Fault-Tolerant Clustering in Ad Hoc and Sensor Networks

Fabian Kuhn
Microsoft Research Silicon Valley
1065 La Avenida
Mountain View, CA 94043
kuhn@microsoft.com

Thomas Moscibroda
Computer Engineering and
Network Laboratory
ETH Zurich, 8092 Zurich
moscitho@tik.ee.ethz.ch

Roger Wattenhofer
Computer Engineering and
Network Laboratory
ETH Zurich, 8092 Zurich
wattenhofer@tik.ee.ethz.ch

Abstract

In this paper, we study distributed approximation algorithms for fault-tolerant clustering in wireless ad hoc and sensor networks. A k-fold dominating set of a graph $G = (V, E)$ is a subset S of V such that every node $v \in V \setminus S$ has at least k neighbors in S. We study the problem in two network models. In general graphs, for arbitrary parameter t, we propose a distributed algorithm that runs in time $O(t^2)$ and achieves an approximation ratio of $O(t\Delta^{2/t} \log \Delta)$, where n and Δ denote the number of nodes in the network and the maximal degree, respectively. When the network is modeled as a unit disk graph, we give a probabilistic algorithm that runs in time $O(\log \log n)$ and achieves an $O(1)$ approximation in expectation. Both algorithms require only small messages of size $O(\log n)$ bits.

1 Introduction

Ad hoc and sensor networks have been envisioned in a large number of application fields. Moreover, there are a growing number of real (even commercial) systems that are being built, ranging from monitoring and surveillance, to medical applications, the observation of biological and chemical processes, and disaster relief. These networks are formed by autonomous nodes that communicate via radio, without any additional a-priori infrastructure. Typically, if two nodes are not within mutual transmission range, they communicate through intermediate nodes relaying their messages. In other words, the communication infrastructure is provided by the nodes themselves. Establishing and maintaining such a virtual infrastructure is an important challenge.

One important approach for dealing with the inherent lack of structure in ad hoc and sensor networks has been *dominating set* based *clustering* [1, 4, 8, 12, 22, 23]. Specifically, clustering allows the formation of virtual backbones, it improves the usage of scarce resources, such as bandwidth and energy, and clustering helps realizing spatial multiplexing in non-overlapping clusters. Clustering is also an effective way of improving the performance of routing algorithms [1, 23] and it is a building block for efficient network initialization [12, 18].

One key characteristics of wireless ad-hoc and particularly sensor networks is that *node failure* is an event of non-negligible, in some cases even high probability. This is particularly the case in sensor networks where the equipment is restricted to a minimum due to limitations in cost and weight. Battery driven sensor nodes may also stop working because they run out of energy supply. Secondly, the *shared wireless medium* is inherently less stable than wired media. This results in more packet losses and a lower throughput. A third aspect that has an influence on the required degree of redundancy and fault-tolerance is *mobility*, which is a key issue in ad hoc networks.

For all these reasons, hierarchical structures such as dominating sets are prone to fail unless they provide enough *fault-tolerance* or redundancy. In this paper, we therefore study the fault-tolerant version of the dominating set problem. Formally, in a graph $G = (V, E)$, a *k-fold dominating set* is a subset $S \subseteq V$ such that, each node $v \in V \setminus S$ has at least k dominators in S in its neighborhood. The minimum k-fold dominating set problem (k-MDS) asks for a k-fold dominating set of minimal cardinality. For $k = 1$, a 1-MDS problem is simply the standard minimum dominating set problem (MDS).

In this paper, we give upper bounds on the distributed approximability of the k-MDS problem in two different models, general graphs and unit disk graphs (UDG). In a UDG $G = (V, E)$, nodes are located in the Euclidean plane and there exists a communication link between two nodes u and v if the distance between the two nodes is at most 1. A node u can send a message to a node v only if there is a communication link between them. UDGs have been a popular model for modeling the characteristic wireless nature of communication in multi-hop radio networks. Since in reality, signal propagation does often not form clear-cut disks, we also study the problem in general graphs, which can be regarded as the pessimistic counterpart to the optimistic UDG model.

Clearly, algorithms that are based on maintaining a global view of the network are infeasible in the context of ad hoc or sensor networks. Hence, *distributed algorithms* that have a running time growing linearly (or sometimes even polylogarithmically) in the number of nodes are typically unemployable in large-scale ad hoc or sensor systems. Instead, we are interested in entirely *distributed and local algorithms* featuring a (very) *low time complexity*, even at the cost of somewhat suboptimal solutions. In particular, we are interested in the achievable trade-off between the amount of communication (time complexity) and the quality of the obtained solution (approximation ratio).

In this paper, we first present a distributed approximation algorithm. For arbitrary values of t, the algorithm achieves an $O(t\Delta^{2/t}\log\Delta)$ approximation in time $O(t^2)$. The algorithm first approximates the *fractional version* of the k-MDS problem in a distributed way. The final solution is then obtained using a distributed form of randomized rounding. This approximation upper bound is particularly interesting in view of the distributed approximability lower bound given in [13]. This lower bound implies that in $O(t)$ communication rounds, no (possibly randomized) algorithm can achieve an approximation ratio better than $\Omega(\Delta^{1/t}/t)$, even if message size is unbounded and nodes have unique identifiers. Specifically, this lower bound indicates that the time-approximation trade-off achieved by our algorithm is not too far from the optimum.

The second contribution in the paper is a a randomized algorithm that computes a k-fold dominating set in time $O(\log\log n)$ in UDGs, if nodes can sense distances to neighboring nodes. The algorithm achieves a constant approximation ratio in expectation. Furthermore, the algorithm uses only small messages of size $O(\log n)$ bits. This limited message size is important, because in all practical scenarios, the size of messages cannot be arbitrarily large. This algorithm for the UDG is based on an algorithm given in [7] for approximating the standard minimum dominating set problem (cf. Section 2).

The remainder of the paper is organized as follows. We give an overview over related work in Section 2. In Section 3, we formally introduce our model of computation. Our algorithms for general graphs and unit disk graphs are presented in Sections 4 and 5, respectively.

2 Related Work

MDS and the closely related minimum set cover problem are two of the first problems that have been shown to be NP-hard. The straightforward adaptation of the greedy set-cover algorithm gives an asymptotically optimal $O(\log\Delta)$ approximation [5], even for the fault-tolerant version [20]. On the other hand, it was shown that unless problems in NP can be solved by deterministic $n^{O(\log\log n)}$ time algorithms, no algorithm can approximate the minimum dominating set problem better than $\ln\Delta$, where Δ is the highest degree in the network [6].

Dominating sets have also been studied in the context of distributed computing. The first distributed approximation algorithms with provable guarantees for MDS was given in [17], a result that was subsequently improved in [19]. The first approximation algorithm achieving a polylogarithmic approximation appeared in [9]. This algorithm has an expected approximation ratio of $O(\log\Delta)$ in $O(\log n\log\Delta)$ communication rounds, where Δ is the highest degree in the network. In [16], an algorithm is given that requires $O(t^2)$ communication rounds for with an $O(t\Delta^{2/t}\log\Delta)$ approximation, for arbitrary values of t. This result has recently been improved to an approximation of $O(\Delta^{1/t}\log\Delta)$ in time $O(t^2)$ in [15]. These upper bounds are in contrast to lower bound for the distributed approximation of dominating sets given in [13]. In t communication rounds, no (possibly randomized) algorithm can achieve an approximation ratio better than $\Omega(n^{c/t^2}/t)$ or $\Omega(\Delta^{1/t}/t)$, for some small constant c. This implies that in arbitrary graphs, every algorithm requires at least time $\Omega(\log\Delta/\log\log\Delta)$ or $\Omega(\sqrt{\log n/\log\log n})$ in order to achieve a constant (or polylogarithmic, for that matter) approximation. Clearly, all these lower bounds carry over to the k-MDS problem as well.

The only previously known upper bound on the distributed approximability of the k-fold dominating set problem in general graphs has been given in [9]. In this paper, Jia, Rajaraman, and Suel propose an algorithm that achieves an expected approximation ratio of $O(\log\Delta)$ in time $O(\log n\log\Delta\log k)$ w.h.p.

In unit disk graphs, MDS remains NP-hard, but constant approximations become possible. Numerous distributed algorithms have been proposed for clustering multi-hop wireless networks, but most of them are of heuristic nature and do not provide worst-case guarantees, e.g., [3, 8, 23]. The distributed algorithms in [1, 22] compute a constant approximation to the regular, non-redundant MDS problem. However, their algorithm requires time $O(n)$ in the worst-case. Recently, [14] has proposed an algorithm that deterministically computes (among other problems) a constant approximation to the minimum dominating set problem in unit disk graphs in time $O(\log^* n)$. However, their algorithm requires messages that are much larger than $O(\log n)$ which may be problematic in practical settings. Clustering under harsher network models that also takes into account asynchronous wake-up and collisions has been considered in [12].

The paper most related to our algorithm in the UDG model is [7]. In this excellent paper, the authors give an algorithm that computes a constant approximation to the MDS problem in sublogarithmic running time. Unfortunately, the analysis given in [7] does not seem to be rigorous

enough. Specifically, it is implicitly assumed that if for a recursive random variable X_i, it holds that $E[X_{i+1}] \leq \sqrt{X_i}$, then it also holds that $E[X_{i+2}] \leq \sqrt{\sqrt{X_i}}$, and consequently $E[X_{i+j}] \leq X_i^{1/2^j}$. However, this argument appears to be based on the assumption that $E[\sqrt{X}] \leq \sqrt{E[X]}$ which is not true in general. Our UDG algorithm for the k-MDS problem takes a similar algorithmic approach as [7] for the MDS problem, but we attempt to provide a more rigorous analysis. For the special case of setting $k = 1$, our UDG algorithm gives a constant approximation in time $O(\log \log n)$.

In contrast to the MDS problem, much less is known about the distributed approximation of the k-fold dominating set in unit disk graphs. In fact, we are not aware of any explicit distributed approximation algorithm that addresses this problem in this model.

3 Model and Notation

We describe the network using the standard *message passing model*. The network is modelled as an undirected graph $G = (V, E)$. Two nodes $u, v \in V$ of the network are connected by an edge $(u, v) \in E$ whenever there is a direct bidirectional communication channel connecting u and v. For simplicity, we assume a synchronous communication model where time is divided in rounds. In each round, every node can send a message to each of its neighbors in G. Message size is restricted to $O(\log n)$ bits per message, i.e., every message can contain only a constant number of node identifiers. The time complexity of an algorithm is the number of rounds it needs to complete. Note that at the cost of higher message complexity, every synchronous message passing algorithm can be turned into an asynchronous algorithm with the same time complexity [2].

While in Section 4.1, we do not make further assumptions about the underlying network graph, we study the case of unit disk graphs in Section 5.1. The *unit disk graph* (UDG) model has become a quasi-standard for the analysis of algorithms designed for wireless networks. In a UDG $G = (V, E)$, nodes are located in the Euclidean plane. There is an edge between two nodes u and v iff the Euclidean distance between u and v is at most 1. As in [7], we assume in Section 5.1 that nodes can sense the distance between themselves and their neighbors.

In this paper, logarithms are to the base 2, unless the base is explicitly stated. In Section 5.2, we use the the i^{th} logarithm of a value defined as

Definition 3.1. *For an integer $i > 0$, the i^{th} logarithm of n is defined as*

$$\log_2^{(i)} n := \begin{cases} \log_2 n & , \quad i = 1 \\ \log_2 (\log_2^{(i-1)} n) & , \quad i > 1 \end{cases}$$

For technical reasons in the proof, we will also use a slightly alternate version, namely, we define the base of the *outermost* logarithm to be a constant ξ, instead of 2. Formally, $\log^{(i)} n = \log_\xi (\log_2^{(i-1)} n)$. Furthermore, $\log^* n := \min \{i \mid \log^{(i)} n \leq 2\}$ is the number of times we have to take the logarithm before the value decreases below the constant 2.

Throughout the paper, we use the following nomenclature. N_v denotes the set of neighbors of node v (including v). By abuse of notation, where appropriate N_v also represents the set of neighboring identifiers. In Section 5, we additionally define $N_v(\tau) := \{w \in V \mid dist(v, w) \leq \tau\}$ as the set of neighbors of node v that are at most at distance τ from v.

4 Algorithm for General Graphs

Our algorithm for general graphs consists of two parts. First a fractional solution for the k-fold dominating set problem is computed. This fractional solution is convereted into an integer one by a distributed randomized rounding scheme. The two parts are described in Sections 4.1 and 4.2, respectively.

4.1 Solving the Fractional Problem

The fractional version of minimum k-fold dominating set can be seen as a linear programming relaxation of the k-MDS problem. We obtain the following pair of primal and dual linear programs, in the following denoted by $(\mathrm{PP})$and (DP), respectively.

$$\begin{aligned} \min \sum_{i=1}^{n} x_i \qquad & \qquad \max \sum_{i=1}^{n} (k_i y_i - z_i) \\ \text{s.t.} \forall i : \sum_{j \in N_i} x_j \geq k_i \qquad & \qquad \text{s.t.} \forall i : \sum_{j \in N_i} y_j - z_i \leq 1 \\ 0 \leq x_i \leq 1 \qquad & \qquad y_i, z_i \geq 0 \end{aligned}$$

In the primal LP (PP) (left-hand side), there is a variable x_i for every node v_i $(i = 1, \ldots, n)$. Further, for every node v_i, there is a parameter k_i denoting the number of times v_i has to be covered. The goal is to minimize the sum of all x-variables under the condition that the sum of the x-values in the neighborhood N_i of each node v_i is at least k_i. In the dual LP (DP) (right-hand side), for every node v_i, there are variables y_i and z_i. Note that the problem defined by (PP) does not exactly match the definition for k-fold dominating sets of Section 1. In (PP), the number of times k_i a node v_i needs to be covered can vary for different nodes whereas in the definition of Section 1, we have $k_i = k$ for all i. Further, according to the definition of Section 1, only nodes which are not in the dominating set have to be covered k times. Nodes in the dominating set cover themselves

and need not be covered by neighboring nodes. In favor of a slightly simpler and better readable algorithm, (PP) demands that also nodes v_i of the dominating set have to be covered by $k_i - 1$ other nodes. We would like to point out that our algorithm can be adapted to the case where dominating set nodes cover themselves. The results concerning time complexity and approximation ratio given by Theorems 4.5 and 4.6 remain the same. It would also be possible to extend our algorithm to also solve the weighted version of the k-MDS problem.

The distributed algorithm to compute a solution for the linear program (PP) is given by Algorithm 1. From a very general point of view, the algorithm and its analysis can be seen as a distributed version of the greedy k-MDS-algorithm as described in [21]. In the greedy algorithm, we start with an empty set S. In each step, a node with a maximal number of not yet completely covered neighbors is added to S.

The main problem of applying the greedy algorithm in a distributed environment is the synchronization of different nodes with equal or similar numbers of uncovered neighbors capable of joining the dominating set. We have to solve a classical symmetry breaking problem. Solving a fractional problem instead of its integral counterpart often is a good way to avoid most problems arising in the context of symmetry breaking [11]. Intuitively, whenever there are q neighbors of a node u which could all join the dominating set according to the greedy condition, instead of selecting one of the q nodes, we can increment the x-value of each of them by $1/q$.

The main techniques of our solution are borrowed from a distributed algorithm for the standard dominating set problem [16]. During the execution of Algorithm 1, all nodes v_i start with $x_i = 0$ and increase their x-values over time. We say that a node v_i is colored gray as soon as the sum of the weights x_j for $v_j \in N_{v_i}$ exceeds 1, that is, as soon as the node is completely covered. Initially all nodes are colored white. The number of white nodes $v_j \in N_{v_i}$ at a given time is called the dynamic degree of v_i and denoted by $\tilde{\delta}_i$. Initially, all nodes are white and thus $\tilde{\delta}_i = \delta(v_i) + 1$, where $\delta(v_i)$ denotes the degree of v_i.

Lemma 4.1. *For all nodes v_i with $x_i < 1$, the dynamic degree $\tilde{\delta}_i \leq (\Delta+1)^{(p+1)/t}$ at all times.*

Proof. We prove the lemma by induction over s. For $p = t-1$, the condition simplifies to $\tilde{\delta}_i \leq (\Delta+1)$ which is always true. In the last iteration of the inner-loop (q-loop), all v_i with $\tilde{\delta}_i \geq (\Delta+1)^{p/t}$ set $x_i := 1$ which clearly makes the condition of the lemma true. □

During an execution of Algorithm 1, a primal solution $\boldsymbol{x}$ and a dual solution $(\boldsymbol{y}, \boldsymbol{z})$ for the linear programs (PP) and (DP) are computed, respectively. The following lemmas establish properties concerning the computed solutions.

Algorithm 1 Distributed LP Approximation

```
1:  x_i := 0; δ̃_i := δ(v_i) + 1; col_i := white;
2:  c_i := 0; ∀j : α_{j,i} := 0; ∀j : β_{j,i} := 0;
3:  for p := t − 1 to 0 by −1 do
4:    for q := t − 1 to 0 by −1 do
5:      if (x_i < 1) ∧ (δ̃_i ≥ (Δ + 1)^{p/t}) then
6:        x_i^+ := min{ 1/(Δ+1)^{q/t}, 1 − x_i };
7:        x_i := x_i + x_i^+
8:      fi;
9:      send x_i, x_i^+, δ̃_i to all neighbors;
10:     if col_i = white then
11:       c_i^+ := Σ_{j∈N_i} x_j^+;
12:       λ := min{1, (k_i − c_i)/c_i^+};
13:       c_i := c_i + c_i^+;
14:       for all j ∈ N_i do
15:         β_{j,i} := β_{j,i} + λx_j^+/(Δ+1)^{p/t};
16:         α_{j,i} := α_{j,i} + λx_j^+
17:       od;
18:       if c_i ≥ k_i then
19:         col_i := gray;
20:         y_i := 1/(Δ+1)^{p/t}
21:       fi
22:     fi;
23:     send col_i to all neighbors;
24:     δ̃_i := |{j ∈ N_i | col_j = white}|
25:   od
26: od;
27: z_i := Σ_{j∈N_i} (α_{i,j} y_j − β_{i,j})
```

Lemma 4.2. *Let $\mathcal{OPT}$ be the objective value of an optimal solution of* (PP). *The sum of the variables $\beta_{i,j}$ can be bounded in the following way:*

$$\big((\Delta+1)^{1/t}+1\big)\cdot\sum_{i=1}^{n}\sum_{j\in N_i}\beta_{i,j} \;\geq\; \sum_{i=1}^{n} x_i - t(\Delta+1)^{1/t}\cdot\mathcal{OPT}$$

Proof. In Line 13 of Algorithm 1, c_i is incremented by the sum of the increases of the x-values of nodes in N_i. Thus, c_i stores the number of times node v_i is covered. As long as the value of c_i is at most k_i after Line 13, λ is set to 1 in Line 12. If $\lambda = 1$, for each of the $\tilde{\delta}_j$ white neighbors v_i of a node v_j, $\beta_{j,i}$ is increased by $x_j^+/(\Delta+1)^{p/t}$. Because $\tilde{\delta}_j \geq (\Delta+1)^{p/t}$, the total increase of β-values upper bounds the total increase of x-values. Consequently, in some sense, we use the β-values in the dual LP (DP) to pay for the x-values in the primal LP (PP), a technique known as dual fitting [21]. The only time, λ can be smaller than 1 is when v_j becomes covered k_i times. There, λ can be arbitrarily close to 0 in which case $\beta_{i,j}$ does not pay its part of x_i^+. To prove the lemma, we have to show that the β-values nevertheless pay for most of the x-values.

Consider the x-values and β-values of a single iteration of the outer loop (p-loop) of the algorithm. Assume that v_i becomes covered k_i times in iteration q_i of the inner loop (q-loop). There are two cases which we have to consider, namely, $q_i < t-1$ and $q_i = t-1$.

If $q_i < t-1$, we know that all neighbors v_j of v_i which increase x_j in iteration q_i have already increased x_j in iteration q_i+1. During iteration q_i+1, v_i remained white and thus λ was 1 when computing the $\beta_{j,i}$ share of x_j^+ of iteration q_i+1. Because x_j^+ increases at most by a factor of $(\Delta+1)^{1/t}$ per inner-loop iteration, the $\beta_{j,i}$-part of iteration q_i+1 pays for the x_j-increases of iterations q_j+1 and q_j up to a factor of $(\Delta+1)^{1/t}+1$.

If $q_i = t-1$, the above argumentation does not hold because there is no iteration q_i+1 of the inner loop. It can therefore be that the x-increases of iteration q_i are not paid for by β-values. However since each of these x-increases is at most $1/(\Delta+1)^{(t-1)/t}$, the sum of the x-increases of iteration q_i is at most $n/(\Delta+1)^{(t-1)/t}$. Combining the two cases, we get that in each iteration of the outer loop, up to a factor of $(\Delta+1)^{1/t}+1$, the β-increases pay for all x-increases except for the ones from inner-loop iteration $t-1$. Summed over all outer-loop iterations, we have

$$\left((\Delta+1)^{1/t}+1\right)\cdot\sum_{i=1}^{n}\sum_{j\in N_i}\beta_{i,j} \;\geq\; \sum_{i=1}^{n}x_i - \frac{tn}{(\Delta+1)^{\frac{t-1}{t}}}.$$

Because in a (fractional) dominating set every node can cover at most $\Delta+1$ nodes, the size of an optimal dominating set is at least $n/(\Delta+1)$. Plugging this into the above inequality completes the proof. □

Lemma 4.3. *For the dual solution $(\boldsymbol{y},\boldsymbol{z})$ constructed by Algorithm 1, we have $\sum_{i=1}^{n}(k_iy_i-z_i)=\sum_{i=1}^{n}\sum_{j\in N_i}\beta_{i,j}$.*

Proof. By the definition of λ in Line 12, we have $\sum_{j\in N_i}\alpha_{j,i}=k_i$. With this, the lemma follows because

$$\begin{aligned}\sum_{i=1}^{n}(k_iy_i-z_i) &= \sum_{i=1}^{n}\left[k_iy_i-\sum_{j\in N_i}(\alpha_{i,j}y_j-\beta_{i,j})\right]\\ &= \sum_{i=1}^{n}\left(k_i-\sum_{j\in N_i}\alpha_{j,i}\right)y_i+\sum_{i=1}^{n}\sum_{j\in N_i}\beta_{i,j}.\end{aligned}$$

□

Lemma 4.4. *If the* (PP) *is feasible, the computed primal solution $\boldsymbol{x}$ is a feasible solution of* (PP). *The solution $(\boldsymbol{y},\boldsymbol{z})$ for* (DP) *is feasible up to a factor of $t(\Delta+1)^{1/t}$.*

Proof. To show that the primal solution $\boldsymbol{x}$ is feasible, let us take a look at the very last iteration of the inner loop where $p=q=0$. All nodes v_i which still have a white node in their neighborhood set x_i to 1. Thus, every node v_j which can be covered ($k_j \leq \delta(v_j)+1$) becomes covered.

To see that the dual solution $(\boldsymbol{y},\boldsymbol{z})$ is feasible up to a factor of $k(\Delta+1)^{2/k}$, we have to prove that all variables are non-negative and that Inequality (1) holds.

$$\forall i:\sum_{j\in N_i}y_j-z_i\leq t(\Delta+1)^{1/t}. \tag{1}$$

It is clear the $y_i\geq 0$ for all i. To see that $z_i\geq 0$, observe that $\alpha_{i,j}y_j\geq\beta_{i,j}$ (see Lines 15 and 20).

Let us now prove Inequality (1). Let us first look at the value of $\sum_{j\in N_i}y_j-z_i$ for outer-loop iterations where $x_i<1$ at the beginning. By Lemma 4.1, v_i has at most $(\Delta+1)^{(p+1)/t}$ white neighbors. Because only white nodes increase their y-values, $\sum_{j\in N_i}y_j$ grows by at most $(\Delta+1)^{(p+1)/t}/(\Delta+1)^{p/t}=(\Delta+1)^{1/t}$ in an outer-loop iteration where $x_i<1$ at the beginning.

Let us now consider the outer-loop iterations where $x_i=1$ at the beginning. If there is a node $v_j\in N_i$ which remains white after x_i is set to 1, we have $\alpha_{i,j}=1$. Let N_i' denote the set of white nodes in N_i just after setting x_i to 1, then

$$\sum_{v_j\in N_i'}y_j-z_i=\sum_{v_j\in N_i'}(1-\alpha_{i,j})y_j+\sum_{v_j\in N_i'}\beta_{i,j}\leq(\Delta+1)^{1/t}.$$

□

Having bounded the degree of infeasibility in Lemma 4.4, we can obtain the main theorem.

Theorem 4.5. *For arbitrary t, Algorithm 1 computes a feasible solution for the linear program* (PP) *in time* $\mathrm{O}(t^2)$. *The approximation ratio of the algorithm is at most $t\cdot\left((\Delta+1)^{2/t}+(\Delta+1)^{1/t}\right)$.*

Proof. For the time complexity note that every iteration of the inner loop can be computed in 2 rounds and that the number of iterations is t^2.

By Lemma 4.4, all inequalities of (DP) are satisfied up to a factor of $\kappa:=t(\Delta+1)^{1/t}$. Dividing all dual variables by κ therefore results in a feasible dual solution. Let $\overline{y}_i=y_i/\kappa$ and $\overline{z}_i=z_i/\kappa$ ($i=1,\ldots,n$) be such a feasible dual solution. By Lemma 4.3 and by LP duality, we have

$$\sum_{i=1}^{n}\sum_{j\in N_i}\beta_{i,j} \;=\; \kappa\cdot\sum_{i=1}^{n}(k_i\overline{y}_i-\overline{z}_i) \;\leq\; \kappa\mathcal{OPT}$$

where $\mathcal{OPT}$ is defined as in Lemma 4.2. By Lemma 4.2 we can conclude the proof as

$$\begin{aligned}\sum_{i=1}^{n}x_i &\leq \left((\Delta+1)^{1/t}+1\right)\cdot\sum_{i=1}^{n}\sum_{j\in N_i}\beta_{i,j}+\kappa\mathcal{OPT}\\ &\leq t\cdot\left((\Delta+1)^{2/t}+(\Delta+1)^{1/t}\right)\cdot\mathcal{OPT}\end{aligned}$$

□

Algorithm 2 Distributed Randomized Rounding

1: $p_i := \min\{1,\ x_i \cdot \ln(\Delta+1)\}$;
2: $x_i' := \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{otherwise} \end{cases}$
3: **send** x_i to all neighbors;
4: **if** $\sum_{j\in N_i} x_j' < k_i$ **then**
5: **send** REQ to $k_i - \sum_{j\in N_i} x_j'$ neighbors $v_\ell \in N_i$ with $x_\ell' = 0$
6: **fi**;
7: **if** REQ is received **then** $x_i' := 1$ **fi**

Remark: Setting, for instance, $t = \mathrm{O}(\log\Delta)$ in Algorithm 1, the algorithm achieves an approximation ratio of $\mathrm{O}(\log\Delta)$ in $\mathrm{O}(\log^2\Delta)$ rounds.

4.2 Randomized Rounding

The fractional solution computed by Algorithm 1 can be converted into an integral one (dominating set) by applying Algorithm 2, a distributed variant of a standard randomized rounding scheme. The algorithm starts with a solution $x_1,\ldots,x_n$ for (PP) and computes an integral solution which is denoted by $x_1',\ldots,x_n'$. The following theorem shows that in expectation, the size of the obtained solution is only by a factor of roughly $\log\Delta$ larger than the objective value of the solution for (PP).

Theorem 4.6. *Starting with a ρ-approximate solution for the linear program (PP), Algorithm 2 computes an integer solution with approximation ratio $\rho\log\Delta + \mathrm{O}(1)$ in constant time.*

Proof. A variable x_i' can be set to 1, that is, v_i can be chosen as dominator in Lines 2 and 5. We introduce two random variables X and Y denoting the number of nodes joining the dominating set in Lines 2 and 5, respectively.

By linearity of expectation, we have

$$\mathrm{E}[X] \;=\; \ln(\Delta+1)\cdot\sum_{i=1}^{n} x_i \;\le\; \ln(\Delta+1)\cdot\rho\cdot\mathcal{OPT}.$$

In order to bound the expected value of Y, we look at the probability q_i that v_i is not covered k_i times after Line 2. We distinguish two cases. If $k_i = 1$, we have

$$\begin{aligned} q_i &= \prod_{j\in N_i}(1-p_j) \;\le\; \prod_{j\in N_i}\left(1-\frac{\sum_{j\in N_i}p_j}{|N_j|}\right)^{|N_j|} \\ &\le \left(1-\frac{\ln(\Delta+1)}{|N_j|}\right) \;\le\; e^{-\ln(\Delta+1)} \;=\; \frac{1}{\Delta+1}. \end{aligned}$$

If $k_i > 1$, we can bound q_i using Chernoff and get

$$q_i \;\le\; \left(\frac{e^{-\frac{\ln(\Delta+1)-1}{\ln(\Delta+1)}}}{\frac{1}{\ln(\Delta+1)}^{\frac{1}{\ln(\Delta+1)}}}\right)^{k_i\ln(\Delta+1)} \;\in\; \mathrm{O}\left(\frac{1}{k_i(\Delta+1)}\right).$$

Note that $k_i \ge 2$. The expected value of Y can therefore be bounded as follows:

$$\begin{aligned} \mathrm{E}[Y] &\le \sum_{i=1}^{n} k_i q_i \;\in\; \sum_{i=1}^{n} k_i\cdot\mathrm{O}\left(\frac{1}{k_i(\Delta+1)}\right) \\ &= \mathrm{O}\left(\frac{n}{\Delta+1}\right) \;=\; \mathrm{O}(\mathcal{OPT}). \end{aligned}$$

Combining the obtained bounds for $\mathrm{E}[X]$ and $\mathrm{E}[Y]$ completes the proof. □

Remark: In Algorithms 1 and 2, it is implicitely assumed that all nodes of the graph know the maximum degree Δ. Using techniques described in [16, 11], it is possible to get rid of this assumption.

5 Algorithm for Unit Disk Graphs

5.1 Algorithm

In this section, we show that an algorithm very similar to the one of [7] can be turned into a fault-tolerant algorithm for the dominating set problem by adding an additional step to the algorithm. Algorithm 3 describes the details of the process, which is executed by all nodes in the system in a distributed way. In a first phase—which is essentially equivalent to the algorithm proposed in [7]—the algorithm selects a simple (not fault-tolerant) set of *leaders* whose cardinality is within a constant factor of the optimal solution in expectation. In a second phase, this dominating set consisting of the leaders is extended to a fault-tolerant dominating set with a constant approximation ratio. As for the first phase of the algorithm, we present a new analysis of the algorithm's approximation guarantee.

Part I of Algorithm 3 works by repeatedly decreasing the number of *active nodes*, that is, the nodes that may eventually become leaders. Initially, all nodes are active. As soon as a node becomes *passive*, it decides not to become a decides not to become a dominator. Part I proceeds in $\log_\xi\log n$ rounds, where $\xi = 3/2$. In each round r_i, an active node v considers only those active neighbors that are within distance θ of v. In a sense, θ represents a node's *transmission range* in a given round. This range θ is doubled in every round.

At the outset of a round r_i, every active node chooses a random identifier $\mathrm{ID}_i(v)$ in the range $[1,\ldots,n^4]$, which ensures with high probability that no two nodes in the network will ever choose the same ID. Each active node v

Algorithm 3 UDG algorithm (code for node v)

```
1:  (* PART I *)
2:  a(v) := true;  leader(v) := false;
3:  ξ = 3/2;  θ := (1/2) (log n)^{-1/log ξ};
4:  for i := 1 to log_ξ log n do
5:     randomly choose ID_i(v) from [1, ..., n^4];
6:     send (a(v), ID_i(v)) to all w ∈ N_v(θ);
7:     recv (a(w), ID_i(w)) from all w ∈ N_v(θ);
8:     A_v := {w ∈ N_v(θ) | a(w) = true};
9:     send M to w ∈ A_v with highest ID_i(v);
10:    if not recv M from any w ∈ A_v then
11:       a(v) := false; stop
12:    fi
13:    θ := 2θ
14: od
15: if a(v) = true then leader(v) := true fi;
16: (* PART II *)
17: c(v) := |{u ∈ N_v | leader(u) = true}|;
18: U(v) := {u ∈ N_v | c(v) < k};
19: while U(v) ≠ ∅ do
20:    select k nodes u_1, ..., u_k ∈ U(v);
21:    inform u_i to set leader(u_i) := true;
22:    c(v) := |{u ∈ N_v | leader(u) = true}|;
23:    U(v) := {u ∈ N_v | c(v) < k}
24: od
```

then elects among its active neighbors (with regard to the restricted transmission range θ) the one active node with the highest ID (Line 8), possibly itself. If an active node is elected by at least one other active node, it remains active. If not, it becomes passive and stops executing Part I. Nodes that have remained active throughout all $\log_\xi \log n$ rounds become leaders.

As for notation, θ_i denotes the transmission range θ of active nodes in round r_i. As θ is doubled in every round, it holds that $\theta_i = 2^{i-1}/(\log n)^{1/\log \xi}$. During Part I, the indicator variable $a(v)$ is **true** for active nodes, and **false** for passive nodes. Finally, S is the set of selected leaders in Part I.

5.2 Analysis

In this section, we prove the performance guarantees of Algorithm 5.1. Without affecting the asymptotic results, we omit ceilings throughout this section for succinctness of presentation. For each round r_i, we cover the plane with imaginary disks C_i of radius $\theta_i/2$ in a hexagonal lattice, as shown in Figure 1. Let D_i be the disk centered at the center of C_i having radius $3\theta_i/2$. As shown in Figure 1, D_i is (fully or partially) covering 19 smaller disks C_i. The transmission range in round r_i being θ_i, every node in a disk C_i can reach all other nodes in C_i, whereas nodes outside D_i are unable to cover any nodes in C_i. Let C be a disk with radius $1/2$. In round r_i, we write $C_i \in C$ to denote all disks C_i that cover C.

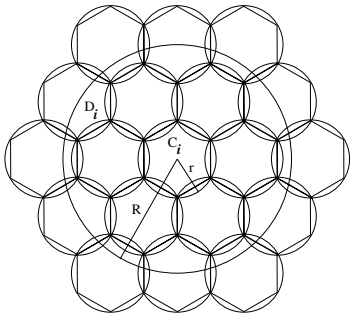

Figure 1. Disks C_i and D_i of round r_i.

It was shown in [7] that in Part I of Algorithm 3, every node is either a leader, or is covered by a leader.

Lemma 5.1 ([7]). *The set of leaders produced by Algorithm 3 forms a correct dominating set, i.e., for all nodes $v \in V \setminus S$, it holds that $|S \cap N_v| \geq 1$.*

Proof. At the end of Algorithm 3, all *active* nodes become leaders. To prove the claim for passive nodes, let $A_v(\theta_i)$ denotes the set of active nodes within distance θ_i of v. We prove the claim by induction over the algorithm's rounds. Specifically, we show that at the beginning of round r_i, it holds for every passive node v that $|A_v(\theta_i)| \geq 1$.

For $i = 1$, the claim is true because every node is active. In any subsequent round, if an active node v has no active neighbors in its current neighborhood, v remains active. Consider an arbitrary node v. Let r_q be the first round in which v becomes passive. By the above argument, it holds that $|A_v(\theta_q)| \geq 1$, which establishes the base case. For the induction step, assume that the claim holds for round r_i, i.e., v is covered by an active node within distance θ_i at the beginning of r_i. We now show that v remains covered by an active node. Let $w \in A_v(\theta_i)_v$ be a covering active node. In case w remains active in round r_i, v remains covered. Otherwise, w must have elected a node w' in round r_i. Since the transmission range in round r_i is θ_i, it holds that $dist(v, w') \leq 2\theta_i = \theta_{i+1}$ by triangle inequality. Hence, w' is in $A_v(\theta_{i+1})_v$, covering v at the beginning of round r_{i+1}, which concludes the induction step. Finally, the claim follows from the fact that the final transmission radius is

$$\theta_{\log_\xi \log n} = \frac{2^{\log_\xi \log n}}{2 \cdot (\log n)^{1/\log \xi}} = \frac{1}{2}$$

and consequently, every passive node is covered by an active node in distance at most 1. □

Next, we show that the expected number of dominators is within the claimed factor of the optimal solution. The next

lemma lower bounds the decrease of the number of active nodes in each round. Specifically, consider a disk C_i in a round r_i, and the corresponding disk D_i. The next lemma from [7] establishes a relationship between $|D_i|$ and $|C_{i+1}|$.

Lemma 5.2 ([7]). *Let m_i denote the number of active nodes in D_i in round r_i and let x_i and x_i' be the number of active nodes in C_i at the beginning and at the end of r_i, respectively. It holds that $x_i' \leq \delta\sqrt{m_i}\ln m_i$ with probability at least $1 - \gamma m_i^{-\ln m_i - 1 - \frac{2}{\log \xi}}$, for some constants γ and δ.*

Lemma 5.2 allows us to obtain a bound on the decrease of active nodes during the course of Part I. Intuitively, in every successful round, the number of active nodes decreases by a factor of $\delta\sqrt{m_i}\ln m_i$ per disk. Showing that the probability of having only good rounds is high enough in expectation then leads to the desired result. Turning this intuition into a rigorous argument requires some technical work. First, we require a simple geometric lemma.

Lemma 5.3. *Let $\alpha(i)$ be the number of disks of radius $\theta_i/2$ to completely cover a disk C of radius $1/2$. Defining $\eta = \frac{16\pi}{3\sqrt{3}}$, it holds that*

$$\alpha(i) \;<\; \frac{\eta}{4\theta_i^2} \;=\; \eta(\log n)^{\frac{2}{\log \xi}} \cdot 2^{-2i}.$$

Proof. The limit of the ratio of the area of C to the area of the smaller disks is $\frac{3\sqrt{3}}{2\pi}$ [10]. All disks intersecting C are completely inside the disk C', where C' has radius $r(C') = \frac{1}{2} + \theta_i$. Hence, we can write

$$\frac{(1/2+\theta_i)^2\,\pi}{\alpha(i)\cdot(\theta_i/2)^2\,\pi} \;\geq\; \frac{3\sqrt{3}}{2\pi}.$$

The lemma now follows by solving for $\alpha(i)$ and plugging in the definitions of η and θ_i. □

In the sequel, we consider a set of disks C_i of radius $\theta_i/2$ that completely cover C for each round r_i. Fix one such C_i and let m_i denote the number of nodes located in the corresponding disk D_i of radius $3\theta_i/2$ as shown in Figure 1. Similarly, let x_i, x_i', and m_i be defined as in Lemma 5.2 with regard to C_i and D_i, i.e., x_i and x_i' are the number of active nodes in C_i at the beginning and the end of round r_i, respectively. Notice that because the node's transmission range in r_i is θ_i, all nodes within C_i can hear each other. Hence, we can apply Lemma 5.2 to each such C_i individually. Unfortunately, we face the problem that the bound implied by Lemma 5.2 only holds with a certain *probability*. Particularly, the high probability result becomes increasingly useless as m_i decreases. For instance, once m_i is in the order of $O(\log\log n)$, Lemma 5.2 holds merely with probability $O(\log\log n)^{O(\log\log\log n)}$ which is asymptotically smaller than any polynomially bounded function in n.

On the other hand, once m_i is small, the consequences of Lemma 5.2 failing to hold may not be as grave with regard to the expected number of leader in the end. To formalize this intuition, we partition the $\log_\xi \log n$ rounds of Part I into $\log^* n$ *phases* consisting of a decreasing number of rounds.

Definition 5.4. *Let r_i denote the i^{th} round of Algorithm 3. For $j = 1, \ldots, \log^* n$,* phase P_j *consists of all rounds r_i such that*

$$\log_\xi \log n - \log^{(j+1)} n \;\leq\; i \;<\; \log_\xi \log n - \log^{(j+2)} n.$$

Hence, by Definition 5.4, phase P_j consists of $\log^{(j+1)} n - \log^{(j+2)} n$ rounds. In phase P_j, we call a disk C_i *active* if and only if $m_i \geq \log^{(j)} n$, i.e., when the number of active nodes in the "neighborhood" (the larger disk D_i) of C_i is not too small. During the first phase, for instance, a disk C_i is active if $m_i \geq \log_\xi n$.

We call a round r_i *good* for an active disk C_i if $x_i' \leq \delta\sqrt{m_i}\ln m_i$ holds, that is, when the random experiment described in Lemma 5.2 succeeds. A disk is *bad* if the above equation does not hold. In this case, we do not know anything about the decrease of x_i, but at least, it follows by the construction of the algorithm that $x_i' \leq x_i$.

Using this set of definitions, we can establish the following key lemma.

Lemma 5.5. *After Part I of Algorithm 3, the number of leaders in an arbitrary disk C of radius $1/2$ is in $\mathrm{O}(1)$ in expectation for some constant τ, i.e., $\mathrm{E}[S \cap C] \in \mathrm{O}(1)$.*

Proof. We distinguish two kinds of leaders. *Bad leaders* are those which during the course of the algorithm have been located in at least one bad disk. *Good leaders* have never been part of a bad disk in any round.

Consider an arbitrary round r_i. If r_i is *not good* for a disk $C_i \in C$, m_i may remain the same in the worst case. In subsequent rounds, such *bad* active nodes may disappear, but clearly, for our worst-case analysis, we obtain an upper bound for the total number of leaders in C if we assume that all such bad nodes will permanently remain leaders. Let the set of all bad nodes (potential bad leaders) in round r_i be U_i, and let W denote the set of good leaders.

In each round r_i, new random IDs are chosen by active nodes. As a consequence, the outcomes of the random experiments of consecutive rounds are independent of each other. Thus, we can bound $\mathrm{E}[S \cap C]$ as

$$\mathrm{E}[S \cap C] \;\leq\; \sum_{i=1}^{\log_\xi \log n} \mathrm{E}[U_i] + \mathrm{E}[W].$$

The expected number of bad nodes in round r_i in a disk $C_i \in C$ is at most m_i times the probability of a failure in

Lemma 5.2. Summing up over all disks $C_i \in C$ yields

$$\mathrm{E}[U_i] \;\le\; \sum_{C_i \in C} \gamma \frac{m_i}{m_i^{\ln m_i + 1 + \frac{2}{\log \xi}}} \;=\; \sum_{C_i \in C} \frac{\gamma}{m_i^{\ln m_i + \frac{2}{\log \xi}}}.$$

Now consider a phase P_j, and denote by $\beta_j := \sum_{r_i \in P_j} \mathrm{E}[U_i]$ the expected number of bad nodes during all rounds r_i of this phase. In phase P_j, all active disks have at least $m_i \ge \log^{(j)} n$ active nodes in neighboring disks. Because the term $\mathrm{E}[U_i]$ is maximized for small m_i, we have

$$\begin{aligned}
\beta_j &= \sum_{r_i \in P_j} \sum_{C_i \in C} \frac{\gamma}{m_i^{\ln m_i + \frac{2}{\log \xi}}} \\
&\le \sum_{r_i \in P_j} \sum_{C_i \in C} \frac{\gamma}{(\log^{(j)} n)^{\ln \log^{(j)} n + \frac{2}{\log \xi}}} \\
&\underset{\text{Lemma 5.3}}{<} \sum_{r_i \in P_j} \frac{\gamma \alpha(i)}{(\log^{(j)} n)^{\ln \log^{(j)} n + \frac{2}{\log \xi}}}.
\end{aligned}$$

We now bound $\alpha(i)$ (Lemma 5.3) in terms of $\log^{(j)} n$,

$$\begin{aligned}
\alpha(i) &\le \frac{\eta (\log n)^{\frac{2}{\log \xi}}}{2^{2i}} \\
&= \frac{\eta (\log n)^{\frac{2}{\log \xi}} \cdot (\log^{(j)} n)^{\frac{2}{\log \xi}}}{(\log n)^{\frac{2}{\log \xi}} \cdot 2^{2(i - \log_\xi \log n + \log^{(j+1)} n)}} \\
&< \eta (\log_2^{(j)} n)^{\frac{2}{\log \xi}} \;<\; \eta (\log^{(j)} n)^{\frac{2}{\log \xi}}.
\end{aligned}$$

The second inequality follows because by definition of phase P_j, it holds that $i \ge \log_\xi \log n + \log^{(j+1)} n$. Plugging this value in the expression for β_j, we get

$$\begin{aligned}
\beta_j &< \sum_{i \in P_j} \frac{\gamma \eta (\log^{(j)} n)^{\frac{2}{\log \xi}}}{(\log^{(j)} n)^{\ln \log^{(j)} n + \frac{2}{\log \xi}}} \\
&= \sum_{i \in P_j} \frac{\gamma \eta}{(\log^{(j)} n)^{\ln \log^{(j)} n}} \\
&< \frac{\gamma \eta \log^{(j+1)} n}{(\log^{(j)} n)^{\ln \log^{(j)} n}} \;<\; \frac{\gamma \eta}{\log^* n}.
\end{aligned}$$

for large enough n. Notice that the second inequality follows from the fact that phase P_j consists of $\log^{(j+1)} n - \log^{(j+2)} n < \log^{(j+1)} n$ rounds.

Having bounded the expected number of bad nodes, we now consider the number of good leaders $\mathrm{E}[W]$. Good leaders never occur in a bad disk and hence, their number decreases in each round as indicated in Lemma 5.2. In case a disk is inactive, its m_i is bounded by $m_i < \log^{(j)} n$ by definition. Let W_i be the set of good leaders that exist after round r_i. Because good leaders always successfully fulfil Lemma 5.2, we can bound $|W_{i+1}|$ as

$$|W_{i+1}| \;\le\; \alpha(i) \cdot \max\{\delta \sqrt{|W_i|} \ln |W_i|,\; \log^{(j)} n\},$$

where P_j is the phase containing round r_i. For a large enough constant τ, the function $\sqrt{x} \ln x$ is bounded by $\tau x^{2/3}$. Therefore,

$$|W_{i+1}| \;\le\; \alpha(i) \cdot \max\{\delta\tau |W_i|^{2/3},\; \log^{(j)} n\}. \qquad (2)$$

In the sequel, we use this recursion in order to bound $E[W]$. Ignoring the second argument of the maximum function for the moment and denoting $|W_0|$ by n_0, this recursion leads to the following upper bound for the size of $|W_{i+1}|$, which can be proven by induction,

$$\begin{aligned}
|W_{i+1}| &\le \alpha(i) \cdot (\delta\tau)^3 \cdot n_0^{(2/3)^i} \\
&\le \frac{\eta (\log n_0)^{\frac{2}{\log \xi}}}{2^{2i}} \cdot (\delta\tau)^3 \cdot n_0^{(2/3)^i},
\end{aligned}$$

where the second inequality follows from Lemma 5.3.

Therefore, the expected number of good dominators at the end of the algorithm is $E[W] \le |W_{\log_\xi \log n}|$, where $\xi = 3/2$. Plugging in $i = \log_\xi \log n$ yields

$$\begin{aligned}
E[W] &\le \frac{\eta (\log n_0)^{\frac{2}{\log \xi}}}{2^{2 \log_\xi \log n_0}} \cdot (\delta\tau)^3 \cdot n_0^{(2/3)^{\log_\xi \log n_0}} \\
&\le \frac{\eta (\log n_0)^{\frac{2}{\log \xi}}}{(\log n_0)^{\frac{2}{\log \xi}}} \cdot (\delta\tau)^3 \cdot n_0^{\frac{1}{\log n_0}} \\
&\le 2\eta(\delta\tau)^3 \;\in\; \mathrm{O}(1).
\end{aligned}$$

In general, we cannot simply ignore the second argument of the maximum function in (2). Assuming that P_j is a phase at the end of which $\log^{(j)} n > \delta k \sqrt{|W_i|} \ln |W_i|$, i.e., $|W_\ell| = \alpha(\ell) \log^{(j)} n$, where r_ℓ is the last round of phase P_j. By the definition of phase P_j, the number of rounds remaining after round r_ℓ until the end of the algorithm is $\log_\xi \log n - \ell = \log^{(j+2)} n$. That is, in this case we can compute $E[W]$ analogously by setting $n_0 = \log^{(j)} n$ and $E[W] \le |W_{\log^{(j+2)} n}|$. That is,

$$\begin{aligned}
E[W] &\le \frac{\eta (\log^{(j+1)} n)^{\frac{2}{\log \xi}}}{2^{2 \log^{(j+2)} n}} (\delta\tau)^3 (\log^{(j)} n)^{(2/3)^{\log^{(j+2)} n}} \\
&\le \eta(\delta\tau)^3 \left(\frac{\log_2^{(j)} n}{\log_2 \xi} \right)^{\frac{1}{\log_2^{(j+1)} n}} \;\in\; \mathrm{O}(1).
\end{aligned}$$

Finally, putting the results for both bad and good nodes together, we obtain

$$\begin{aligned}
\mathrm{E}[S \cap C] &\le \sum_{i=1}^{\log_\xi \log n} \mathrm{E}[U_i] + \mathrm{E}[W] \\
&= \sum_{j=1}^{\log^* n} \beta_j + \mathrm{E}[W] \;\in\; \mathrm{O}(1),
\end{aligned}$$

because in the sum $\sum_{j=1}^{\log^* n} \beta_j$, all but the last term (which is constant) are at most $1/\log^* n$ for large enough n and therefore $\sum_{j=1}^{\log^* n} \beta_j \in \mathrm{O}(1)$. □

This result on Part I leads to a bound on the expected number of leaders at the end of Algorithm 3.

Lemma 5.6. *After Part II of Algorithm 3, the expected number of leaders in an any disk C of radius $1/2$ is $\mathrm{O}(k)$.*

Proof. We first prove that as soon as the number of leaders in a disk C of radius $1/2$ exceeds k at the beginning of an iteration of the while loop of Part II of Algorithm 3, no more nodes in C are selected as leaders. Because the nodes in C form a clique—they are all within distance one of each other—, the 'coverage' $c(v)$ is lower-bounded by the number of leaders in C for all $v \in C$. We therefore have $c(v) \geq k$ for all $v \in C$ if there are at least k leaders in C. Because only nodes $u \in U(v)$ for some leader v are selected in Part II and because we have $c(u) < k$ for all $u \in U(v)$, nodes w for which $c(w) \geq k$ cannot become new leaders in Part II.

All leader nodes v which select new leaders in C have to be in a disk C' with radius $3/2$ around the center of C. Because C' can be covered with a constant number of disks of radius $1/2$ (cf. Figure 1), by Lemma 5.5 and by linearity of expectation, the expected number of leader nodes v selecting new leaders in C is constant. Because every such node selects at most k new leaders, the expected number of new leaders in C is $\mathrm{O}(k)$ in every iteration of the while loop. □

Using Lemma 5.6, we can now derive the main theorem.

Theorem 5.7. *Algorithm 3 runs in time $\mathrm{O}(\log\log n)$ and uses messages of size $\mathrm{O}(\log n)$ bits. In expectation, it computes an $\mathrm{O}(1)$ approximation to the minimum k-fold dominating set problem in UDGs.*

Proof. Running time and message size follow directly from the definition of Algorithm 3. The running time of Part II is constant because every leader v from Part I can select at most $\mathrm{O}(k)$ new leader nodes until all disks of radius $1/2$ at distance at most 1 from v contain at least k leader nodes. As for the approximation ratio, the number of dominators in a disk C is $\mathrm{O}(k)$ in expectation by Lemma 5.6. The theorem follows from the linearity of expectation and the fact that the optimal algorithm has to choose at least k dominators in a disk C' of radius $3/2$ in order to cover the nodes in C. □

References

[1] K. Alzoubi, P.-J. Wan, and O. Frieder. Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks. In *Proc. of the 3^{rd} MOBIHOC*, pages 157–164, 2002.

[2] B. Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32(4):804–823, 1985.

[3] D. J. Baker and A. Ephremides. The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm. *IEEE Transactions on Communications*, COM-29(11):1694–1701, 1981.

[4] M. Chatterjee, S. K. Das, and D. Turgut. WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks. *Journal of Cluster Computing*, 5(2):193–204, 2002.

[5] V. Chvátal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3), 1979.

[6] U. Feige. A Threshold of ln n for Approximating Set Cover. *Journal of the ACM*, 45(4):634–652, 1998.

[7] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete Mobile Centers. In *Proc. 17^{th} Symposium on Computational Geometry (SCG)*, pages 188–196, 2001.

[8] M. Gerla and J. Tsai. Multicluster, mobile, multimedia radio network. *ACM/Baltzer Journal of Wireless Networks*, 1(3):255–265, 1995.

[9] L. Jia, R. Rajaraman, and R. Suel. An efficient distributed algorithm for constructing small dominating sets. *Distributed Computing*, 15(4):193–205, 2002.

[10] R. Kershner. The number of circles covering a set. *American Journal of Mathematics*, 62, 1939.

[11] F. Kuhn. *The Price of Locality: Exploring the Complexity of Distributed Coordination Primitives*. PhD thesis, ETH Zurich, 2005.

[12] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing Newly Deployed Ad Hoc and Sensor Networks. In *Proceedings of the 10^{th} Int. Conf. on Mobile Computing and Networking (MOBICOM)*, 2004.

[13] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What Cannot be Computed Locally! In *Proc. of the 23^{rd} Symposium on the Principles of Distributed Computing (PODC)*, 2004.

[14] F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the Locality of Bounded Growth. In *Proc. of the 24^{rd} Symposium on the Principles of Distributed Computing (PODC)*, 2005.

[15] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The Price of Being Near-Sighted. In *Proc. of the 17^{th} ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[16] F. Kuhn and R. Wattenhofer. Constant-Time Distributed Dominating Set Approximation. In *Proc. of 22^{nd} Symposium on the Principles of Distributed Computing (PODC)*, pages 25–32, 2003.

[17] S. Kutten and D. Peleg. Fast Distributed Construction of Small k-Dominating Sets and Applications. *Journal of Algorithms*, 28:40–66, 1998.

[18] T. Moscibroda, P. von Rickenbach, and R. Wattenhofer. Analyzing the Energy-Latency Trade-off during the Deployment of Sensor Networks. In *Proc. 25^{th} INFOCOM*, 2006.

[19] L. D. Penso and V. C. Barbosa. A Distributed Algorithm to find k-Dominating Sets. *Discrete Applied Mathematics*, 141(1-3):243–253, 2004.

[20] S. Rajagopalan and V. Vazirani. Primal-Dual RNC Approximation Algorithms for Set Cover and Covering Integer Programs. *SIAM Journal on Computing*, 28:525–540, 1998.

[21] V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[22] P. Wan, K. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. In *Proceedings of INFOCOM*, 2002.

[23] J. Wu and H. Li. On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks. In *Proc. of DIALM*, pages 7–14, 1999.