Distrib. Comput. (2005) 17: 303–310
Digital Object Identifier (DOI) 10.1007/s00446-004-0112-5

DISTRIBUTED COMPUTING

Constant-time distributed dominating set approximation*

Fabian Kuhn, Roger Wattenhofer

Computer Engineering and Networks Laboratory, ETH Zürich, 8092 Zürich, Switzerland (e-mail: kuhn@tik.ee.ethz.ch)

Received: September 9, 2003 / Accepted: September 2, 2004
Published online: January 13, 2005 –

Abstract. Finding a small dominating set is one of the most fundamental problems of classical graph theory. In this paper, we present a new fully distributed approximation algorithm based on LP relaxation techniques. For an arbitrary, possibly constant parameter k and maximum node 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. 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.

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 [9,13]. 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.

Two main distinctions can be made between traditional wired networks and mobile ad-hoc networks: First, typically wireless devices have much lower bandwidth than their wired counterparts and second, wireless devices are mobile and thus 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 (often called *local algorithms*). Because of the restrictive conditions imposed by the properties of wireless networks, it is often beneficial to keep the number of rounds small even at the cost of a somewhat non-optimal solution. So far, for MDS the only algorithm which achieves a nontrivial approximation ratio – $\mathrm{o}(\Delta)$ – in a nontrivial number of rounds – $\mathrm{o}(diam(G))$ – is that developed by Jia, Rajaraman, and Suel [11]. 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, we achieve an expected approximation ratio of $\mathrm{O}\big(k\Delta^{2/k}\log\Delta\big)$ in $\mathrm{O}\big(k^2\big)$ rounds. All messages are of size $\mathrm{O}(\log\Delta)$. The established trade-off between time complexity and approximation quality is especially interesting in light of a recent lower bound showing that in k rounds, minimum dominating set cannot be approximated better than $\Omega(\Delta^{1/k}/k)$ [14].

The paper is structured in the following way. Section 2 gives an overview over relevant previous work, Sect. 3 introduces some notation as well as some well-known facts, and the dominating set algorithm is developed in Sects. 4 and 5. Thereby Sect. 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 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 problem, whereas Sect. 5 shows how to approximate the fractional dominating set problem by means of a distributed algorithm. The paper is concluded in Sect. 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 been studied extensively over the last 30 years. The problem of finding a minimum dominating set has been proven to be NP-hard in [9,13]. The best known approximation is achieved by the greedy algorithm [4,12,16,21]. As long as there are uncovered nodes, the greedy algorithm picks a node which covers the largest 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 [7]. 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 [10].

For the distributed construction of dominating sets, several algorithms have been developed. An algorithm which calculates a dominating set of size at most $n/2$ in $\mathrm{O}(\log^* n)$ rounds has been proposed in [15]. [22] 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 [11]. 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 [11] is related to the parallel set cover algorithms in [3,19], 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 recently presented in [6]. 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 [17] 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, a number of papers studied MDS for the unit disk graph (e.g. [1,8]). 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 [20].

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 j of the nodes v_j 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 $\delta_i^{(2)}$ as follows:

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

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.

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 1. (Means Inequality) *Let $\mathcal{A} \subset \mathbb{R}^+$ be a set of positive real numbers. Then,*

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

Fact 2. *For $n \geq x \geq 1$, we have*

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

4 Approximating MDS by LP relaxation

In this section, we show how to obtain a $\ln \Delta$ approximation for MDS by using LP relaxation techniques. For an introduction to linear programming see e.g. [5]. 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 $\forall i \in [1, n] : \sum_{j \in N_i} x_j \geq 1$. 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:

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

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

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

The corresponding dual linear program looks very similar to $\mathrm{LP}_{\mathrm{MDS}}$:

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

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 at most 1 (for the corresponding x-values, this sum has to be at least 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 1. *Let $\delta_i^{(1)}$ be the maximum of the degrees of all nodes in N_i as defined in Sect. 3. For any dominating set DS (including 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 $\mathrm{DLP}_{\mathrm{MDS}}$. By the weak duality theorem, the value of the objective function for any feasible solution for $\mathrm{DLP}_{\mathrm{MDS}}$ is smaller than or equal to the value of the objective function for any feasible solution for $\mathrm{LP}_{\mathrm{MDS}}$. Hence, the objective function for the $\mathrm{DLP}_{\mathrm{MDS}}$-solution is also smaller or equal to the size of any dominating set because any feasible solution for the integer program $\mathrm{IP}_{\mathrm{MDS}}$ is feasible for $\mathrm{LP}_{\mathrm{MDS}}$ too. □

Let $\underline{x}^*$ be an optimal solution for $\mathrm{LP}_{\mathrm{MDS}}$. Let $\underline{x}^{(\alpha)}$ be an α-approximation for $\mathrm{LP}_{\mathrm{MDS}}$, 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 $\underline{x}_{\mathrm{DS}}$ for $\mathrm{IP}_{\mathrm{MDS}}$ from an α-approximation $\underline{x}^{(\alpha)}$ for $\mathrm{LP}_{\mathrm{MDS}}$, each node applies the distributed Algorithm 1. The algorithm uses a standard randomized rounding scheme [18], applied in a distributed scenario. For each node v_i, there is a variable $x_{\mathrm{DS},i}$ which is set to 1 if and only if v_i joins the dominating set.

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 $\max_{j\in N_i} \delta_j^{(1)}$.

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

Input: feasible solution $\underline{x}^{(\alpha)}$ for $\mathrm{LP}_{\mathrm{MDS}}$
Output: $\mathrm{IP}_{\mathrm{MDS}}$-solution $\underline{x}_{\mathrm{DS}}$ (dom. set)
1: calculate $\delta_i^{(2)}$
2: $p_i := \min\{1,\ x_i^{(\alpha)} \cdot \ln(\delta_i^{(2)}+1)\}$
3: $x_{\mathrm{DS},i} := \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{otherwise} \end{cases}$
4: **send** $x_{\mathrm{DS},i}$ to all neighbors
5: **if** $x_{\mathrm{DS},j} = 0$ for all $j \in N_i$ **then**
6: $\quad x_{\mathrm{DS},i} := 1$
7: **fi**

Theorem 3. *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 $\mathrm{LP}_{\mathrm{MDS}}$ and $\underline{x}_{\mathrm{DS}}$ is the $\mathrm{IP}_{\mathrm{MDS}}$-solution calculated by Algorithm 1 with $\underline{x}^{(\alpha)}$ as its input. For the expected value of the resulting dominating set, we have*

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

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 expected value of X, we have

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

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 $x_j^{(\alpha)} \ln(\delta_j^{(2)}) \geq 1$ for a $v_j \in N_i$, $p_j = 1$ and therefore $q_i = 0$. Therefore, we only have to consider the case where all $p_j < 1$. We obtain

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

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

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

The last inequality follows from Lemma 1. Adding $\mathrm{E}[X]$ and $\mathrm{E}[Y]$ concludes the proof. □

Remark: 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 $q_i \leq \ln(\Delta+1)/(\delta_{v_i}^{(2)}+1)$ and the expected total size of the resulting dominating set would be at most $2\alpha\big(\ln(\Delta+1)-\ln\ln(\Delta+1)\big)|DS_{\mathrm{OPT}}|$.

In [7], Feige has proven that up to lower order terms, the dominating set problem cannot be approximated better than $\ln\Delta$ unless $NP \in \mathrm{DTIME}(n^{\mathrm{O}(\log\log n)})$. Hence, unless problems in NP can be solved in quasi-polynomial time, 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 in time $\mathrm{O}(n\Delta)$ [21] 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 $\mathrm{LP}_{\mathrm{MDS}}$

In this section, we present the main algorithm of this paper. We show how to find a $\mathrm{O}(k\Delta^{2/k})$-approximation of $\mathrm{LP}_{\mathrm{MDS}}$ in $\mathrm{O}(k^2)$ 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 generalize this algorithm such that the knowledge of Δ is not necessary any more.

During the algorithms executions, the nodes increase their x-values over time. In accordance with other dominating set papers (e.g. [10,11]), 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.

Before coming to a detailed analysis of Algorithm 2, we give a general overview. The algorithm is closely related to the classic greedy dominating set algorithm [4,12,16,21]. 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. Compared to the sequential greedy algorithm where in each step exactly one node increases its x-value from 0 to 1, Algorithm 2 raises the x-values of many nodes simultaneously. In order to avoid the problem of overloading a node which has many neighbors increasing their x_i, we only increase the x-values by small amounts each time. Initially all x_i are set to 0, they are gradually increased as the algorithm progresses.

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

```
1: x_i := 0; δ̃(v_i) := δ_i + 1;
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:       if δ̃(v_i) ≥ (Δ + 1)^{ℓ/k} then
7:          x_i := max{x_i, 1/(Δ+1)^{m/k}}
8:       fi;
9:       send color_i to all neighbors;
10:      δ̃(v_i) := |{j ∈ N_i | color_j = 'white'}|;
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
```

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. Each iteration of the outer loop yields a primal infeasible solution. By rearranging the weights in a similar way as in the original greedy set cover proof, this primal solution can be converted into a dual solution $\underline{z} = (z_1, \ldots, z_n)$ which is feasible up to a factor $(\Delta+1)^{2/k}$. The combined primal solutions of all outer loop iterations give a primal feasible $k(\Delta+1)^{2/k}$-approximation. As in the sequential greedy algorithm, only the nodes of large degrees increase their x-values. In the inner loop, the x-values are increased stepwise. We call the high-degree nodes which increase their x-values, active nodes. The increase of x_i of a node v_i is at most indirectly proportional to the number of active neighbors of the white neighbors of v_i. By this, the number of active neighbors $a(v_i)$ of v_i is reduced in every iteration of the inner loop (invariant in line 5) and we can guarantee that the total x-increase is not too high. That is, no nodes is overloaded because of too many high-degree neighbors increasing their x_i by too much.

Lemma 2 explains the invariant of line 3.

Lemma 2. *At the beginning of each iteration ℓ 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*

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

Proof. For $\ell = k-1$ the condition reduces to $\tilde{\delta}(v_i) \leq \Delta+1$ 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 7. 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 6-8). 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

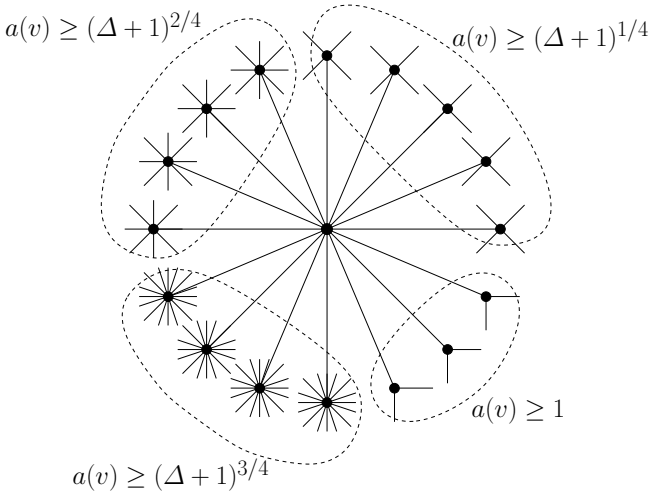

Fig. 1. Example with $k = 4$: First, the nodes which have $a(v) \geq (\Delta+1)^{3/4}$ active neighbors are covered when the x-values are set to $1/(\Delta+1)^{3/4}$, then the nodes which have $a(v) \geq (\Delta+1)^{2/4}$ active neighbors are covered when the x-values are set to $1/(\Delta+1)^{2/4}$, and so on. By this, it is guaranteed that the dual weights do not become too high

$a(v_i) := 0$ 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 3. *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 $m \neq k-1$, we prove that all nodes v_i with more than $(\Delta+1)^{(m+1)/k}$ active neighbors are gray and therefore $a(v_i) = 0$. It is sufficient to 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 6-8 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. Figure 1 serves as an illustration for Lemma 3. □

In order to bound 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. In Lemma 4, we show that at the end of every iteration of the outer loop, i.e., at line 14, all z_i are bounded. Together with the invariant in line 3 (Lemma 2), this enables us to prove a bound on the total weight of the additional x-values in each iteration of the outer loop.

Lemma 4. *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 (ℓ-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 7 because the x-values are increased only there. 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 7 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 $a(v_i)$ active nodes $v_j \in N_i$ have $x_j \geq 1/(\Delta+1)^{(m+1)/k}$. In line 7, the x-values of these nodes 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 so 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 Eq. (2). From Lemma 3 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 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 2, 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 4 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 $\mathrm{DLP_{MDS}}$ 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 an optimal dominating set 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. □

Remark: Algorithm 2 can easily be extended to solve the weighted fractional dominating set problem. For simplicity, assume that all nodes v_i have weights c_i between 1 and $c_{\max}$. Let $\tilde{\gamma}(v_i) := c_{\max}/c_i \tilde{\delta}(v_i)$ and let a node be active if $\tilde{\gamma}(v_i) \geq [c_{\max}(\Delta+1)]^{\ell/k}$. If we change lines 6 and 10 of Algorithm 2 in the appropriate way, we obtain an algorithm which approximates the weighted fractional dominating set. The new approximation ratio is $k(\Delta+1)^{1/k}[c_{\max}(\Delta+1)]^{1/k}$.

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.

As for Algorithm 2, we first introduce some notation. By $\gamma^{(d)}(v_i)$, we denote 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 (ℓ-loop) while $\tilde{\delta}(v_i)$ potentially changes after every

Algorithm 3 $\mathrm{LP_{MDS}}$ 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**

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 now show that Lemma 2 and Lemma 3 (cf. Lemma 5 and 6) also hold for Algorithm 3.

Lemma 5. *At the beginning of each iteration ℓ 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 2, 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 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 3 (see also Fig. 1), 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 7 is the analogue to Lemma 4.

Lemma 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 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:

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

Theorem 5. *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 $\mathrm{LP}_{\mathrm{MDS}}$. 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 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 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 Eq. (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 $\mathrm{DLP}_{\mathrm{MDS}}$:

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

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

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

where $|DS_{\mathrm{OPT}}|$ denotes the size of an optimal dominating set. □

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

Theorem 6. *Applying Algorithm 3 to obtain an approximation for* $\mathrm{LP}_{\mathrm{MDS}}$ *and Algorithm 1 to convert this approximation into a dominating set yields a dominating set whose expected size is within* $\mathrm{O}\big(k\Delta^{2/k}\log\Delta\big)$ *of the size of an optimal dominating set in* $\mathrm{O}\big(k^2\big)$ *rounds.*

Proof. Theorem 6 directly follows from Theorems 3 and 5. □

Remark: By setting $k = \Theta(\log\Delta)$, we achieve a dominating set algorithm which computes a $\mathrm{O}\big(\log^2\Delta\big)$ approximation in $\mathrm{O}\big(\log^2\Delta\big)$ rounds.

6 Conclusions

In this paper, we presented a distributed approximation algorithm for the minimum dominating set problem. The algorithm can be seen as a distributed implementation of the greedy dominating set algorithm. In order to parallelize the computation of the greedy algorithm, different nodes must be able to become dominator simultaneously. We avoid the problem that too many nearby nodes join the dominating set at the same time by first solving the fractional version of the problem and then applying randomized rounding. The increase of the weight of a node v is at most indirectly proportional to the number of active neighbors of the white neighbors of v. Jia et. al. [11] present another distributed variant of the greedy dominating set algorithm. Active nodes are chosen in a similar way as in our algorithm. The algorithm of [11] directly computes an integer solution for the problem. Symmetries are broken by randomly choosing a set of dominators among the active nodes. The probability for an active node v to become dominator is indirectly proportional to the median number of active neighbors of the white neighbors. We believe that LP relaxation is a promising technique for the design of distributed approximation algorithms. It allows to postpone symmetry breaking to the end of the algorithm where the fractional solution is converted to an integer one.

Acknowledgements. We would like to thank Maurice Cochand, Juraj Hromkovič, David Peleg, Peter Widmayer, and Aaron Zollinger for fruitful discussions about the subject. We would further like to thank the anonymous reviewers for valuable comments and corrections.

References

1. Alzoubi K, Wan P-J, Frieder O: 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), EPFL Lausanne, Switzerland, 2002, pp 157–164
2. Bartal Y, Byers JW, Raz D: Global Optimization Using Local Information with Applications to Flow Control. In: Proc. of the 38th IEEE Symposium on the Foundations of Computer Science (FOCS), 1997, pp 303–312
3. Berger B, Rompel J, Shor P: Efficient NC Algorithms for Set Cover with Applications to Learning and Geometry. J Comput Syst Sci 49:454–477 (1994)
4. Chvátal V: A Greedy Heuristic for the Set-Covering Problem. Math Oper Res 4(3):233–235 (1979)
5. Chvátal V: Linear Programming. W.H. Freeman and Company, 1983
6. Dubhashi D, Mei A, Panconesi A, Radhakrishnan J, Srinivasan A: Fast Distributed Algorithms for (Weakly) Connected Dominating Sets and Linear-Size Skeletons. In: Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA), 2003, pp 717–724
7. Feige U: A Threshold of $\ln n$ for Approximating Set Cover. J. ACM (JACM) 45(4):634–652 (1998)
8. Gao J, Guibas L, Hershberger J, Zhang L, Zhu A: Discrete Mobile Centers. In: Proc. of the 17th annual symposium on Computational geometry (SCG), ACM Press, 2001, pp 188–196
9. Garey MR, Johnson DS: Computers and Intractability, A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, 1979
10. Guha S, Khuller S: Approximation Algorithms for Connected Dominating Sets. In: Proc. of the 4th Annual European Symposium on Algorithms (ESA). Lecture Notes in Computer Science, vol 1136, 1996, pp 179–193
11. Jia L, Rajaraman R, Suel R: An Efficient Distributed Algorithm for Constructing Small Dominating Sets. In: Proc. of the 20th ACM Symposium on Principles of Distributed Computing (PODC), 2001, pp 33–42
12. Johnson DS: Approximation Algorithms for Combinatorial Problems. J Comput Syst Sci 9:256–278 (1974)
13. Karp RM: Reducibility Among Combinatorial Problems. In: Proc. of a Symposium on the Complexity of Computer Computations, 1972, pp 85–103
14. Kuhn F, Moscibroda T, Wattenhofer R: What Cannot Be Computed Locally! In: Proc. of the 23rd ACM Symposium on Principles of Distributed Computing (PODC), 2004, pp 300–309
15. Kutten S, Peleg D: Fast Distributed Construction of Small k-Dominating Sets and Applications. J Algorithms 28:40–66 (1998)
16. Lovasz L: On the Ratio of Optimal Integral and Fractional Covers. Discrete Math 13:383–390 (1975)
17. Luby M, Nisan N: A Parallel Approximation Algorithm for Positive Linear Programming. In: Proc. of the 25th ACM Symposium on Theory of Computing (STOC), 1993, pp 448–457
18. Raghavan P, Thompson CD: Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs. Combinatorica 7(4):365–374 (1987)
19. Rajagopalan S, Vazirani V: Primal-Dual RNC Approximation Algorithms for Set Cover and Covering Integer Programs. SIAM J Comput 28:525–540 (1998)
20. Rajaraman R: Topology Control and Routing in Ad hoc Networks: A Survey. SIGACT News 33:60–73 (2002)
21. Slavík P: A Tight Analysis of the Greedy Algorithm for Set Cover. In: Proc. of the 28th ACM Symposium on Theory of Computing (STOC), 1996, pp 435–441
22. Wu J, Li H: 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), 1999, pp 7–14