Constant-Time Distributed Dominating Set Approximation*

Fabian Kuhn
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
kuhn@inf.ethz.ch

Roger Wattenhofer
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
wattenhofer@inf.ethz.ch

ABSTRACT

Finding a small dominating set is one of the most fundamental problems of traditional graph theory. In this paper, we present a new fully distributed approximation algorithm based on LP relaxation techniques. For an arbitrary parameter k and maximum degree Δ, our algorithm computes a dominating set of expected size $\mathrm{O}\big(k\Delta^{2/k}\log\Delta|DS_{\mathrm{OPT}}|\big)$ in $\mathrm{O}\big(k^2\big)$ rounds where each node has to send $\mathrm{O}\big(k^2\Delta\big)$ messages of size $\mathrm{O}(\log\Delta)$. This is the first algorithm which achieves a non-trivial approximation ratio in a constant number of rounds.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*;
G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*;
G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*

General Terms

Algorithms, Theory

Keywords

Dominating Sets, Approximation Algorithms, Distributed Algorithms, Linear Programming, Ad-Hoc Networks

1. INTRODUCTION

In a graph, a *dominating set* is a subset of nodes such that for every node v either a) v is in the dominating set or b) a direct neighbor of v is in the dominating set. The *minimum dominating set* (MDS) problem asks for a dominating set of minimum size. MDS and the closely related *minimum set cover* problem are two of the first problems that have been shown to be NP-hard [8, 12]. In this paper, we present a distributed approximation algorithm for MDS. In computer networks it is often desirable to have a dominating set in order to enable a hierarchical structure in which the members of the dominating set provide a service for their neighbors.

A particular application can be found in the fast growing field of *mobile ad-hoc networks*. In mobile ad-hoc networks, wireless devices (called nodes) communicate without stationary server infrastructure. When sending a message from one node to another, intermediate network nodes have to serve as routers. Although a number of interesting suggestions have been made, finding efficient algorithms for the routing process remains the most important problem for ad-hoc networks. Since the topology of an ad-hoc network is constantly changing, routing protocols for ad-hoc networks differ significantly from the standard routing schemes which are used in wired networks. One effective way to improve the performance of routing algorithms is by grouping nodes into clusters. The routing is then done between clusters. The most basic method for clustering is by calculating a dominating set. Only the nodes of the dominating set (the 'cluster heads') act as routers, all other nodes communicate via a neighbor in the dominating set.

Between traditional wired networks and mobile ad-hoc networks two main distinctions can be made: 1) typically wireless devices have much lower bandwidth than their wired counterparts and 2) wireless devices are mobile and therefore the topology of the network changes rather frequently. As a consequence, distributed algorithms which run on such devices should have as little communication as possible and they should run as fast as possible. Both goals can only be achieved by developing algorithms requiring a small number of communication rounds only (often called *local algorithms*). So far, the only algorithm which achieves a nontrivial approximation ratio—$\mathrm{o}(\Delta)$—in a nontrivial number of rounds—$\mathrm{o}(diam(G))$—for MDS was developed by Jia, Rajaraman, and Suel [10]. In expectation, their algorithm achieves an $\mathrm{O}(\log\Delta)$-approximation while the number of rounds is $\mathrm{O}(\log n\log\Delta)$ with high probability. In this paper, we present the first distributed MDS algorithm which achieves a nontrivial approximation ratio in a constant number of rounds. Precisely, for an arbitrary parameter k, in $\mathrm{O}\big(k^2\big)$ rounds, we achieve an expected approximation ratio of $\mathrm{O}\big(k\Delta^{2/k}\log\Delta\big)$. All messages are of size $\mathrm{O}(\log\Delta)$.

*The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

The paper is structured in the following way. Section 2 gives an overview over relevant previous work, Section 3 introduces some notation as well as some well-known facts, and in Sections 4 and 5 the dominating set algorithm is developed. Thereby Section 4 introduces the fractional dominating set problem (LP relaxation) and presents an algorithm to deduce a dominating set from a solution to the fractional variant of the problem, whereas Section 5 shows how to approximate the fractional dominating set problem by means of a distributed algorithm. The paper is concluded in Section 6.

2. RELATED WORK

The problem of finding small dominating sets in a graph and the closely related problem of finding small set covers has extensively been studied over the last 30 years. The problem of finding a minimum dominating set has been proven to be NP-hard in [8, 12]. The best known approximation is achieved by the greedy algorithm [11, 14, 18]. As long as there are uncovered nodes, the greedy algorithm picks a node which covers the biggest number of uncovered nodes and puts it into the dominating set. It achieves an approximation ratio of $\ln\Delta$ where Δ is the highest degree in the graph. Unless the problems of NP can be solved by deterministic $n^{\mathrm{O}(\log\log n)}$ algorithms, this is the best possible up to lower order terms [6]. For the related problem of finding small *connected* dominating sets, a similar approach is shown to be a $(\ln\Delta + \mathrm{O}(1))$-approximation in [9].

For the distributed construction of dominating sets, several algorithms have been developed. In [13] an algorithm which calculates a dominating set of size at most $n/2$ in $\mathrm{O}(\log^* n)$ rounds has been proposed. [19] presents a (connected) dominating set algorithm which runs in a constant number of rounds. None of those algorithms achieves a non-trivial asymptotic bound on the approximation ratio. Note that $\mathrm{O}(\Delta)$ is trivial since the set V of all nodes of G forms a dominating set of size at most $(\Delta+1)$ times the size of an optimal one. The first algorithm which achieves a non-trivial approximation ratio in less than $\Theta(diam(G))$ rounds was presented in [10]. The expected approximation ratio is asymptotically optimal—$\mathrm{O}(\log\Delta)$—and the algorithm terminates after $\mathrm{O}(\log n\log\Delta)$ rounds with high probability. The algorithm of [10] is related to the parallel set cover algorithms in [3, 16], which achieve $\mathrm{O}(\log\Delta)$ approximations in polylogarithmic time. For the connected dominating set problem, a distributed algorithm which also achieves an approximation ratio of $\mathrm{O}(\log\Delta)$ in a polylogarithmic number of rounds has been presented in [5], recently. In our algorithm, we first solve the LP relaxation—a positive linear program—of MDS. Parallel and distributed algorithms for positive linear programming have been studied in [15] and [2], respectively. In polylogarithmic time they both achieve a $(1+\epsilon)$-approximation for the linear program.

For ad-hoc networks, the (connected) dominating set problem has also been studied for special graphs. In particular for the unit disk graph a number of publications have been written (e.g. [1, 7]). For the unit disk graph the problem is known to remain NP-hard; however, constant factor approximations are possible in this case. For a recent survey on ad-hoc routing and related problems, we refer to [17].

3. NOTATION AND PRELIMINARIES

In this section we introduce notations as well as some mathematical theorems which are used in the paper.

The subject of this paper is the distributed construction of dominating sets of a network graph $G=(V,E)$. For convenience, we assume that $V=\{v_1,v_2,\ldots,v_n\}$, i.e. we assume that the network nodes are labeled from 1 to n. These labels are not used in our algorithms, but they simplify some proofs. By N_i, we denote the closed neighborhood of v_i, i.e. N_i includes v_i as well as all direct neighbors of v_i. Where appropriate, N_i also denotes the set of the indices of the nodes in N_i. The degree of a node v_i is called δ_i whereas Δ denotes the maximum degree in the network graph G. We will often make use of the maximum degree in a certain range around a node v_i. For this purpose we define $\delta_i^{(1)}$ and $\delta_i^{(2)}$:

$$\delta_i^{(1)} := \max_{j\in N_i}\delta_j, \quad \delta_i^{(2)} := \max_{j\in N_i}\delta_j^{(1)}.$$

Thus $\delta_i^{(1)}$ is the maximum degree of all nodes in the closed neighborhood N_i of v_i whereas $\delta_i^{(2)}$ is the maximum degree among all nodes at distance at most 2 from v_i.

For our algorithms, we use a purely synchronous model for communication. That is, in every communication round, each node is allowed to send a message to each of its direct neighbors in G. In principle, those messages can be of arbitrary size; however, our algorithms only use messages of size $\mathrm{O}(\log\Delta)$.

We conclude this section by giving two facts which will then be used in subsequent sections. Proofs are omitted and can be found in standard mathematical text books.

FACT 3.1. **(Means Inequality)** *Le $\mathcal{A}\subset\mathbb{R}^+$ be a set of positive real numbers. The product of the values in $\mathcal{A}$ can be upper bounded by replacing each factor with the arithmetic mean of the elements of $\mathcal{A}$:*

$$\prod_{x\in\mathcal{A}} x \;\le\; \left(\frac{\sum_{x\in\mathcal{A}} x}{|\mathcal{A}|}\right)^{|\mathcal{A}|}.$$

FACT 3.2. *For $n\ge x\ge 1$, we have*

$$\left(1-\frac{x}{n}\right)^n \;\le\; e^{-x}.$$

4. APPROXIMATING MDS BY LP RELAXATION

The Minimum Dominating Set (MDS) problem has been introduced in Section 1. In this section, we show how to obtain a $\ln\Delta$ approximation by using LP relaxation techniques. For an introduction to linear programming see e.g. [4]. We first derive the integer program which describes the MDS problem. Let $S\subseteq V$ denote a subset of the nodes of G. To each $v_i\in V$, we assign a bit x_i such that $x_i=1\Leftrightarrow v_i\in S$. For S to be a dominating set, we have to demand that for each node $v_i\in V$, at least one of the nodes in N_i is in S. Therefore, S is a dominating set of G if and only if $\forall i\in[1,n]:\sum_{j\in N_i}x_j\ge 1$. We define the neighborhood matrix N to be the sum of the adjacency matrix of G and the

identity matrix (N is the adjacency matrix with ones in the diagonal). The MDS problem can then be formulated as an integer program:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i \\ \text{subject to} \quad & N \cdot \underline{x} \geq \underline{1} \\ & \underline{x} \in \{0,1\}^n. \end{aligned} \tag{IP_{MDS}}$$

By relaxing the condition $\underline{x} \in \{0,1\}^n$ to $\underline{x} \geq \underline{0}$, we get the following linear program:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i \\ \text{subject to} \quad & N \cdot \underline{x} \geq \underline{1} \\ & \underline{x} \geq \underline{0}. \end{aligned} \tag{LP_{MDS}}$$

In the literature, the LP form of the dominating set problem has also been named fractional dominating set problem. The corresponding dual linear program looks very similar to LP_{MDS}:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} y_i \\ \text{subject to} \quad & N \cdot \underline{y} \leq \underline{1} \\ & \underline{y} \geq \underline{0}. \end{aligned} \tag{DLP_{MDS}}$$

We have to assign a positive value y_i to each node v_i. The sum of the y-values of the nodes in the neighborhood N_i of a node v_i has to be less than or equal to 1 (for the corresponding x-values, this sum has to be greater than or equal to 1) and the sum of all y-values, i.e. the objective function has to be maximized. As a consequence we get the following lower bound on the size of a minimum dominating set.

LEMMA 4.1. *Let $\delta_i^{(1)}$ be the maximum of the degrees of all nodes in N_i as defined in Section 3. For any dominating set DS (i.e. also for an optimal one), we have*

$$\sum_{i=1}^{n} \frac{1}{\delta_i^{(1)}+1} \leq |DS|.$$

PROOF. Assigning $y_i := 1/(\delta_i^{(1)}+1)$ yields a feasible solution to the dual linear program DLP_{MDS}. By the weak duality theorem, the value of the objective function for any feasible solution for DLP_{MDS} is smaller or equal to the value of the objective function for any feasible solution for LP_{MDS}. Hence, the objective function for the DLP_{MDS}-solution is also smaller or equal to the size of any dominating set because any feasible solution for the integer program IP_{MDS} is feasible for LP_{MDS} too. □

Let $\underline{x}^*$ be an optimal solution for LP_{MDS}. Further let $\underline{x}^{(\alpha)}$ be an α-approximation for LP_{MDS}, i.e. $\underline{x}^{(\alpha)}$ is a feasible solution for which

$$\sum_{i=1}^{n} x_i^{(\alpha)} \leq \alpha \cdot \sum_{i=1}^{n} x_i^*. \tag{1}$$

In order to get an approximate solution $\underline{x}_{\text{DS}}$ for IP_{MDS} from an α-approximation $\underline{x}^{(\alpha)}$ for LP_{MDS}, each node applies the distributed Algorithm 1.

Algorithm 1 $\text{LP}_{\text{MDS}} \longrightarrow \text{IP}_{\text{MDS}}$

Input: feasible solution $\underline{x}^{(\alpha)}$ for LP_{MDS}
Output: IP_{MDS}-solution $\underline{x}_{\text{DS}}$ (dom. set)

1: calculate $\delta_i^{(2)}$
2: $p_i := \min\{1,\ x_i^{(\alpha)} \cdot \ln(\delta_i^{(2)}+1)\}$
3: $x_{\text{DS},i} := \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{otherwise} \end{cases}$
4: **send** $x_{\text{DS},i}$ to all neighbors
5: **if** $x_{\text{DS},j} = 0$ for all $j \in N_i$ **then**
6: $\quad x_{\text{DS},i} := 1$
7: **fi**

Remark:
In line 2, $\delta_i^{(2)}$ is calculated as follows. In a first round, each node v_i sends its degree δ_j to all neighbors. Afterwards $\delta_i^{(1)}$ ($:= \max_{j \in N_i} \delta_k$) is sent to all neighbors in a second round. $\delta_i^{(2)}$ can then be computed as $\max_{j \in N_i} \delta_j^{(1)}$.

THEOREM 4.2. *Let DS_{OPT} be a minimum dominating set and let Δ be the greatest degree of the network graph G. $\underline{x}^{(\alpha)}$ is an α-approximation for LP_{MDS} and $\underline{x}_{\text{DS}}$ is the IP_{MDS}-solution calculated by Algorithm 1 with $\underline{x}^{(\alpha)}$ as its input. For the expected value of the size of the corresponding dominating set DS ($v_i \in DS \iff x_{\text{DS},i} = 1$), we have*

$$\text{E}\left[|DS|\right] \leq (1 + \alpha \ln(\Delta+1)) \cdot |DS_{\text{OPT}}|.$$

PROOF. A node can become a member of the dominating set in lines 3 and 6 of Algorithm 1. Let the random variables X and Y denote the numbers of nodes which are selected in lines 3 and 6, respectively. For the the expected value of X, we have

$$\begin{aligned} \text{E}\left[X\right] \quad &= \quad \sum_{i=1}^{n} p_i \ \leq \ \sum_{i=0}^{n} x_i^{(\alpha)} \cdot \ln(\delta_i^{(2)}+1) \\ &\underset{(\Delta \geq \delta_i^{(2)})}{\leq} \quad \ln(\Delta+1) \sum_{i=1}^{n} x_i^{(\alpha)} \\ &\underset{\text{Eqn. (1)}}{\leq} \quad \alpha \ln(\Delta+1) \sum_{i=0}^{n} x_i^* \\ &\leq \quad \alpha \ln(\Delta+1) \cdot |DS_{\text{OPT}}|. \end{aligned}$$

In order to compute the expected value of Y, we look at the probability q_i that no node in the direct neighborhood of node v_i (i.e. no node in N_i) has been selected. If $x_j^{(\alpha)} \cdot \ln(\delta_j^{(2)}) \geq 1$ for a $v_j \in N_i$, the corresponding $p_j = 1$ and therefore $q_i = 0$. Thus, we only have to consider the case

where all $p_j < 1$. We obtain

$$\begin{aligned} q_i &= \prod_{j\in N_i}(1-p_j) \le \prod_{j\in N_i}\Big(1-x_j^{(\alpha)}\ln(\delta_i^{(1)}+1)\Big) \\ &\le \left(1-\frac{\sum_{j\in N_i}x_j^{(\alpha)}\ln(\delta_i^{(1)}+1)}{\delta_i+1}\right)^{\delta_i+1} \\ &\le \left(1-\frac{\ln(\delta_i^{(1)}+1)}{\delta_i+1}\right)^{\delta_i+1} \le e^{-\ln(\delta_i^{(1)}+1)} \\ &= \frac{1}{\delta_i^{(1)}+1}. \end{aligned}$$

The first inequality follows from $\delta_i^{(1)} \le \delta_j^{(2)}$, the second inequality follows from Fact 3.1, the third inequality holds because $\underline{x}^{(\alpha)}$ is feasible and therefore the sum $\sum_{j\in N_i} x_j^{(\alpha)} \ge 1$, and the fourth inequality follows from Fact 3.2. For $\mathrm{E}[Y]$, we then have

$$\mathrm{E}[Y] = \sum_{i=1}^{n} q_i \le \sum_{i=1}^{n}\frac{1}{\delta_i^{(1)}+1} \le |DS_{\mathrm{OPT}}|.$$

The last inequality follows from Lemma 4.1. Adding $\mathrm{E}[X]$ and $\mathrm{E}[Y]$ concludes the proof. □

Remark 1:

In line 3 of Algorithm 1 we could multiply x_i with $\big(\ln(\delta_{v_i}^{(2)}+1)-\ln\ln(\delta_{v_i}^{(2)}+1)\big)$ instead of $\ln(\delta_{v_i}^{(2)}+1)$. We would then obtain $q_i \le \ln(\Delta+1)/(\delta_{v_i}^{(2)}+1)$ and the expected total size of the resulting dominating set would be less than or equal to $2\alpha\big(\ln(\Delta+1)-\ln\ln(\Delta+1)\big)|DS_{\mathrm{OPT}}|$.

Remark 2:

Note that for regular graphs, Algorithm 1 provides a very simple distributed algorithm to approximate MDS. Let the degree of each node of a regular graph be δ. Assigning $x_i := 1/(\delta+1)$ for all nodes v_i yields an optimal solution for $\mathrm{LP}_{\mathrm{MDS}}$. Applying Algorithm 1 then results in a $(1+\ln(\delta+1))$-approximation for the MDS problem.

In [6], Feige has proven that the dominating set problem cannot be approximated better than by an approximation ratio of $\ln\Delta$ unless $NP \in \mathrm{DTIME}(n^{\mathrm{O}(\log\log n)})$ (up to lower order terms). Hence, unless NP almost equals P, the above algorithm is optimal when applied to an optimal solution of the LP relaxation $\mathrm{LP}_{\mathrm{MDS}}$ of the dominating set problem. However, the strength of the approach of Algorithm 1 lies in the potential of distributing the calculation over the nodes of the network graph. When applied on a single computer, the greedy algorithm achieves the same approximation ratio in time $\mathrm{O}(n\Delta)$ [18] while computing the linear program $\mathrm{LP}_{\mathrm{MDS}}$ with an interior point method would take significantly longer. In the next section, we will show how to compute an approximation of the linear program $\mathrm{LP}_{\mathrm{MDS}}$ using a distributed algorithm.

5. APPROXIMATING THE LINEAR PROGRAM

In this section, we present the main algorithm of this paper. We show how to find a $\mathrm{O}\big(k\Delta^{2/k}\big)$-approximation of $\mathrm{LP}_{\mathrm{MDS}}$ in $\mathrm{O}\big(k^2\big)$ rounds. We will present the algorithm in two variants. For the sake of simplicity and clarity, we will first present an algorithm for the case that all nodes know the highest degree Δ in the network. In a second step, we will then generalize this algorithm such that the knowledge of Δ is not necessary any more.

During the algorithms, the nodes increase their x-values over time. In accordance with other dominating set papers (e.g. [9, 10]), we say that a node v_i is colored gray as soon as the sum of the weights x_j for $v_j \in N_i$ exceeds 1, i.e. as soon as the node is covered. Initially all nodes are colored white. The number of white nodes $v_j \in N_i$ at a given time is called the dynamic degree of v_i and denoted by $\tilde{\delta}(v_i)$. When starting the algorithms, all nodes are white, thus $\tilde{\delta}(v_i) = \delta_i + 1$.

Assume now that all nodes know Δ, the maximum degree of the network. Algorithm 2 is synchronously executed by all nodes ($a(v_i)$ and z_i are auxiliary variables which are explained later).

Algorithm 2 $\mathrm{LP}_{\mathrm{MDS}}$ approximation (Δ known)

```
1: x_i := 0;
2: for ℓ := k − 1 to 0 by −1 do
3:    (* δ̃(v_i) ≤ (Δ + 1)^{(ℓ+1)/k}, z_i := 0 *)
4:    for m := k − 1 to 0 by −1 do
5:       (* a(v_i) ≤ (Δ + 1)^{(m+1)/k} *)
6:       send color_i to all neighbors;
7:       δ̃(v_i) := |{j ∈ N_i | color_j = 'white'}|;
8:       if δ̃(v_i) ≥ (Δ + 1)^{ℓ/k} then
9:          x_i := max{x_i, 1/(Δ+1)^{m/k}}
10:      fi;
11:      send x_i to all neighbors;
12:      if Σ_{j∈N_i} x_j ≥ 1 then color_i := 'gray' fi;
13:   od
14:   (* z_i ≤ 1/(Δ + 1)^{(ℓ−1)/k} *)
15: od
```

Before coming to a detailed analysis of Algorithm 2, we give a general overview. During the algorithm, each node v_i calculates the corresponding component x_i of the solution for $\mathrm{LP}_{\mathrm{MDS}}$. Initially all x_i are set to 0, they are then gradually increased as the algorithm progresses. The algorithm consists of two nested loops. The purpose of the outer loop is to gradually reduce the highest dynamic degree in the network. As indicated by the invariant in line 3, $\tilde{\delta}(v_i)$ is reduced by a factor $(\Delta+1)^{1/k}$ in every iteration of the outer loop. In the inner loop, the x-values are increased stepwise. By this we can guarantee that the total weight is not too high.

Lemma 5.1 explains the invariant of line 3.

LEMMA 5.1. *At the beginning of each iteration of the outer loop of Algorithm 2, i.e. at line 3, the dynamic degree $\tilde{\delta}(v_i)$ of each node v_i is $\tilde{\delta}(v_i) \le (\Delta+1)^{(\ell+1)/k}$.*

PROOF. For $\ell = k-1$ the condition reduces to $\tilde{\delta}(v_i) \le \Delta+1$ and therefore follows from the definition of Δ. For all other iterations the lemma is true because in the very last step of the preceding iteration $(\ell+1)$, all nodes with

$\tilde{\delta}(v_i) \geq (\Delta+1)^{(\ell+1)/k}$ have set $x_i := 1$ in line 9. By this all nodes in N_i have turned gray and therefore $\tilde{\delta}(v_i)$ has become 0. Thus all degrees exceeding $(\Delta+1)^{(\ell+1)/k}$ have been set to 0, for all others the invariant already held beforehand. □

In a single iteration of the outer loop, only nodes with $\tilde{\delta}(v_i) \geq (\Delta+1)^{\ell/k}$ increase their x-value (lines 8-10). We call those nodes *active*. The number of active nodes in the closed neighborhood N_i of a white node v_i at the beginning of an inner-loop iteration (line 5) is called $a(v_i)$. We define $a(v_i) := 0$ if v_i is a gray node. The purpose of the inner loop is to gradually reduce the maximum $a(v)$ in the graph (invariant in line 5):

LEMMA 5.2. *At the beginning of each iteration of the inner loop of Algorithm 2, i.e. at line 5, $a(v_i) \leq (\Delta+1)^{(m+1)/k}$ for all nodes $v_i \in V$.*

PROOF. For $m = k-1$ we have $a(v_i) \leq (\Delta+1)$ which is always true. For the other cases, we prove that all nodes v_i with $a(v_i)$ too high have been covered in the previous iteration of the inner loop (i.e. they have become gray and therefore $a(v_i)$ has become 0). We show that all nodes v_i for which $a(v_i) > (\Delta+1)^{m/k}$ at line 5 are colored gray at the end of the inner-loop iteration (i.e. after line 14). All active nodes v_j increase x_j such that $x_j \geq 1/(\Delta+1)^{m/k}$ (lines 8-10 of Algorithm 2). If $a(v_i) > (\Delta+1)^{m/k}$ there are more than $(\Delta+1)^{m/k}$ active nodes in N_i. Therefore the sum of the x-values in N_i is greater or equal to 1 after line 10. □

In order to count the weights assigned during the iterations of the inner loop, we assign a variable z_i to each node v_i. In line 3 all z_i are set to 0. Whenever a node v_i increases x_i, the additional weight is equally distributed among the z_j of all the nodes v_j in N_i which are white before the increase of x_i. Hence the sum of the z-values is always equal to the sum of the x-increases during the current iteration of the outer loop. We can show that at the end of every iteration of the outer loop, i.e. at line 14, all $z_i \leq 1/(\Delta+1)^{(\ell-1)/k}$. Together with the invariant in line 3, this enables us to prove a bound on the total weight of the additional x-values in each iteration of the outer loop.

LEMMA 5.3. *At the end of an iteration of the outer loop of Algorithm 2, i.e. at line 14,*

$$z_i \leq \frac{1}{(\Delta+1)^{\frac{\ell-1}{k}}}$$

for all nodes $v_i \in V$.

PROOF. Because z_i is set to 0 in line 3, we only have to consider a single iteration of the outer loop, i.e. a period in which ℓ remains constant. z_i can only be increased as long as v_i is a white node. The increases all happen in line 9 because only there the x-values are increased. For each white node v_i, we divide the iteration of the outer loop into two phases. The first phase consists of all inner-loop iterations where v_i remains white. The second phase consist of the remaining inner-loop iterations where v_i becomes or is gray. During the whole first phase $\sum_{j\in N_i} x_j < 1$. Because all increases of x-values are distributed among at least $(\Delta+1)^{\ell/k}$ z-values we therefore get

$$z_i \;<\; \frac{\sum_{j\in N_i} x_j}{(\Delta+1)^{\frac{\ell}{k}}} \;\leq\; \frac{1}{(\Delta+1)^{\frac{\ell}{k}}} \tag{2}$$

for phase 1. In line 9 of the first inner-loop iteration of the second phase, z_i gets its final value because only z-values of white nodes are increased. All active nodes have already been active in the preceding inner-loop iteration because $\tilde{\delta}(v_j)$ can only become smaller over time. Thus from the preceding iteration, all active nodes $v_j \in N_i$ have $x_j \geq 1/(\Delta+1)^{(m+1)/k}$. In line 9 they are now increased to $1/(\Delta+1)^{m/k}$. The difference of this value is distributed among at least $(\Delta+1)^{\ell/k}$ z-values and because the number of active nodes in N_i is $a(v_i)$, the increase of z_i is at most

$$\frac{\frac{1}{(\Delta+1)^{\frac{m}{k}}} - \frac{1}{(\Delta+1)^{\frac{m+1}{k}}}}{(\Delta+1)^{\frac{\ell}{k}}} a(v_i). \tag{3}$$

To obtain a bound on z_i, we have to add its value before the increase which is given by Equation (2). From Lemma 5.2 we know that $a(v_i) \leq (\Delta+1)^{(m+1)/k}$. Plugging this into the sum of (2) and (3), we obtain

$$z_i \leq \frac{(\Delta+1)^{\frac{1}{k}} - 1}{(\Delta+1)^{\frac{\ell}{k}}} + \frac{1}{(\Delta+1)^{\frac{\ell}{k}}} = \frac{1}{(\Delta+1)^{\frac{\ell-1}{k}}},$$

which concludes the proof. □

We are now ready to consider the overall approximation ratio of Algorithm 2.

THEOREM 5.4. *For all network graphs G, Algorithm 2 computes a feasible solution x for the linear program $\mathrm{LP}_{\mathrm{MDS}}$ such that x is a $k(\Delta+1)^{2/k}$-approximation of $\mathrm{LP}_{\mathrm{MDS}}$. Further Algorithm 2 terminates after $2k^2$ rounds.*

PROOF. For the number of rounds, we see that each iteration of the inner loop involves the sending of two messages and therefore takes two rounds. The number of such iterations is k^2.

Further, the calculated x-values form a feasible solution of $\mathrm{LP}_{\mathrm{MDS}}$ because in the very last iteration of the inner loop ($\ell = 0$, $m = 0$) all nodes v_i with $\tilde{\delta}(v_i) \geq 1$ set $x_i := 1$. This includes all remaining white nodes. We prove the approximation ratio of $k(\Delta+1)^{2/k}$ by showing that the additional weight (i.e. sum of x-values) is upper-bounded by $(\Delta+1)^{2/k}$ in each iteration of the outer loop. From Lemma 5.1, we know that at line 3, i.e. when the iteration starts, the dynamic degree $\tilde{\delta}(v_i)$ of each node v_i is $\tilde{\delta}(v_i) \leq (\Delta+1)^{(\ell+1)/k}$. Hence there are at most $(\Delta+1)^{(\ell+1)/k}$ non-zero z-values in the closed neighborhood of every node v_i at the end of an outer-loop iteration at line 14. Further Lemma 5.3 implies that all z-values are less than or equal to $(\Delta+1)^{-(\ell-1)/k}$ at line 14. The sum of the z-values in the direct neighborhood of a node v_i during each iteration of the outer loop is

therefore upper-bounded by

$$\sum_{j\in N_i} z_j \leq \frac{(\Delta+1)^{\frac{\ell+1}{k}}}{(\Delta+1)^{\frac{\ell-1}{k}}} = (\Delta+1)^{\frac{2}{k}}.$$

If we assign $y_i := z_i/(\Delta+1)^{2/k}$, the y-values form a feasible solution for the dual LP $\mathrm{DLP}_{\mathrm{MDS}}$ because $\forall i : \sum_{j\in N_i} y_j \leq 1$. Hence the sum of all y-values is a lower bound on the size of DS_{OPT} and therefore $\sum_{i=1}^{n} z_i \leq (\Delta+1)^{2/k}|DS_{\mathrm{OPT}}|$ for every iteration of the outer loop. Because $\underline{z}$ is defined such that the sum over all z-values is equal to the sum over all increases of the x-values, and because there are k iterations of the outer loop, we have

$$\sum_{i=1}^{n} x_i \leq k(\Delta+1)^{\frac{2}{k}}|DS_{\mathrm{OPT}}|.$$

at the end of Algorithm 2. □

The only thing which cannot be calculated locally in Algorithm 2 is the maximum degree Δ. Algorithm 3 is an adaptation of Algorithm 2 where nodes do not need to know Δ. In each iteration, Algorithm 3 assigns an x_i which is greater or equal to the x_i assigned in the corresponding iteration of Algorithm 2. However, the x_i are chosen such that the approximation ratio of $k(\Delta+1)^{2/k}$ is preserved.

Algorithm 3 $\mathrm{LP}_{\mathrm{MDS}}$ approximation (Δ not known)

1: $x_i := 0$;
2: calculate $\delta_i^{(2)}$; (* 2 communication rounds *)
3: $\gamma^{(2)}(v_i) := \delta_i^{(2)} + 1$; $\tilde{\delta}(v_i) := \delta_i + 1$;
4: **for** $\ell := k-1$ to 0 by -1 **do**
5: (* $\tilde{\delta}(v_i) \leq (\Delta+1)^{(\ell+1)/k}$, $z_i := 0$ *)
6: **for** $m := k-1$ to 0 by -1 **do**
7: **if** $\tilde{\delta}(v_i) \geq \gamma^{(2)}(v_i)^{\frac{\ell}{\ell+1}}$ **then**
8: **send** 'active node' to all neighbors
9: **fi**;
10: $a(v_i) := |\{j \in N_i | v_j \text{ is 'active node'}\}|$;
11: **if** $\mathrm{color}_i =$ 'gray' **then** $a(v_i) := 0$ **fi**;
12: **send** $a(v_i)$ to all neighbors;
13: $a^{(1)}(v_i) := \max_{j\in N_i}\{a(v_j)\}$;
14: (* $a(v_i), a^{(1)}(v_i) \leq (\Delta+1)^{(m+1)/k}$ *)
15: **if** $\tilde{\delta}(v_i) \geq \gamma^{(2)}(v_i)^{\frac{\ell}{\ell+1}}$ **then**
16: $x_i := \max\left\{x_i, a^{(1)}(v_i)^{-\frac{m}{m+1}}\right\}$
17: **fi**;
18: **send** x_i to all neighbors;
19: **if** $\sum_{j\in N_i} x_j \geq 1$ **then** $\mathrm{color}_i :=$ 'gray' **fi**;
20: **send** color_i to all neighbors;
21: $\tilde{\delta}(v_i) := \left|\{j \in N_i \mid \mathrm{color}_j = \text{'white'}\}\right|$
22: **od**;
23: (* $z_i \leq (1+(\Delta+1)^{1/k})/\gamma^{(1)}(v_i)^{\ell/(\ell+1)}$ *)
24: **send** $\tilde{\delta}(v_i)$ to all neighbors;
25: $\gamma^{(1)}(v_i) := \max_{j\in N_i}\{\tilde{\delta}(v_j)\}$;
26: **send** $\gamma^{(1)}(v_i)$ to all neighbors;
27: $\gamma^{(2)}(v_i) := \max_{j\in N_i}\{\gamma^{(1)}(v_j)\}$
28: **od**

As for Algorithm 2, we first introduce some notation. $\gamma^{(d)}(v_i)$ denotes the maximum dynamic degree of all nodes with distance at most d from v_i at the beginning of the outer-loop iteration. We use the notation $\gamma^{(d)}(v_i)$ instead of $\tilde{\delta}^{(d)}(v_i)$ because $\gamma^{(d)}(v_i)$ remains constant during an iteration of the outer loop while $\tilde{\delta}(v_i)$ potentially changes after every iteration of the inner loop. In each inner-loop iteration, all nodes which assign a new x-value in line 16 of Algorithm 3 are called active. As before, $a(v_i)$ denotes the number of active nodes in the direct neighborhood N_i of a white node v_i; for gray nodes $a(v_i) := 0$. $a^{(1)}(v_i)$ is the maximum $a(v_j)$ among all $j \in N_i$. $\tilde{\delta}(v_i)$ and z_i are used as in the previous algorithm. We are now showing that Lemma 5.1 and Lemma 5.2 (cf. Lemma 5.5 and 5.6) also hold for Algorithm 3.

LEMMA 5.5. *At the beginning of each iteration of the outer loop of Algorithm 3, i.e. at line 5, the dynamic degree $\tilde{\delta}(v_i)$ of each node v_i is $\tilde{\delta}(v_i) \leq (\Delta+1)^{(\ell+1)/k}$.*

PROOF. We use induction to prove the lemma. Analogously to Lemma 5.1, for the first iteration ($\ell = k-1$), the lemma follows from the definition of Δ. To prove the lemma for subsequent iterations (iteration step), we show that as for Algorithm 2, all nodes with $\tilde{\delta}(v_i) \geq (\Delta+1)^{\ell/k}$ set $x_i := 1$ in the last iteration ($m = 0$) of the inner loop. According to lines 15-17 of the algorithm, we see that all nodes with $\tilde{\delta}(v_i) \geq \gamma^{(2)}(v_i)^{\ell/(\ell+1)}$ set $x_i := 1$ for $m = 0$. Hence we have to show that $\forall i : \gamma^{(2)}(v_i)^{\ell/(\ell+1)} \leq (\Delta+1)^{\ell/k}$. By the induction hypothesis, we know that $\forall i : \tilde{\delta}(v_i) \leq (\Delta+1)^{(\ell+1)/k}$ at the beginning of the outer-loop iteration. Because $\gamma^{(2)}(v_i)$ represents $\tilde{\delta}(v_j)$ of some node v_j in the two-hop neighborhood of v_i, we also have $\forall i : \gamma^{(2)}(v_i) \leq (\Delta+1)^{(\ell+1)/k}$ and therefore

$$\gamma^{(2)}(v_i)^{\frac{\ell}{\ell+1}} \leq (\Delta+1)^{\frac{\ell+1}{k}\cdot\frac{\ell}{\ell+1}} = (\Delta+1)^{\frac{\ell}{k}}.$$

□

LEMMA 5.6. *Before assigning a new value x_i to v_i in lines 15-17 of Algorithm 3, $a(v_i) \leq (\Delta+1)^{(m+1)/k}$ for all nodes $v_i \in V$.*

PROOF. As for Lemma 5.2, we prove that all nodes v_i for which $a(v_i) > (\Delta+1)^{m/k}$ at line 14 are colored gray at the end of the inner-loop iteration (i.e. after line 21). We use induction over the iterations of the inner loop. By the definition of Δ for every first iteration of the inner loop ($a(v_i) \leq \Delta+1$) and by the induction hypothesis for all other iterations, we have $\forall i : a(v_i) \leq (\Delta+1)^{(m+1)/k}$ at line 14. Therefore the weight each active node v_j assigns in line 16 is

$$x_j \geq \frac{1}{a^{(1)}(v_j)^{\frac{m}{m+1}}} \geq \frac{1}{(\Delta+1)^{\frac{m+1}{k}\cdot\frac{m}{m+1}}} = \frac{1}{(\Delta+1)^{\frac{m}{k}}}.$$

Because nodes v_i with $a(v_i) \geq (\Delta+1)^{m/k}$ have at least $(\Delta+1)^{m/k}$ active nodes in the direct neighborhood, they are covered after each of their $a(v_i)$ neighbor nodes v_j assigns a weight $x_j \geq 1/(\Delta+1)^{m/k}$. □

Lemma 5.7 is the analogue to Lemma 5.3.

LEMMA 5.7. *At line 23 of Algorithm 3,*

$$z_i \leq \frac{1+(\Delta+1)^{\frac{1}{k}}}{\gamma^{(1)}(v_i)^{\frac{\ell}{\ell+1}}} \tag{4}$$

for all nodes $v_i \in V$.

PROOF. As in Algorithm 2, z_i is set to 0 at line 5. Therefore, we only have to consider a single iteration of the outer loop. Again we consider two phases. In the iterations of the first phase v_i remains white, the second phase consists of the iterations where v_i becomes or is gray. While the algorithm is in the first phase $\sum_{j\in N_i} x_j < 1$. Further, all increases of values x_j are distributed among at least $\gamma^{(2)}(v_j)^{\ell/(\ell+1)} \geq \gamma^{(1)}(v_i)^{\ell/(\ell+1)}$ z-values. Therefore, in analogy to (2), we have

$$z_i \;\leq\; \sum_{j\in N_i} \frac{x_j}{\gamma^{(2)}(v_j)^{\frac{\ell}{\ell+1}}} < \frac{1}{\gamma^{(1)}(v_i)^{\frac{\ell}{\ell+1}}} \tag{5}$$

for phase 1. In line 16 of the first inner-loop iteration of the second phase, z_i is changed for the last time because only z-values of white nodes are increased. There each active neighbor x_j contributes at most

$$\frac{1}{a^{(1)}(v_j)^{\frac{m}{m+1}}} \cdot \frac{1}{\gamma^{(1)}(v_i)^{\frac{\ell}{\ell+1}}}$$

to the values z_i. Because $a(v_i) \leq a^{(1)}(v_j)$ and because v_i has $a(v_i)$ active nodes in the closed neighborhood N_i the total increase of z_i is at most

$$\frac{1}{a(v_i)^{\frac{m}{m+1}}} \cdot \frac{1}{\gamma^{(1)}(v_i)^{\frac{\ell}{\ell+1}}} \cdot a(v_i) = \frac{a(v_i)^{\frac{1}{m+1}}}{\gamma^{(1)}(v_i)^{\frac{\ell}{\ell+1}}}. \tag{6}$$

By Lemma 5.6, we have $a(v_i) \leq (\Delta+1)^{(m+1)/k}$ during an iteration of the inner loop. Plugging this into (6) and adding the value of z_i from the preceding iterations (5) concludes the proof:

$$z_i \leq \frac{\left((\Delta+1)^{\frac{m+1}{k}}\right)^{\frac{1}{m+1}}+1}{\gamma^{(1)}(v_i)^{\frac{\ell}{\ell+1}}} = \frac{(\Delta+1)^{\frac{1}{k}}+1}{\gamma^{(1)}(v_i)^{\frac{\ell}{\ell+1}}}.$$

□

THEOREM 5.8. *For all network graphs G, Algorithm 3 computes a feasible solution $\underline{x}$ with approximation ratio*

$$k\left((\Delta+1)^{1/k}+(\Delta+1)^{2/k}\right)$$

for the linear program $\mathrm{LP_{MDS}}$ *. Further Algorithm 3 terminates after $4k^2+\mathrm{O}(k)$ rounds.*

PROOF. The running time (i.e. number of rounds) can be determined as for Algorithm 2. In each iteration of the inner loop, 4 messages have to be sent. This yields $4k^2$ rounds for the totally k^2 inner-loop iterations. There is a constant number of additional rounds in each outer-loop iteration as well as at the beginning of the algorithm. Together, we get the claimed $4k^2+\mathrm{O}(k)$ rounds.

Analogously to Algorithm 2 $\underline{x}$ is feasible because in the very last iteration of the inner loop ($\ell = 0$, $m = 0$), all white nodes v_i set $x_i := 1$.

As for the other algorithm, we analyze each outer-loop iteration separately to determine the approximation ratio of Algorithm 3. By the definition of $\underline{z}$, the sum of the x-values of an outer-loop iteration is equal to the sum of the corresponding z-values. By Lemma 5.7 the sum of the z-values in the closed neighborhood of a node v_i in a single iteration of the outer loop is

$$\sum_{j\in N_i} z_j \leq \frac{1+(\Delta+1)^{\frac{1}{k}}}{\gamma^{(1)}(v_i)^{\frac{\ell}{\ell+1}}} \cdot \tilde{\delta}(v_i). \tag{7}$$

Because $\gamma^{(1)}(v_i)$ is the maximum dynamic degree in N_i, $\tilde{\delta}(v_i) \leq \gamma^{(1)}(v_i)$. Equation (7) can thus be formulated as

$$\sum_{j\in N_i} z_j \leq \left(1+(\Delta+1)^{\frac{1}{k}}\right)\gamma^{(1)}(v_i)^{\frac{1}{\ell+1}}. \tag{8}$$

By Lemma 5.5 we know that $\gamma^{(1)}(v_i) \leq (\Delta+1)^{(\ell+1)/k}$ and therefore

$$\gamma^{(1)}(v_i)^{\frac{1}{\ell+1}} \leq (\Delta+1)^{1/k}.$$

Plugging this into Equation (8) yields

$$\sum_{j\in N_i} z_j \leq (\Delta+1)^{1/k}+(\Delta+1)^{2/k}.$$

By dividing all z_i by the right hand side of the above inequality, we obtain a feasible solution for $\mathrm{DLP_{MDS}}$:

$$y_i := \frac{z_i}{(\Delta+1)^{\frac{1}{k}}+(\Delta+1)^{\frac{2}{k}}} \implies \sum_{j\in N_i} y_i \leq 1.$$

The sum of the z-values of an outer-loop iteration is therefore at most by a factor $(\Delta+1)^{1/k}+(\Delta+1)^{2/k}$ larger than the size of an optimal dominating set. At the end of the algorithm the sum over all x_i (objective function of $\mathrm{LP_{MDS}}$) is equal to the sum over the sums of the z_i for each outer loop iteration. Therefore

$$\sum_{i=1}^{n} x_i \leq k\left((\Delta+1)^{1/k}+(\Delta+1)^{2/k}\right)\cdot|DS_{\mathrm{OPT}}|.$$

□

Combining Algorithms 3 and 1 we obtain a distributed dominating set algorithm.

THEOREM 5.9. *Applying Algorithm 3 to obtain a* $\mathrm{LP_{MDS}}$*-approximation and Algorithm 1 to convert this approximation into a dominating set yields a distributed algorithm for the minimum dominating problem which achieves an approximation ratio of* $\mathrm{O}\left(k\Delta^{2/k}\log\Delta\right)$ *in* $\mathrm{O}\left(k^2\right)$ *rounds.*

PROOF. Theorem 5.9 directly follows from Theorems 4.2 and 5.8. □

Remark:

By setting $k = \Theta(\log\Delta)$, we obtain an algorithm which computes a $\mathrm{O}\left(\log^2\Delta\right)$ approximation for MDS in $\mathrm{O}\left(\log^2\Delta\right)$ rounds.

6. CONCLUSION

In this paper, we presented a distributed approximation algorithm for the minimum dominating set problem. By computing an $O(k\Delta^{2/k}\log\Delta)$-approximation in $O(k^2)$ rounds it is the first algorithm which achieves a non-trivial approximation ratio in a constant number of rounds. Particularly in the context of mobile ad-hoc networks but also in more general network settings, we believe that it is often advantageous to deploy algorithms which are very fast even when the calculated solution is not as good as the solution of a less local algorithm.

7. ACKNOWLEDGMENTS

We would like to thank Maurice Cochand, Juraj Hromkovič, David Peleg, Peter Widmayer, and Aaron Zollinger for fruitful discussions about the subject.

8. REFERENCES

[1] K. Alzoubi, P.-J. Wan, and O. Frieder. Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks. In *Proc. of the 3rd ACM Int. Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, pages 157–164, EPFL Lausanne, Switzerland, 2002.

[2] Y. Bartal, J. W. Byers, and D. Raz. Global Optimization Using Local Information with Applications to Flow Control. In *Proc. of the 38th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 303–312, 1997.

[3] B. Berger, J. Rompel, and P. Shor. Efficient NC Algorithms for Set Cover with Applications to Learning and Geometry. *Journal of Computer and System Sciences*, 49:454–477, 1994.

[4] V. Chvátal. *Linear Programming*. W. H. Freeman and Company, 1983.

[5] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast Distributed Algorithms for (Weakly) Connected Dominating Sets and Linear-Size Skeletons. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 717–724, 2003.

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

[7] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete Mobile Centers. In *Proc. of the 17th annual symposium on Computational geometry (SCG)*, pages 188–196. ACM Press, 2001.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[9] S. Guha and S. Khuller. Approximation Algorithms for Connected Dominating Sets. In *Proc. of the 4th Annual European Symposium on Algorithms (ESA)*, volume 1136 of *Lecture Notes in Computer Science*, pages 179–193, 1996.

[10] L. Jia, R. Rajaraman, and R. Suel. An Efficient Distributed Algorithm for Constructing Small Dominating Sets. In *Proc. of the 20th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 33–42, 2001.

[11] D. S. Johnson. Approximation Algorithms for Combinatorial Problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[12] R. M. Karp. Reducibility Among Combinatorial Problems. In *Proc. of a Symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

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

[14] L. Lovasz. On the Ratio of Optimal Integral and Fractional Covers. *Discrete Mathematics*, 13:383–390, 1975.

[15] M. Luby and N. Nisan. A Parallel Approximation Algorithm for Positive Linear Programming. In *Proc. of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 448–457, 1993.

[16] 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.

[17] R. Rajaraman. Topology Control and Routing in Ad hoc Networks: A Survey. *SIGACT News*, 33:60–73, June 2002.

[18] P. Slavík. A Tight Analysis of the Greedy Algorithm for Set Cover. In *Proc. of the 28th ACM Symposium on Theory of Computing (STOC)*, pages 435–441, 1996.

[19] J. Wu and H. Li. On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks. In *Proc. of the 3rd Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM)*, pages 7–14, 1999.