16A(G)|M|$. The rule that a node needs to be elected by a node it covers in order to enter D implies that each node joining D while having residual degree at most $8A(G)$ is chosen by some node from S in Line 12. As each node executing Line 12 in some phase gets covered in this phase, the statement of the lemma directly follows from the derived bound on $|S|$. □

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

Lemma 5.9 *If M is an MDS, in each phase of Algorithm 3 at most $16A(G)|M|$ nodes $v \in V$ that compute $\bar{\delta}_v > 8A(G)$ in Line 5 join D in Line 13.*

Proof Fix some phase of the algorithm and denote by D' the set of nodes $v \in V$ joining D in Line 13 of this phase after computing $\bar{\delta}_v > 8A(G)$. Define V' to be the set of nodes that had not been covered at the beginning of the phase. Define for $i \in \{0, \dots, \lceil \log n \rceil\}$ that

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

Note that $\bigcup_{i=\lceil \log 8A(G) \rceil}^{\lceil \log n \rceil} D_i = D'$.

Consider any $j \in \{\lceil \log 8A(G) \rceil, \dots, \lceil \log n \rceil\}$. Nodes in V_j may be covered by nodes from M_i for $i \leq j$ only. 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 $\bigcup_{i \in \{j, \dots, \lceil \log n \rceil\}} V_i$, as by definition they have no neighbors in V_i for $i < j$. On the other hand, Lines 5 to 8 of the algorithm impose that these nodes must not have any neighbors of residual degree larger than $2^{\lceil \log \bar{\delta}_v \rceil} = 2^j$, 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} neighbors in V_j. This observation implies that the subgraph induced by $D_j \cup V_j$ has at least $2^{j-2}|D_j| \geq 2A(G)|D_j|$ edges. On the other hand, by definition of the arboricity, this subgraph has fewer than $A(G)(|D_j| + |V_j|)$ edges. It follows that

$$\begin{aligned} |D_j| &< \frac{A(G)|V_j|}{2^{j-2} - A(G)} \\ &\leq 2^{3-j} A(G) |V_j| \\ &\leq 2^{3-j} A(G) \sum_{i=0}^{j} 2^i |M_i|. \end{aligned}$$

We conclude that

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

as claimed. □

We now can bound the approximation quality of the algorithm.

Theorem 5.10 *Algorithm 3 terminates after at most* $6\lceil\log(\Delta+1)\rceil$ *rounds and outputs a dominating set that is at most a factor* $16A(G)\log\Delta$ *larger than optimum. The 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 maximal residual degree after the i^{th} phase, i.e., $\Delta(0) = \Delta + 1$ (as a node also covers itself). According to Lemma 5.7, each phase of Algorithm 3 takes six rounds. Because all nodes v computing a $\bar{\delta}_v$ satisfying $\lceil\log\bar{\delta}_v\rceil = \lceil\log\Delta(i)\rceil$ 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. Note also that in the final phases $\lceil\log(\Delta+1)\rceil-2$, $\lceil\log(\Delta+1)\rceil-1$, and $\lceil\log(\Delta+1)\rceil$ we have that $\Delta(i) \leq 8$.

Having established the bound on the running time of the algorithm, its approximation ratio follows by applying Lemma 5.9 to the first $\lceil\log(\Delta+1)\rceil - 3$ phases and Lemma 5.8 once, and finally observing that $\log\Delta \geq \lceil\log(\Delta+1)\rceil-2$. The bound on the message size follows from the observation that in each phase nodes need to exchange residual degrees rounded to powers of two and a constant number of binary values only. □

Like it is possible for the MDS approximation algorithm for general graphs from [27], we can sacrifice accuracy in order to speed up the computation.

Corollary 5.11 *For any integer* $\alpha \geq 2$*, Algorithm 3 can be modified so that it has running time* $\mathcal{O}(\log_\alpha\Delta)$ *and approximation ratio* $\mathcal{O}(A(G)\alpha\log_\alpha\Delta)$*. The size of messages becomes* $\mathcal{O}(\log\log_\alpha\Delta)$ *with this modification.*

Proof We simply change the base of the logarithms in Line 8 of the algorithm, i.e., instead of rounding residual degrees to integer powers of two, we round to integer powers of α. Naturally, this affects the approximation guarantees linearly. In the proof of Lemma 5.9, we just replace the respective powers of two by powers of 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 5.10. Similarly, the bound on the message size becomes $\mathcal{O}(\log\log_\alpha\Delta)$. □

If it was not for the computation of an MIS, we could speed up Algorithm 2 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 an MIS in time $o(\log n)$ is known.

Finally, we would like to mention that 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 $\bar{\delta}_v > 8A(G)$ is satisfied. To complete the dominating set, uncovered nodes may join D independently of $\bar{\delta}_v$ once their neighborhood contains no more nodes of residual degree larger than $8A(G)$. It is not hard to see that with this modification, essentially the same analysis as for Algorithm 3 applies, both with regard to time complexity and approximation ratio.

6 Constant-Time Constant Approximation in Planar Graphs

In this section, which is based on [29], we introduce an algorithm computing a constant approximation of a minimum dominating set in planar graphs in constant time.[4] Assuming maximum degree Δ and identifiers of size $\mathcal{O}(\log n)$, the algorithm makes use of messages of size $\mathcal{O}(\Delta\log n)$. As planar graphs exhibit unbounded degree, the algorithm is thus not suitable for practice. Moreover, the constant in the approximation ratio is 130, i.e., there is a large gap to the strongest known lower bound of $5-\varepsilon$ (for any constant $\varepsilon > 0$) [10]. Nevertheless, we demonstrate that in planar graphs in principle it is feasible to obtain a constant MDS approximation in a constant number of distributed rounds.

[4] Note that the original paper [28] contained an error and the stated algorithm does not compute a constant MDS approximation. Moreover, the proof of the algorithm from [10] is incomplete.

Algorithm 4: MDS Approximation in Planar Graphs

```
   output: DS D of G
 1 D := ∅
 2 for v ∈ V in parallel do
 3 |   if ∄A ⊆ N^2(v) \ {v} such that N(v) ⊆ N^+(A)
   |   and |A| ≤ 6 then
 4 |   |   D := D ∪ {v}
 5 |   end
 6 end
 7 for v ∈ V in parallel do
 8 |   δ̄_v := |N^+(v) \ N^+(D)|
 9 |   if v ∈ V \ N^+(D) then
10 |   |   Δ_v := max_{w∈N^+(D)} {δ̄_w}
11 |   |   choose any d(v) ∈ {w ∈ N^+(v) | δ̄_w = Δ_v}
12 |   |   D := D ∪ {d(v)}
13 |   end
14 end
```

6.1 Algorithm

The key idea of the algorithm is to exploit planarity in two ways. On the one hand, planar graphs have arboricity three, i.e., the number of edges of any subgraph is linear in its number of nodes. What is more, as planarity is preserved under taking minors, so does any minor of the graph. On the other hand, in a planar graph circles are barriers separating parts of the graph from others; any node enclosed in a circle cannot cover nodes on the outside. This is a very strong structural property enforcing that dominating sets are either large or exhibit a simple structure. It will become clear in the analysis how these properties are utilized by the algorithm.

The algorithm consists of two main steps. In the first step all nodes check whether their neighborhood can be covered by six or less other nodes. After learning about their two-hop neighborhood in two rounds, nodes can determine if this is the case locally with a polynomial-time algorithm.[5] Otherwise, they join the (future) dominating set. In the second step, any node that is not yet covered elects a neighbor of maximal residual degree (i.e., one that covers the most uncovered nodes, see Definition 5.6) into the set. Algorithm 4 summarizes this scheme.

6.2 Analysis

The algorithm can be executed in six rounds and, due to the second step, computes a dominating set.

Lemma 6.1 *Algorithm 4 can be executed in six rounds and computes a dominating set.*

[5] Trivially, one can try all combinations of six nodes, but planarity permits more efficient solutions.

Proof All nodes can learn within two rounds about the identifiers of their neighbors' neighbors and decide whether they need to join D in the first step of the algorithm or not. In the third round this is communicated. Rounds 4 and 5 are required for each node $v \in V$ to learn about $\mathcal{N}^+(v) \cap \mathcal{N}^+(D)$ and send $\bar{\delta}_v$ to its neighbors. In the final round, uncovered nodes elect a neighbor into D and communicate their choice. This step also guarantees that D is a dominating set. □

Therefore, our task is to bound the number of nodes selected in each step in terms of the size of a minimum dominating set M of the planar graph G. For the purpose of our analysis, we fix some MDS M of G. By D_1 and D_2 we denote the sets of nodes that enter D in the first and second step of the algorithm, respectively. Moreover, we denote neighborhoods in a graph $H \neq G$ by $\mathcal{N}_H(v), \mathcal{N}_H^+(A)$, etc. We will need the following basic statements about planar graphs.

Lemma 6.2 *A minor of a planar graph is planar. A planar graph of $n \geq 3$ nodes has at most $3n - 6$ edges (e.g. [13, 34]).*

We begin by bounding the number of nodes in $D_1 \setminus M$ after the first step.

Lemma 6.3 $|D_1 \setminus M| < 3|M|$.

Proof We construct the subgraph $H = (V_H, E_H)$ of G as follows (see Figure 6.1).

- Set $V_H := \mathcal{N}^+(D_1 \setminus M) \cup M$ and $E_H := \emptyset$.
- Add all edges with at least one endpoint in $D_1 \setminus M$ to E_H.
- Add a minimal subset of edges from E to E_H such that $V_H = \mathcal{N}_H^+(M)$, i.e., M is a DS in H.

Thus, each node $v \in V_H \setminus (D_1 \cup M)$ has exactly one neighbor $m \in M$, as we added a minimal number of edges for M to cover V_H. For all such nodes v, we contract the edge $\{v, m\}$, where we identify the resulting node with m. In other words, the star subgraph of H induced by $\mathcal{N}_H^+(m) \setminus D_1$ is collapsed into m. By Lemma 6.2, the resulting minor $\bar{H} = (V_{\bar{H}}, E_{\bar{H}})$ of G satisfies that $|E_{\bar{H}}| < 3|V_{\bar{H}}|$. Due to the same lemma, the subgraph of $\bar{H}$ induced by $D_1 \setminus M$ has fewer than $3|D_1 \setminus M|$ edges. As the neighborhood in G (and thus also in H) of a node from $D_1 \setminus M \subset V_{\bar{H}}$ cannot be covered by fewer than seven nodes, the performed edge contractions did not reduce the degree of such a node below seven.

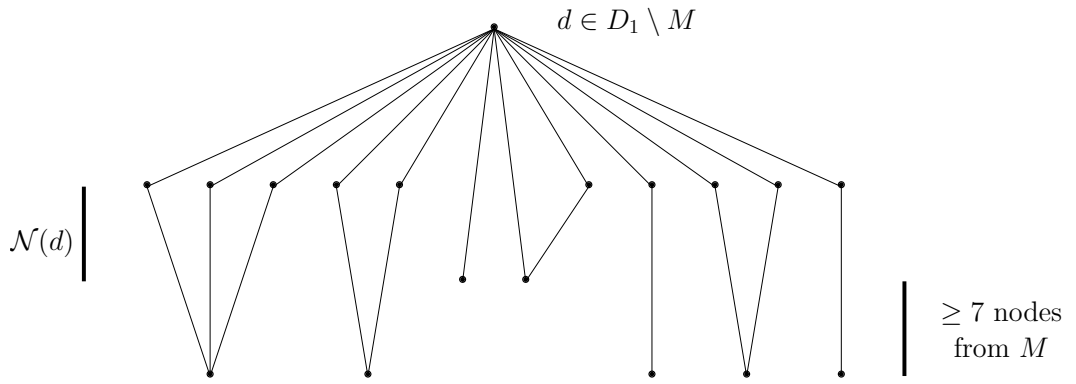

Fig. 6.1 Part of the subgraph constructed in Lemma 6.3.

Altogether, we get that

$$\begin{aligned}
&7|D_1 \setminus M| - 3|D_1 \setminus M| \\
&< \sum_{d \in D_1 \setminus M} \delta_{\bar{H}}(d) - |\{\{d, d'\} \in E_{\bar{H}} \,|\, d, d' \in D_1 \setminus M\}| \\
&\leq |E_{\bar{H}}| \\
&< 3|V_{\bar{H}}| \\
&\leq 3(|D_1 \setminus M| + |M|),
\end{aligned}$$

which can be rearranged to yield the claimed bound. □

To bound the number of nodes $|D_2|$ chosen in the second step of the algorithm, more effort is required. Let us first sketch the key ideas behind our reasoning. The potentially "problematic" nodes with respect to the approximation ratio are nodes $d \in D_2 \setminus M$ that are not chosen by nodes from M. Such nodes satisfy two important properties: On the one hand, they had a large residual degree before they joined D, and on the other hand they have been chosen by some node in $V \setminus (\mathcal{N}^+(D_1) \cup M)$. For each node in $D_2 \setminus M$ chosen by a node not in M, we pick exactly one such node that chose it, resulting in the set A. Each node in this set must be covered twice, (i) by a node in M and (ii) by another node from a set C comprising at most $6|M|$ nodes. This is true since each $m \in M$ whose neighborhood can not be covered by 6 different nodes has been selected in the first step of the algorithm, i.e., $m \in D_1$.

Assuming that D_2 is indeed large compared to M, there is now a large number of two-hop paths between nodes in $D_2 \setminus M$ and nodes in M via nodes in A. Contracting these paths results in a minor of G of $|M \cup C| \leq 7|M|$ nodes that has $\mathcal{O}(|M|)$ edges. Hence, if it holds that, for all pairs of nodes $m \in M$ and $v \in C \setminus \{m\}$, there are only few shared neighbors in A, we can bound $|D_2| \in \mathcal{O}(|M|)$, which is the purpose of Lemma 6.5.

On the other hand, if we have a pair $(m, v) \in M \times C$, $m \neq v$, that shares a lot of neighbors in A, planarity entails that some of these neighbors are "shielded" by circles formed by m, v, and the "outer" nodes from $A \cap \mathcal{N}(m) \cap \mathcal{N}(v)$ according to some fixed embedding of the graph. In case there is no further node from M enclosed, this implies that all enclosed nodes are connected to m (or possibly v if $v \in M$ as well). Roughly speaking, leveraging the fact that the elected nodes must be also inside these circles, they must have larger degrees than both m and c to be chosen by their corresponding node in A. However, if there are no further nodes from M enclosed, additional nodes will also increase the degree of m (or alternatively v, if $v \in M$), as they must be covered. The number of required egdes thus becomes quadratic in the number of enclosed nodes from D_2, resulting in a constant bound on the number of enclosed nodes from D_2. The actual reasoning is more involved, as it is based on a recursive counting argument using the above arguments, and will be formalized in Lemma 6.6.

We consider the following subgraph of G, which is tailored specifically to arguing as outlined above.

Definition 6.4 We define $H = (V_H, E_H)$ as the subgraph of G obtained from the following construction.

- Set $V_H := \emptyset$ and $E_H := \emptyset$.
- For each node $d \in D_2$ for which this is possible, add one node $v \in V \setminus M$ to V_H such that $d = d(v)$ in Line 11 of the algorithm.
- Add $M \setminus D_1$ to V_H and a minimal number of edges to E_H such that $\mathcal{N}_H^+(M \setminus D_1) = V_H$, i.e., $M \setminus D_1$ covers the nodes added to H so far (this is possible as only nodes from $V \setminus \mathcal{N}^+(D_1)$ elect nodes into D_2).
- For each $m \in M \setminus D_1$, add a minimal number of nodes and edges to H such that a set $C_m \subseteq V_H \setminus \{m\}$ of minimal size satisfying $\mathcal{N}_H(m) \subseteq \mathcal{N}_H^+(C_m)$ exists, i.e., C_m covers m's neighbors in H. We define that $C := \cup_{m \in M \setminus D_1} C_m$.
- Remove all $v \in V_H \setminus (C \cup M)$ for which $d(v) \in M \cup C$.
- For each $m \in M \setminus D_1$, remove all edges to C_m.

See Figure 6.2 for an illustration.

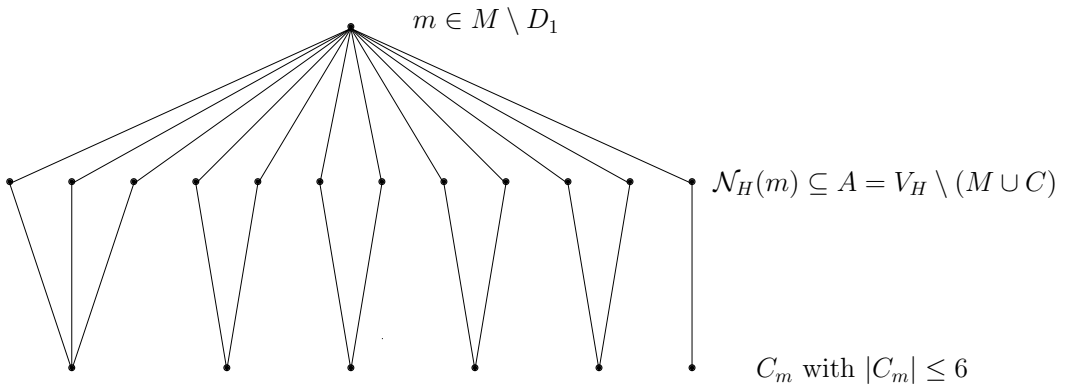

Fig. 6.2 Part of the subgraph H from Definition 6.4.

In order to derive our bound on $|D_2|$, we consider a special case first.

Lemma 6.5 *Assume that for each node $m \in M \setminus D_1$ it holds that*

(i) no node $m' \in M \cap C_m$ covers more than seven nodes in $\mathcal{N}_H(m)$ and
(ii) no node $v \in C_m \setminus M$ covers more than four nodes in $\mathcal{N}_H(m)$.

Then it holds that $|D_2| < 98|M|$.

Proof Each node in $V_H \setminus M$ that elects some node into D_2 has exactly two neighbors, one in M and a different one in C. Denote by $A_1 \subseteq V_H \setminus (M \cup C)$ the nodes in V_H that elect others into D_2 and have two neighbors in M, i.e., when we added C to V_H, they became covered by a node in $M \cap C$. Analogously, denote by $A_2 \subseteq V_H \setminus (M \cup C)$ the set of electing nodes for which the neighbor in C is not in M. Observe that $A := A_1 \cup A_2 = V_H \setminus (M \cup C)$ and $A_1 \cap A_2 = \emptyset$. Moreover, we claim that $|A| \geq |D_2| - 14|M|$. To see this, recall that in the first step of the construction of H, we choose for each element of D_2 that is not elected by elements of M only one voting node v, i.e., at least $|D_2| - |M|$ nodes in total. In the second last step of the construction, we remove v if $d(v) \in \{m\} \cup C_m$ for some $m \in M \setminus D_1$. As $m \in M \setminus D_1$, its neighborhood can be covered by six or less nodes from $V \setminus \{m\}$. Therefore $|C_m| \leq 6$ for all $m \in M \setminus D_1$ and we remove in total at most $7|M|$ nodes in the second last step. Finally, in the last step we cut off at most $|C| \leq 6|M|$ voting nodes from their dominators in $M \setminus D_1$. The definition of A explicitly excludes these nodes, hence $|A| \geq |D_2| - 14|M|$.

Recall that for each node $a \in A$, there is a unique node $m \in M \setminus D_1$ by which it became covered in the third step of the construction of H. For each such a, we contract the edge to m, identifying the resulting node with m. Denote the resulting minor of G by $\bar{H} = (V_{\bar{H}}, E_{\bar{H}})$. No pair of nodes $m, m' \in M \setminus D_1$ satisfying that $m \in C_{m'}$ and $m' \in C_m$ shares more than seven neighbors in A_1. Thus, there are at least $|A_1|/7$ different such pairs $m, m' \in M \setminus D_1$. For each of these pairs, we have two nodes less in $V_{\bar{H}}$ than the upper bound of $|V_{\bar{H}}| \leq |M| + |C| \leq 7|M|$. By Lemma 6.2, $\bar{H}$ thus has fewer than

$$\begin{aligned} 3|V_{\bar{H}}| &\leq 3|M \cup C| \\ &\leq 3|M| + 3\left(6|M| - \frac{2|A_1|}{7}\right) \\ &= 21|M| - \frac{6|A_1|}{7} \end{aligned}$$

edges.

On the other hand,

$$|E_{\bar{H}}| \geq \frac{|A_1|}{7} + \frac{|A_2|}{4},$$

as by assumption each pair of nodes from M may share at most seven neighbors in A_1 and pairs of nodes $m \in M \setminus D_1$, $v \in C_m \setminus M$ share at most four neighbors. We conclude that

$$|A_2| < 84|M| - 4|A_1|$$

and therefore

$$|D_2| \leq |A_1| + |A_2| + 14|M| < 98|M| - 3|A_1| \leq 98|M|.\square$$

In order to complete our analysis, we need to cope with the case that a node $m \in M \setminus D_1$ and an element of C_m share many neighbors. In a planar graph, this results in a considerable number of nested circles which separate their interior from their outside space. This necessitates that nodes from the optimal solution M are enclosed that we may use to compensate for the increased number of nodes in A in comparison to the special case from Lemma 6.5.

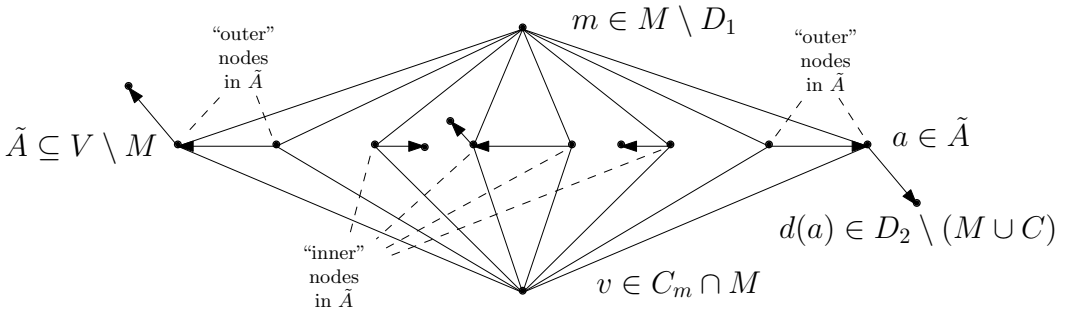

Fig. 6.3 Example of a subgraph considered in the first case of the proof of Lemma 6.6. While the choices $d(a)$ of the two leftmost and rightmost nodes $a \in \tilde{A}$ may have large degrees because of nodes outside the outer circle, the nodes elected by the four inner nodes must have many neighbors that are not covered by D_1 on or inside the outer circle.

Lemma 6.6 *Suppose the subgraph from Definition 6.4 violates condition (i) or (ii) from Lemma 6.5. Fix a planar embedding of G and consider either*

(i) nodes $m \in M \setminus D_1$ and $v \in M \cap C_m$ with $|\mathcal{N}_H(m) \cap \mathcal{N}_H(v)| \geq 8$ or
(ii) nodes $m \in M \setminus D_1$ and $v \in C_m \setminus M$ with $|\mathcal{N}_H(m) \cap \mathcal{N}_H(v)| \geq 5$.

Then the outermost circle formed by m, v, and two of their common neighbors in H must enclose some node $m' \in M$ (with respect to the embedding).

Proof Set $\tilde{A} := \mathcal{N}_H(m) \cap \mathcal{N}_H(v)$. Consider case (i) first and assume for contradiction that there is no node from M enclosed in the outermost circle. W.l.o.g., we may assume that $|\tilde{A}| = 8$ (otherwise we simply ignore some nodes from $\tilde{A}$). There are four nodes from $\tilde{A}$ that are enclosed by two nested circles consisting of v, m, and the four nodes that are the outer nodes from $\tilde{A}$ according to the embedding (see Figure 6.3). Recall that by the second last step of the construction of H nodes $a \in \tilde{A}$ satisfy that $d(a) \notin \{m, v\} \subseteq M$. Therefore, these enclosed nodes elected (distinct) nodes into D_2 that are enclosed by the outermost circle. As the electing nodes $a \in \tilde{A}$ are connected to m and v, by Line 11 of the Algorithm the nodes $d(a)$ they elected must have at least residual degree $\bar{\delta}_{d(a)} \geq \max\{\bar{\delta}_v, \bar{\delta}_m\}$. In other words, they cover at least as many nodes from $V \setminus \mathcal{N}^+(D_1)$ as both m and v.

Consider the subgraph S of G induced by $\tilde{A}$, v, m and L, the set of nodes that are enclosed in the outermost circle and that are neither in $\tilde{A} \subseteq V \setminus \mathcal{N}^+(D_1)$ nor already covered by D_1. Thus $S = (V_S, E_S)$ contains $|V_S| = |L| + |\tilde{A}| + |\{v, m\}| = |L| + 10$ nodes. Regarding the number of edges we claim that the cardinality of E_S is at least

$$\begin{aligned}|E_S| &\geq |\mathcal{N}_S(v)| + |\mathcal{N}_S(m)| \\ &\quad +4\max\{|\mathcal{N}_S(v)|, |\mathcal{N}_S(m)|\} - 18 \\ &\geq 3(|\mathcal{N}_S(v)| + |\mathcal{N}_S(m)| - 6).\end{aligned}$$

To see that this claim holds note that the subgraph S contains at least the edges to all neighbors of v and m in S and the edges incident to the four nodes from $\tilde{A}$ that are enclosed by two nested circles consisting of v, m and the four outer nodes from $\tilde{A}$ according the embedding. Remember that the residual degree of these four nodes from $\tilde{A}$ in the second step of the algorithm is at least as large as the residual degrees of v and m, as they would not have been chosen in Line 11 otherwise. By adding $4\max\{|\mathcal{N}_S(v)|, |\mathcal{N}_S(m)|\}$ we might count some edges twice, therefore we subtract 18 edges to account for the facts that (i) there might be up to $\binom{4}{2} = 6$ edges between pairs of the four considered nodes $d(a) \in D_2$, (ii) up to 8 edges between these four nodes and $\{v, m\}$ might exist, and (iii) each such node $d(a)$ might contribute 1 to its own residual degree by itself without the necessity for an edge.

The second construction step of Definition 6.4 ensures that $\tilde{A} \cap M = \emptyset$ by only adding nodes from $V \setminus M$ to V_H. Hence, the assumption that no other node from M is enclosed by the outermost circle implies that everything inside is covered by $\{v, m\}$. Therefore, it holds that

$$|\mathcal{N}_S(v)| + |\mathcal{N}_S(m)| \geq 2|\tilde{A}| + |L| = |L| + 16.$$

However, Lemma 6.2 lower bounds $|V_S|$ in terms of $|E_S|$, giving that

$$\begin{aligned}3(|L| + 10) &= 3|V_S| \\ &> |E_S| \\ &\geq 3(|\mathcal{N}_S(v)| + |\mathcal{N}_S(m)| - 6) \\ &\geq 3(|L| + 10),\end{aligned}$$

a contradiction.

Case (ii) is treated similarly, but it is much simpler. This time, the assumption that no node from M is enclosed by the outermost circle implies that all the nodes inside must be covered by m alone, as M is a DS. Since v and m are connected via the (at least)

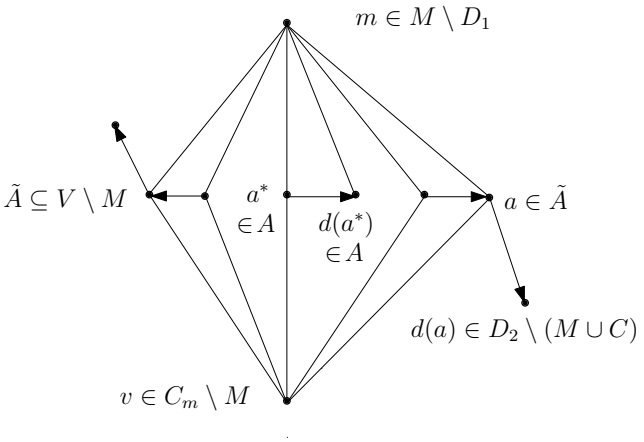

Fig. 6.4 Example of a subgraph considered in the second case of the proof of Lemma 6.6. Supposing there is no other node $m' \in M$ inside the outer circle, apart from v all neighbors of the node chosen by the innermost node from $\tilde{A}$ must also be neighbors of m.

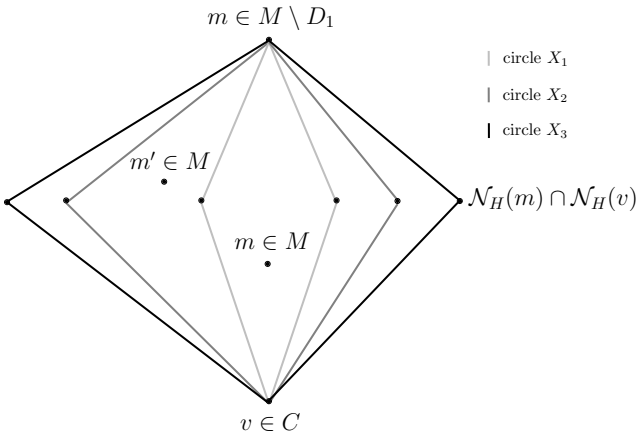

Fig. 6.5 Example of a sequence of three nested circles as considered in Corollary 6.7. Each pair of two voting nodes involved in a circle is deleted from H' after it has been accounted for. Therefore, all neighbors of the two outermost nodes from $\mathcal{N}_H(m) \cap \mathcal{N}_H(v)$ are not adjacent to nodes inside the innermost circle.

five nodes in $\tilde{A}$, for the node $d(a^*) \notin \{m, v\}$ elected into D_2 by the innermost node $a^* \in \tilde{A}$, it holds that $\mathcal{N}^+(d(a^*)) \setminus \mathcal{N}^+(m) \subseteq \{v\}$ (see Figure 6.4). However, there are at least two nodes in $\tilde{A} \subseteq V \setminus \mathcal{N}^+(D_1)$ that are not connected to $d(a^*)$, i.e., we get the contradiction that a^* would have preferred m over $d(a^*)$ in Line 11 of the algorithm. □

Next, we repeatedly delete nodes from H until eventually the preconditions of Lemma 6.5 are met. Arguing as in the proof of Lemma 6.6, we can account for deleted nodes by allocating them to enclosed nodes from $M \cup C$. Doing this carefully, we can make sure that no nodes from $M \cup C$ need to compensate for more than four deleted nodes.

Corollary 6.7 $|D_2| < 126|M|$.

Proof Fix an embedding of G and thus of all its subgraphs. We will argue with respect to this embedding only. We use the notation from the proof of Lemma 6.5. Starting from H, we iteratively delete nodes from $A := V_H \setminus (M \cup C)$ until we obtain a subgraph H' satisfying the prerequisites of the lemma. Assume that $H' := H$ violates one of the preconditions of Lemma 6.5. No matter which of the conditions (i) and (ii) from Lemma 6.5 is violated, we choose respective nodes $m \in M \setminus D_1$ and $v \in C_m$ satisfying precondition (i) or (ii) of Lemma 6.6 such that the smallest circle formed by the nodes m, v, and some $a_1, a_2 \in \tilde{A} := \mathcal{N}^+_{H'}(v) \cap \mathcal{N}_{H'}(m)$ enclosing an element $m' \in M$ has minimal area. We delete the two elements from $\tilde{A} \subseteq A$ participating in the circle. Since the area of the circle is minimal, there is no third element from $\tilde{A}$ enclosed in the circle.

We repeat this process until H' satisfies the preconditions of Lemma 6.5. We claim that we can account for deleted nodes in terms of nodes from $M \cup C$ in a way such that no element of $M \cup C$ needs to compensate for more than four deleted nodes. Whenever we delete a pair of nodes, we count a node from $M \cup C$ enclosed by the respective circle that has not yet been counted twice.

We need to show that this is indeed always possible. To see this, observe that the minimality of the enclosed area of a chosen circle X together with the planarity of G ensures that any subsequent circle X' either encloses this circle or its enclosed area is disjoint from the one of X. In the latter case, we obviously must find a different node from $M \cup C$ enclosed in X' than the one we used when deleting nodes from X. Hence, we need to examine the case when there are three nested circles X_1, X_2, and X_3 that occur in the construction. If the nodes $m \in M$ and $v \in C_m$ participating in each circle are not always the same, one node from the first such pair becomes enclosed by one of the subsequent circles.

Hence, the remaining difficulty is that we could have three such nested circles formed by nodes $m \in M$, $v \in C_m$, and three pairs of nodes from $\mathcal{N}_H(m) \cap \mathcal{N}_H(v)$ (see Figure 6.5). Any node chosen by a node $a \notin \{m, v\}$ lying on the outermost circle X_3 is separated from nodes enclosed by X_1. Therefore, nodes in M enclosed by X_1 can cover only nodes that are either not adjacent to the nodes from D_2 considered in Lemma 6.6 (when applied to H' after X_1 and X_2 have already been removed) or lie on X_1. Since the nodes on X_1 are m, v, and two of their shared neighbors in H, we can thus argue analogously

to the proof of Lemma 6.6 in order to find a node $m' \in M$ enclosed by X_3, but not enclosed by X_1.

Altogether, for each element of $M \cup C$ we remove at most two times two nodes each from A, i.e., in total no more than $4|M \cup C| \leq 28|M|$ nodes. To the remaining subgraph H', we apply Lemma 6.5, yielding

$$|D_2| < (28 + 98)|M| = 126|M|. \qquad \square$$

Having determined the maximum number of nodes that enter the dominating set in each step, it remains to assemble the results and finally state the approximation ratio our algorithm achieves.

Theorem 6.8 $|D| < 130|M|$.

Proof Combining Lemma 6.3 and Corollary 6.7, we obtain

$$\begin{aligned} |D| &\leq |M| + |D_1 \setminus M| + |D_2| \\ &< (1 + 3 + 126)|M| = 130|M|. \end{aligned} \qquad \square$$

Acknowledgements We would like to thank Topi Musto, Jukka Suomela, and the anonymous reviewers who helped in improving this article and its predecessors in many ways.

References

1. Alon, N., Babai, L., Itai, A.: A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. Journal of Algorithms **7**(4), 567–583 (1986)
2. Awerbuch, B.: Complexity of Network Synchronization. Journal of the ACM **32**(4), 804–823 (1985)
3. Awerbuch, B., Sipser, M.: Dynamic Networks are as Fast as Static Networks. In: Proc. 29th Symposium on Foundations of Computer Science (FOCS), pp. 206–219 (1988)
4. Barenboim, L., Elkin, M.: Distributed $(\Delta + 1)$-Coloring in Linear (in Δ) Time. In: Proc. 41st Symposium on Theory of Computing (STOC), pp. 111–120 (2009)
5. Basagni, S.: Distributed Clustering for Ad Hoc Networks. In: Proc. Fourth International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN), p. 310 (1999)
6. Chan, H., Luk, M., Perrig, A.: Using Clustering Information for Sensor Network Localization. In: Proc. Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 109–125 (2005)
7. Clark, B., Colbourn, C., Johnson, D.: Unit Disk Graphs. Discrete Mathematics **86**(1), 165–177 (1990)
8. Czygrinow, A., Hańćkowiak, M.: Distributed Almost Exact Approximations for Minor-Closed Families. In: Proc. 14th European Symposium on Algorithms (ESA), pp. 244–255 (2006)
9. Czygrinow, A., Hanckowiak, M.: Distributed Approximation Algorithms for Weighted Problems in Minor-Closed Families. In: Proc. 13th Computing and Combinatorics Conference (COCOON), pp. 515–525 (2007)
10. Czygrinow, A., Hańćkowiak, M., Wawrzyniak, W.: Fast Distributed Approximations in Planar Graphs. In: Proc. 22nd Symposium on Distributed Computing (DISC), pp. 78–92 (2008)
11. Deb, B., Nath, B.: On the Node-Scheduling Approach to Topology Control in ad hoc Networks. In: Proc. 6th Symposium on Mobile ad hoc Networking and Computing, pp. 14–26. ACM (2005)
12. Deo, N., Litow, B.: A Structural Approach to Graph Compression. In: Proc. 23rd Symposium on Mathematical Foundations of Computer Science (MFCS), pp. 91–101 (1998)
13. Diestel, R.: Graph theory. 2005. Grad. Texts in Math (2005)
14. Dijkstra, E.W.: Self-Stabilizing Systems in Spite of Distributed Control. Communications of the ACM **17**(11), 643–644 (1974)
15. Funke, S., Kesselman, A., Kuhn, F., Lotker, Z., Segal, M.: Improved Approximation Algorithms for Connected Sensor Cover. Wireless networks **13**(2), 153–164 (2007)
16. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
17. Gfeller, B., Vicari, E.: A Faster Distributed Approximation Scheme for the Connected Dominating Det Problem for Growth-Bounded Graphs. In: Proc. 6th Conference on Ad-hoc, Mobile and Wireless Networks, pp. 59–73. Springer-Verlag (2007)
18. Gupta, H., Navda, V., Das, S., Chowdhary, V.: Efficient Gathering of Correlated Data in Sensor Networks. ACM Transactions on Sensor Networks (TOSN) **4**(1), 4 (2008)
19. Heinzelman, W., Chandrakasan, A., Balakrishnan, H.: Energy-efficient Communication Protocol for Wireless Microsensor Networks. In: Proc. 33rd Hawaii Conference on System Sciences, p. 10 (2002)
20. Islam, K., Akl, S., Meijer, H.: Distributed Generation of a Family of Connected Dominating Sets in Wireless Sensor Networks. In: Proc. 5th Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 343–355. Springer (2009)
21. Israeli, A., Itai, A.: A Fast and Simple Randomized Parallel Algorithm for Maximal Matching. Information Processing Letters **22**(2), 77–80 (1986)
22. Johnson, D.: Approximation Algorithms for Combinatorial Problems. Journal of Computer and System Sciences **9**(3), 256–278 (1974)
23. Jurdzinski, T., Stachowiak, G.: Probabilistic Algorithms for the Wake-Up Problem in Single-Hop Radio Networks. Theory of Computing Systems **38**(3), 347–367 (2005)
24. Kuhn, F., Moscibroda, T.: Distributed Approximation of Capacitated Dominating Sets. In: Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures, pp. 161–170. ACM (2007)
25. Kuhn, F., Moscibroda, T., Wattenhofer, R.: Unit Disk Graph Approximation. In: Proc. 2nd Workshop on Foundations of Mobile Computing (DIALM-POMC), pp. 17–23 (2004)
26. Kuhn, F., Moscibroda, T., Wattenhofer, R.: Local Computation: Lower and Upper Bounds. CoRR **abs/1011.5470** (2010)
27. Kuhn, F., Wattenhofer, R.: Constant-Time Distributed Dominating Set Approximation. Distributed Computing **17**(4), 303–310 (2005)
28. Lenzen, C., Oswald, Y.A., Wattenhofer, R.: What Can Be Approximated Locally? Case Study: Dominating Sets in Planar Graphs. In: Proc. 20th Symposium on Parallelism in Algorithms and Architectures (SPAA), pp. 4–54 (2008)
29. Lenzen, C., Pignolet, Y.A., Wattenhofer, R.: What Can Be Approximated Locally? Case Study: Dominating Sets in Planar Graphs. Tech. rep., ETH Zurich (2010). ftp.tik.ee.ethz.ch/pub/publications/TIK-Report-331.pdf

30. Lenzen, C., Wattenhofer, R.: Leveraging Linial's Locality Limit. In: Proc. 22nd Symposium on Distributed Computing (DISC), pp. 394–407 (2008)
31. Lenzen, C., Wattenhofer, R.: Minimum Dominating Set Approximation in Graphs of Bounded Arboricity. In: Proc. 24th Symposium on Distributed Computing (DISC), pp. 510–524 (2010)
32. Linial, N.: Locality in Distributed Graph Algorithms. SIAM Journal of Computing **21**(1), 193–201 (1992)
33. Luby, M.: A Simple Parallel Algorithm for the Maximal Independent Set Problem. SIAM Journal of Computing **15**(4), 1036–1055 (1986)
34. Malkevitch, J.: The First Proof of Euler's Formula. Mitt. Math. Sem. Giessen (165) (1984)
35. Marathe, M., Breu, H., Hunt III, H.B., Ravi, S.S., Rosenkrantz, D.J.: Simple Heuristics for Unit Disk Graphs. Journal of Networks **25**, 59–68 (1995)
36. Métivier, Y., Robson, J.M., Saheb Djahromi, N., Zemmari, A.: An Optimal Bit Complexity Randomised Distributed MIS Algorithm. In: Proc. 16th Colloquium on Structural Information and Communication Complexity (SIROCCO), pp. 1–15 (2009)
37. Moscibroda, T., Wattenhofer, R.: Maximal Independent Sets in Radio Networks. In: Proc. 24th Symposium on the Principles of Distributed Computing (PODC), Las Vegas, Nevada, USA (2005)
38. Naor, M.: A Lower Bound on Probabilistic Algorithms for Distributive Ring Coloring. SIAM Journal on Discrete Mathematics **4**(3), 409–412 (1991)
39. Panconesi, A., Srinivasan, A.: On the Complexity of Distributed Network Decomposition. Journal of Algorithms **20**(2), 356–374 (1996)
40. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. Society for Industrial and Applied Mathematics (2000)
41. Raz, R., Safra, S.: A Sub-Constant Error-Probability Low-Degree Test, and a Sub-Constant Error-Probability PCP Characterization of NP. In: Proc. 29th Symposium on Theory of Computing (STOC), pp. 475–484 (1997)
42. Schneider, J., Wattenhofer, R.: A Log-star Distributed Maximal Independent Set Algorithm for Growth-Bounded Graphs. In: Proc. 27th Symposium on Principles of Distributed Computing (PODC), pp. 35–44 (2008)
43. Schneider, J., Wattenhofer, R.: What Is the Use of Collision Detection (in Wireless Networks)? In: Proc. 24th Symposium on Distributed Computing (2010)
44. Slavík, P.: A Tight Analysis of the Greedy Algorithm for Set Cover. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pp. 435–441. ACM (1996)
45. Wagner, K.: Über eine Eigenschaft der ebenen Komplexe. Mathematische Annalen **114**(1), 570–590 (1937)
46. Wan, P., Alzoubi, K., Frieder, O.: Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks. In: Proc. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), vol. 3, pp. 1597–1604. IEEE (2002)
47. Wang, Y., Wang, W., Li, X.: Distributed Low-Cost Backbone Formation for Wireless Ad Hoc Networks. In: Proc. 6th Symposium on Mobile ad hoc Networking and Computing, pp. 2–13. ACM (2005)
48. Wiese, A., Kranakis, E.: Local PTAS for Dominating and Connected Dominating Set in Location Aware Unit Disk Graphs. Approximation and Online Algorithms pp. 227–240 (2009)
49. Wu, J., Gao, M., Stojmenovic, I.: On Calculating Power-Aware Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Networks. In: International Conference on Parallel Processing (ICPP), p. 0346 (2001)
50. Wu, Y., Li, Y.: Construction Algorithms for k-connected m-dominating Sets in Wireless Sensor Networks. In: Proc. 9th Symposium on Mobile ad hoc Networking and Computing, pp. 83–90. ACM (2008)
51. Yao, A.: Some Complexity Questions related to Distributive Computing (preliminary report). In: Proceedings of the eleventh annual ACM symposium on Theory of computing, pp. 209–213. ACM (1979)