An Efficient Distributed Algorithm for Constructing Small Dominating Sets

Lujun Jia* Rajmohan Rajaraman† Torsten Suel‡

Abstract

The dominating set problem asks for a small subset D of nodes in a graph such that every node is either in D or adjacent to a node in D. This problem arises in a number of distributed network applications, where it is important to locate a small number of centers in the network such that every node is nearby at least one center. Finding a dominating set of minimum size is NP-complete, and the best known approximation is logarithmic in the maximum degree of the graph and is provided by the same simple greedy approach that gives the well-known logarithmic approximation result for the closely related set cover problem.

We describe and analyze new randomized distributed algorithms for the dominating set problem that run in polylogarithmic time, independent of the diameter of the network, and that return a dominating set of size within a logarithmic factor from optimal, with high probability. In particular, our best algorithm runs in $O(\log n \log \Delta)$ rounds with high probability, where n is the number of nodes, Δ is one plus the maximum degree of any node, and each round involves a constant number of message exchanges among any two neighbors; the size of the dominating set obtained is within $O(\log \Delta)$ of the optimal in expectation and within $O(\log n)$ of the optimal with high probability. We also describe generalizations to the weighted case and the case of multiple covering requirements.

*College of Computer Science, Northeastern University, Boston, MA 02115, Email: lujunjia@ccs.neu.edu. Supported by NSF CAREER award NSF CCR–9983901.

†College of Computer Science, Northeastern University, Boston, MA 02115, USA. Email: rraj@ccs.neu.edu. Supported by NSF CAREER award NSF CCR–9983901.

‡Department of Computer and Information Science, Polytechnic University, Brooklyn, NY 11201, Email: suel@poly.edu.

1 Introduction

Given an undirected graph $G = (V, E)$, a *dominating set* D of G is a subset of V such that for every node $v \in V$, either $v \in D$ or there exists a node $u \in D$ such that $(u, v) \in E$. The *minimum dominating set problem*, henceforth referred to as MDS, is to find a dominating set of minimum size. In this paper, we devise fast distributed approximation algorithms for MDS and its generalizations.

The MDS problem is a classic NP-complete optimization problem [12] and is closely tied to the well-known set cover problem. Given a universe $\mathcal{U}$, containing n elements, and a collection $\mathcal{S} = \{S_i : S_i \subseteq \mathcal{U}\}$ of subsets of $\mathcal{U}$, the *minimum set cover problem* asks for the minimum-size subcollection $\mathcal{C} \subseteq \mathcal{S}$ that covers all of the n elements in $\mathcal{U}$. The set cover problem, henceforth referred to as MSC, was one of the first problems shown to be NP-complete [17]. It can be easily seen that MDS is a special case of MSC; in any instance I of MDS, for each node v, the corresponding instance of MSC has an element v and a set S_v that consists of v and all the neighbors of v in I. On the other hand, MSC is equivalent to a version of MDS that has the constraint that the desired dominating set can only contain nodes from a prespecified subset of nodes. Both MDS and MSC can be approximated well by a natural greedy algorithm which repeatedly adds the node (resp., set) that covers the most number of uncovered nodes (resp., elements). The greedy algorithm achieves an H_Δ-approximation where Δ equals the maximum degree of a node plus one (resp., maximum number of elements in a set), and H_i is the ith harmonic number [9, 16, 22]. Furthermore, Feige has shown that the approximation ratio achieved by the greedy algorithm for either problem is the best possible (to within a lower order additive term) unless NP has $n^{O(\log \log n)}$-time deterministic algorithms [11].

Dominating sets and their variants (e.g., k-dominating sets) are useful in a number of distributed network applications, where the goal is to select a subset of nodes that will provide a certain service, such that every node in the network is close to some node in the subset. For example, a set of k-dominating centers can be selected to locate servers [5] or copies of a distributed directory [26] so that every node is within distance k of some server or directory copy, respectively. In recent work on mobile ad-hoc network routing [20], dominating sets have been proposed for storing location information of the network nodes. Dominating sets have also been used in the distributed construction of minimum spanning trees [19].

1.1 Our contributions

Our main contribution is a fast randomized distributed approximation algorithm for MDS. Our algorithm yields a dominating set of size within $O(H_\Delta) = O(\log \Delta)$ of the optimal in expectation and $O(\log n)$ of the optimal whp[1] and terminates in $O(\log n \log \Delta)$ time whp, where n is the number of nodes in the network. Our algorithm consists of a simple local procedure in which each node repeatedly performs a small number of fixed operations. For our analysis, we adopt a standard synchronous model of computation [3, 4, 21, 25] in which, in a communication step, each node can exchange a message with each of its neighbors; the time complexity of an algorithm is measured by the total number of communication steps. Our focus in this paper is on the time complexity and the approximation factor achieved; we do not attempt to optimize message complexity. Nevertheless, we note that all of our time bounds require $O(\log n)$-bit messages only. Although our results are stated for a synchronous model, our algorithm can be implemented in an asynchronous model using a standard synchronizer such as the α synchronizer of [2].

We generalize our main result in two directions. We consider the *multi-dominating set* problem (MMDS) in which we are also given for each node v a coverage requirement $r(v)$, and the goal is to find a dominating set that covers v at least $r(v)$ times, for all $v \in V$. For MMDS, our randomized

[1] We use the abbreviation "whp" throughout the paper to mean "with high probability" or, more precisely, "with probability $1 - n^{-c}$, where n is the number of nodes in the network and c is a constant that can be set arbitrarily large by appropriately adjusting other constants defined within the relevant context."

algorithm terminates in $O(\log n \log R \log \Delta)$ time whp and achieves an approximation ratio of $O(\log \Delta)$ in expectation and $O(\log n)$ whp, where R equals $\max_{u \in V} r(u)$. Our approximation results also extend to the weighted dominating set problem in which the nodes have weights and we wish to select a dominating set of minimum total weight; while we achieve the same expected approximation ratio as for MDS, the time complexity increases to $O(\log n \log(\Delta W))$, where W is the ratio of the maximum and minimum weights.

1.2 Related work

Our algorithm is a refinement of a distributed implementation of the greedy algorithm, in which we relax the criteria for adding a node to the dominating set and use randomization to break symmetries when several nodes attempt to add to the dominating set. The notion of relaxing the selection criteria is not new and has been used before, most notably in the parallel set cover algorithms of Berger et al [7] and Rajagopalan and Vazirani [28]. They describe RNC^5 and RNC^3 algorithms, respectively, that achieve an $O(\log \Delta)$-approximation. These results, however, make use of certain global synchronization steps or computations which can be performed efficiently in parallel but do not directly lend themselves to fast distributed implementations[2]. Furthermore, our particular relaxation and selection criteria are new and the resulting algorithms are different than the ones considered in [7, 28]. We note that [28] also considers other generalizations of set cover including MMDS, which we study as well. The RNC algorithm of [28] has been derandomized in [8].

Kutten and Peleg describe a distributed dominating set algorithm which takes $O(\log^* n)$ time [19] on any network, and hence is asymptotically much faster than the algorithms we propose in this paper. Their algorithm is based on the $O(\log^* n)$ algorithm of [13] for finding a maximal independent set in graphs, which uses the deterministic coin-flipping technique of [10]. The primary emphasis in [19] is on time complexity, however, and there is no nontrivial asymptotic upper bound on the approximation ratio. In contrast, Haas and Liang [20] have recently presented a distributed implementation of the greedy algorithm, which we refer to as *distributed greedy*, that achieves the same approximation ratio as the greedy algorithm; however, as we discuss in Section 2, there exist networks for which the distributed greedy algorithm takes $\Omega(n)$ time.

Closely related to MDS and MSC is positive linear programming, which captures linear relaxations of the two problems. Luby and Nisan [23] introduce an NC algorithm for positive linear programming. As noted in [23], their result taken together with the randomized rounding approach of Raghavan [27] yields an RNC algorithm for MSC and MDS. In recent work, Bartal et al. [6] present a distributed algorithm for positive linear programming with applications to flow control.

1.3 Outline of the paper

The remainder of the paper is organized as follows. Section 2 presents our randomized distributed algorithm. Section 3 establishes a tight analysis of the algorithm. Section 4 presents generalizations of our basic result. Finally, Section 5 discusses future research directions.

2 A randomized distributed algorithm for dominating sets

In this section, we present a simple randomized algorithm for MDS that admits an efficient distributed implementation. The algorithm is developed in Section 2.1 and the distributed implementation is discussed in Section 2.2.

[2] It is worth mentioning that the algorithms of [7] and [28] were not intended for the distributed computation model that we consider here; they were primarily designed for the PRAM model.

2.1 The algorithm

Our randomized algorithm is a refinement of the distributed greedy algorithm of [20][3]. In order to motivate our algorithm, we first present the distributed greedy algorithm and discuss its time complexity. The distributed greedy algorithm proceeds in rounds, and each round consists of 3 steps. First, each node u calculates its *span*, which is the number of yet uncovered nodes that u covers. (Note that the span of a node initially equals its degree plus one.) Next, each node u sends its span and its ID to all nodes within distance 2 of u. The final step is the *selection step*, in which a node u adds itself to the dominating set if its ordered pair of span and ID is lexicographically higher than that of any node within distance 2.

The above algorithm is a faithful implementation of the sequential greedy algorithm, and consequently the approximation ratio achieved is identical to that of the greedy algorithm. The distributed time complexity of the algorithm, however, could be polynomial in the number of nodes. To see this, we first note that in the distributed greedy algorithm a node adds itself to the dominating set in a round only if its span is maximum among all nodes within distance 2. Thus, in networks where there is a long chain of nodes with decreasing degrees, the number of rounds until completion may be proportional to the length of the chain. For instance, the distributed greedy algorithm takes time $\Omega(\sqrt{n})$ to obtain a dominating set for the caterpillar graph shown in Figure 1. The problem raised by the caterpillar graph

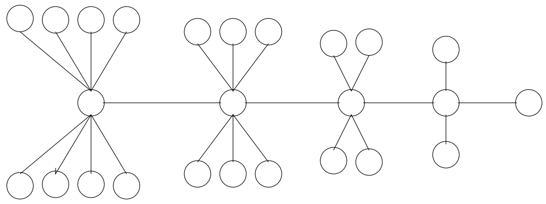

Figure 1: The caterpillar graph.

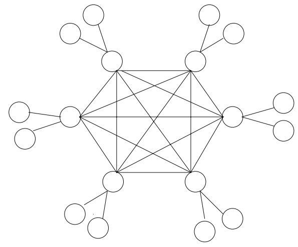

Figure 2: The star-complete network.

can be addressed by "rounding up" the span of each node to the nearest power of 2 and allowing any node that has maximum rounded span among nodes within distance 2 to participate in the selection step. More precisely, in the selection step a node adds itself to the dominating set if its ordered pair of rounded span and ID is lexicographically higher than that of any other node within distance 2. (This natural idea of grouping nodes with spans to within a constant factor is also used in the RNC algorithm of [7].) It can be easily seen that such a modification will yield $O(1)$-approximate dominating sets in $O(\log n)$ rounds for networks like the caterpillar graph. Both the distributed greedy algorithm and the modification with grouped spans, however, suffer from another serious drawback that results in a linear running time even for constant-diameter networks.

Consider the network of Figure 2, which consists of a complete network $K_{n/3}$ with two additional neighbors for each node of the network. It is easy to see that in each round, the distributed greedy algorithm or the modification with grouped spans will add exactly one node from $K_{n/3}$ into the dominating set; hence the algorithm will complete in $n/3$ rounds. The primary reason for the inefficiency of the two algorithms is that they only consider the span of the nodes but not how the nodes covered by an individual node are also covered by other nodes. This leads us to our algorithm, which we refer to as the *Local Randomized Greedy* algorithm, or LRG for short.

The LRG algorithm proceeds in rounds. We now describe the computation in each round.

- **Span calculation:** We first calculate for each v the span $d(v)$, which equals the number of uncovered nodes that are adjacent to v (including v itself if uncovered). Let the *rounded span*

[3]What we call the distributed greedy algorithm is referred to as the *distributed database coverage heuristic* (DDCH) in [20], where it is proposed as an algorithm for locating routing databases in mobile ad-hoc networks.

$\widehat{d}(v)$ of node v be the smallest power of *base* b that is at least $d(v)$, where $b > 1$ is a real constant. The parameter b presents a tradeoff between time complexity and approximation ratio.

- **Candidate selection:** We say that a node $v \in V$ is a *candidate* if $\widehat{d}(v)$ is at least $\widehat{d}(w)$ for all $w \in F$ within distance 2 of v. For each candidate v, let *cover* $C(v)$ denote the set of uncovered nodes that v covers.

- **Support calculation:** For each uncovered node u, we calculate its support $s(u)$, which is the number of candidates that cover u.

- **Dominator selection:** For each candidate v, we add v to D with probability $1/med(v)$, where $med(v)$ is the median support of all the nodes in $C(v)$.

We now provide some intuition for the definition of the dominator selection step of LRG. Consider a node u that is covered by $s(u)$ candidates. We would like the dominator selection step to include at least one, but not many, dominators that cover u. One approach to breaking the symmetry among the candidates that cover u is to select each of the candidates as a dominator independently with a certain probability. Suppose we set this probability to $1/s(u)$ for each of the candidates that cover u. Then u is covered with constant probability; furthermore, the expected number of dominators that cover u in this step is only 1. It is not hard to see, however, that no setting of the probabilities will guarantee the preceding property to be satisfied for all nodes simultaneously. A key challenge in this approach for dominator selection is to set the probabilities such that a small number of dominators are selected, and yet a large fraction of nodes are covered by the dominators.

In LRG, we set the probability $P_s(v)$ of selecting a candidate v as a dominator to be the median of the reciprocal of the supports (or equivalently, the reciprocal of the median support) of all the nodes that are covered by v. One consequence of this choice is that if a node u has support larger than $1/P_s(v)$ for more than half the candidates v that cover u, then u is covered with constant probability. Furthermore, we are able to show that the expected ratio of the number of nodes covered to the sum of the spans of the dominators selected is constant. The preceding two claims are key ingredients in the efficiency and the effectiveness (in terms of the approximation ratio) of LRG, which are analyzed in Sections 3.1 and 3.2, respectively.

An alternative natural choice for the setting of the probabilities in the dominator selection step is to select a candidate as a dominator with a probability equal to the average of the reciprocal of the supports of the nodes covered by the candidate. In Section 4.1, we show that this approach also leads to a $O(\text{polylog}(n))$-time $O(\text{polylog}(n))$-approximation algorithm.

2.2 Distributed implementation

The span calculation step is implemented by simply exchanging identification messages with those neighbors that are still uncovered. For candidate selection, each node v sends $d(v)$ to all of its neighbors; each neighbor then computes the maximum and forwards it one more step, resulting in at most one message per step and communication link. After this second step, every node can determine whether it is a candidate. For support calculation, each node v sends a message to each of its neighbors indicating whether v is a candidate. On the receipt of the "candidacy" messages from all of the neighbors, a node determines its support. The final dominator selection step can then be implemented by having each node send its support to each of the neighboring candidates, and then each candidate selects itself with the appropriate probability. Once a node u and all of the neighbors of u are covered, u need not participate in the algorithm any longer (other than as an intermediate node for routing messages).

3 Analysis

In this section, we show that LRG terminates in $O(\log n \log \Delta)$ rounds whp and yields a dominating set of expected size $O(\log \Delta)$ times the optimal, where Δ is one plus the maximum degree of the network. Section 3.1 analyzes the running time of the algorithm and Section 3.2 analyzes the approximation ratio of the algorithm. Finally, in Section 3.3, we establish the tightness of our analysis.

3.1 Time complexity

We place a bound on the time complexity by analyzing the progress of the algorithm in any given round. Fix a round and let $H = (V', E')$ be the subgraph of G where V' is the union of the set of all candidates in the round and the set of nodes covered by the candidates, and E' is the set of edges (v, w) where v is a candidate in the round and $w \in C(v)$. We first show that any two candidates in the same connected component of H have the same rounded span.

Lemma 3.1 *If v_1 and v_2 are two candidates in any connected component of H, then $\widehat{d}(v_1) = \widehat{d}(v_2)$.*

Proof: Consider any path p connecting v_1 and v_2. From the definition of subgraph H, at least every alternate node on p is a candidate. From the definition of a candidate, two candidates within distance 2 have the same rounded span. Hence, the desired claim holds. ■

We establish the progress of the algorithm by a potential function argument similar to the one used in [28]. We define the *i-potential* Φ to be $\sum_{v:\widehat{d}(v)=i} |C(v)|$. Let m denote the maximum rounded span of any node at the start of a round. The following main lemma establishes a bound on the expected decrease in the m-potential in any round.

Lemma 3.2 *If m is the maximum rounded span of any node at the start of a round and Φ and Φ' are the m-potentials at the start and end of a round, then $E[\Phi'] \leq d\Phi$ for some positive constant $d < 1$.*

To prove Lemma 3.2, we first introduce some definitions. For each candidate v, we sort $C(v)$ according to the supports of the nodes in $C(v)$ in nonincreasing order. Let $T(v)$ and $B(v)$ denote the set of the top $\lceil d(v)/2 \rceil$ (resp., bottom $\lceil d(v)/2 \rceil$) nodes in $C(v)$ in this sorted order. (Note that if $d(v)$ is odd, then $T(v)$ and $B(v)$ share a common element.) For a candidate v and a node u, we say that v is *good for u* if $u \in T(v)$. We say that a given node u is *nice* if at least $s(u)/4$ candidates covering u are good for u. Let $P_s(v)$ denote the probability that candidate v will be added to D at the end of the round, and $P_c(u)$ denote the probability that node u will be covered at the end of the round. The following lemma places a lower bound on $P_c(u)$ for any nice node u.

Lemma 3.3 *If u is nice, then $P_c(u) > 1 - \frac{1}{\sqrt[4]{e}}$.*

Proof: For any candidate v that is good for u, we have $P_s(v) = 1/med(v) \geq 1/s(u)$. Then, for a nice node u, we have

$$\begin{aligned} P_c(u) &= 1 - \prod_{u \in C(v)} (1 - P_s(v)) \\ &\geq 1 - \prod_{v \text{ is good for } u} (1 - P_s(v)) \\ &\geq 1 - \left(1 - \frac{1}{s(u)}\right)^{s(u)/4} \\ &> 1 - \frac{1}{\sqrt[4]{e}}. \end{aligned}$$

■

We refer to each element in $C(v)$, for any v with rounded span m, as an *entry*. Thus, Φ is simply the total number of entries (where a node may have entries in several sets $C(v)$). Any entry that occurs in $T(v)$, for some v, is referred to as a *top entry*. Finally, any occurrence of a nice node in $T(v)$ is referred to as a *nice top entry*.

Lemma 3.4 *At least one-third of the total entries are nice top entries.*

Proof: Fix a connected component S in which the maximum rounded span is m. By Lemma 3.1, all candidates in S have the same rounded span. Let x (resp., y) denote the total number of entries (resp., number of non-nice top entries) in S. It follows from the definition of nice nodes that for any nice node u the number of candidates v such that $T(v)$ contains u is at least $s(u)/4$. Therefore, there exist at least $3y$ non-nice node entries in $B(v)$. Clearly, $x/2 - 3y \geq 0$, from which we get $y \leq x/6$. Thus, the total number of nice node entries in $T(v)$ is $x/2 - y \geq x/3$. Adding over all connected components with rounded span m, we obtain the desired claim. ■

Proof of Lemma 3.2: Let v be any candidate in S and let y denote the number of nice top entries in $C(v)$ at the start of the round and let z denote the number of nice top entries covered in the round. By Lemma 3.3, we have

$$E[z] > y(1 - \frac{1}{\sqrt[4]{e}}).$$

Adding over all components with rounded span m and invoking Lemma 3.4, we obtain that

$$E[\Phi'] \leq \Phi(1 - \frac{1}{3}(1 - \frac{1}{\sqrt[4]{e}})),$$

which completes the proof of the desired claim. ■

We are now ready to establish the bound on the running time of the algorithm.

Theorem 1 LRG *terminates in $O(\log n \log \Delta)$ rounds whp.*

Proof: We divide the running time of the algorithm into phases. A phase consists of a maximal sequence of rounds with the same maximum rounded span. The number of phases is at most the total number of distinct values for the rounded span, which is at most $\log_b \Delta$. We now show that the number of rounds in each phase is $O(\log n)$ whp. The proof follows from Lemma 3.2 and standard probabilistic analysis. We include the proof details for completeness.

Consider a phase with maximum rounded span m. Let $T(P)$ denote the number of rounds remaining in the phase when the m-potential at the start of a round is P. (Note that $T(P)$ is a random variable.) We have the following probabilistic recurrence for $T(\cdot)$. For $P > \lfloor m/b \rfloor$, we have

$$T(P) = a(P) + T(P'), \tag{1}$$

where $a(P) = 1$ for all P and P' is the random variable denoting the m-potential at the end of the round. For $P \leq \lfloor m/b \rfloor$, we have $T(P) = 0$.

We now invoke a result on probabilistic recurrence relations due to Karp [18, Theorem 1.3] to obtain the following claim. Let $w = \lfloor c \log_{\frac{1}{d}}(bn) \rfloor + 2$, where d is the constant in the statement of Lemma 3.2. For any instance with m-potential P, the number of rounds $T(P)$ until the end of the phase satisfies the inequality

$$\Pr[T(P) \geq \lfloor \log_{\frac{1}{d}}(\frac{Pb}{m}) \rfloor + 1 + w] \leq d^w = \frac{1}{n^c},$$

for any constant $c > 0$. We note that the term $\lfloor \log_{\frac{1}{d}}(\frac{Pb}{m}) \rfloor + 1$ is the solution to the recurrence $\tau(P) = a(P) + \tau(dP)$, which is a deterministic counterpart of Equation 1, corresponding to the case where the $m-$potential decreases by a factor of exactly d in each round; note that the expected decrease is lower bounded by a factor of d by Lemma 3.2.

Since P is at most bnm, it follows every phase ends in at most $(c+1)\log_{\frac{1}{d}}(bn)+3$ rounds with probability at least $1-1/n^c$.

Since the number of phases is is at most $\log_b \Delta$, we have

$$\Pr[\text{LRG terminates in } \log_b \Delta((c+1)\log_{\frac{1}{d}}(bn)+2) \text{ rounds}] \geq (1-\frac{1}{n^c})^{\log_b \Delta} \geq 1-\frac{1}{n^{c-1}},$$

for a sufficiently large n. This yields the desired high probability bound on the running time of LRG. ■

3.2 Approximation ratio

In this section, we place a bound on the size of the dominating set constructed by the algorithm. Let OPT denote the minimum dominating set for the given instance. The main result of this section is the following.

Theorem 2 LRG *yields a dominating set of expected size* $4bH_\Delta \cdot |OPT|$ *and size* $O(|OPT| \cdot \log n)$ *whp.*

We estimate the size of the dominating set by associating a cost with each node covered in a given round, and then placing a bound (in Lemma 3.6) on the expected number of dominators selected in any round in terms of the expected cost of the nodes covered. For node u, the cost $cost(u)$ is determined in the round when u is covered. Consider a round when u is covered and let v be a dominator that covers u in that round and is also selected in that round. We set $cost(u)$ to be $1/\widehat{d}(v)$. Recall that, due to Lemma 3.1, all candidates that cover u in a given round have the same rounded span, so the particular choice of v is immaterial. Furthermore, u is covered in exactly one round; thus, $cost(u)$ is well-defined. The sum of $cost(u)$, taken over all the nodes u, is closely tied to the performance of the greedy algorithm, and the following lemma, which is based on the well-known analysis of the greedy algorithm [16, 22], places an upper bound on this sum.

Lemma 3.5 $\sum_{u \in V} cost(u) \leq H_\Delta \cdot |OPT|$.

Proof: Consider any $v \in \text{OPT}$. Let $C_{\text{OPT}}(v)$ denote the set of nodes covered by v at the beginning of the algorithm, and let ℓ be $|C_{\text{OPT}}(v)|$. We sort all $u \in C_{\text{OPT}}(v)$ to obtain the sequence u_1, u_2, ...,u_ℓ such that for any $1 \leq i < j \leq \ell$, u_i is assigned a cost not after u_j. Then we have $cost(u_i) \leq \frac{1}{\ell-i+1}$. Consequently,

$$\sum_{u \in C_{\text{OPT}}(v)} cost(u) = \sum_{i=1}^{\ell} cost(u_i) \leq \frac{1}{\ell} + \frac{1}{\ell-1} + \ldots + \frac{1}{1} = H_\Delta.$$

Since every vertex u is covered by some vertex v in OPT, we have

$$\sum_{u \in V} cost(u) \leq \sum_{v \in \text{OPT}} \sum_{u \in C_{\text{OPT}}(v)} cost(u) \leq H_\Delta \cdot |\text{OPT}|.$$

■

Lemma 3.6 *If S is the set of candidates added to D in any round and Z is the total cost assigned in the round, then $E[|S|] \leq 4bE[Z]$.*

Proof: Let V' denote the set of nodes that are uncovered at the start of the round. For any $u \in V'$, let $c(u) = 1/\widehat{d}(v)$, where $u \in C(v)$. For any $v \in S$, we have $|C(v)| \geq \widehat{d}(v)/b$. We thus obtain

$$|S| \leq \sum_{v \in S} |C(v)| \frac{b}{\widehat{d}(v)} \leq \sum_{v \in S} 2|B(v)| \frac{b}{\widehat{d}(v)} \leq 2b \sum_{v \in S} \sum_{u \in B(v)} c(u).$$

For any $u \in V'$, let $t(u)$ denote the number of candidates v that are added to D at the end of this round such that $u \in B(v)$. Note that $t(u) = 0$ if u is not covered or $u \notin B(v)$ for all $v \in S$ that cover u. Thus, we have

$$|S| \leq 2b \sum_{v \in S} \sum_{u \in B(v)} c(u) = 2b \sum_{u \in V'} c(u) t(u).$$

Since $c(u)$ is fixed in a given round and $E[t(u)]$ equals $\Pr[t(u) > 0] \cdot E[t(u) \mid t(u) > 0]$, we obtain

$$E[|S|] \leq 2b \sum_{u \in V'} c(u) \Pr[t(u) > 0] \cdot E[t(u) | t(u) > 0]. \tag{2}$$

Now consider $t(u)$. Let $W = \{v : u \in B(v)\}$ and let $p(v)$ denote the probability that v is added to D for a given $v \in W$. For $v \in W$, since $u \in B(v)$, we have $p(v) \leq 1/med(v) \leq 1/s(u) \leq 1/|W|$. Thus,

$$\begin{aligned}
E[t(u)|t(u) > 0] &= \sum_{v \in W} \Pr[v \text{ is added to } D | t(u) \geq 1] \\
&= \sum_{v \in W} \frac{\Pr[v \text{ is added to } D]}{\Pr[t(u) \geq 1]} \\
&= \frac{\sum_{v \in W} p(v)}{1 - \prod_{v \in W} (1 - p(v))} \\
&\leq \frac{\sum_{v \in W} p(v)}{\sum_{v \in W} p(v) - \sum_{x,y \in W, x \neq y} p(x)p(y)} \\
&\leq 2.
\end{aligned}$$

(In the fourth step, we use the inequality $\prod_{v \in W}(1 - p(v)) \leq 1 - \sum_{v \in W} p(v) + \sum_{x,y \in W, x \neq y} p(x)p(y)$. In the final step, we use the inequality $\sum_{x,y \in W, x \neq y} p(x)p(y) \leq \sum_{v \in W} p(v)/2$ since $\sum_{v \in W} p(v) \leq 1$.)

Substituting the bound on $E[t(u)|t(u) > 0]$ in Equation 2, we get

$$E[|S|] \leq 4b \sum_{u \in V'} c(u) \Pr[t(u) > 0] \leq 4b E[Z],$$

which completes the proof of Lemma 3.6. ■

Proof of Theorem 2: Let S_i denote the set of candidates added to the dominating set in round i, and let Z_i denote the cost assigned in round i. Then the expected size of the dominating set computed by LRG is

$$\sum_i E[|S_i|] \leq 4b \sum_i E[Z_i] = 4b \sum_{u \in V} E[cost(u)] \leq 4b H_\Delta \cdot |\mathrm{OPT}|,$$

where the last step follows from Lemma 3.5. This completes the bound on the expected size of the dominating set.

For a high probability bound, we note that each round can be viewed as a set of independent coin tosses, one for each node in the network. Let $p_i(v)$ denote the probability of selecting node v in round i. Note that $p_i(v)$ is a random variable and may be dependent on the outcome of earlier rounds. For instance, if a node v is selected in the dominating set in round i, then $p_j(v)$ is 0 for $j > i$. Therefore, the overall sequence of coin tosses does not correspond to a set of independent Bernoulli trials. However, since we have shown that $\sum_i E[|S_i|]$ is upper bounded by $4b\sum_{u\in V} E[cost(u)]$, and $\sum_{u\in V} cost(u)$ is *always* bounded by $H_\Delta \cdot |\mathrm{OPT}|$, it follows that if we consider any particular execution of the random experiment, the sum of the probabilities of the coin tosses is upper bounded by $H_\Delta \cdot |\mathrm{OPT}|$. We now invoke a Chernoff-type argument to establish the high probability bound on the size of the dominating set. We formalize this approach in the following.

We construct a tree T that captures all possible executions of LRG on the given network. Each path in T depicts one execution of LRG. Consider round i in a particular execution E of LRG. The outcomes of rounds 1 though $i-1$ specify a particular path from the root of T to a vertex x. The vertex x depicts the coin tosses corresponding to the random dominator selections in round i. Note that there are 2^n theoretically possible outcomes of this round.

We then transform T into a binary tree T_b. Any vertex of T_b depicts an independent coin toss which corresponds to the random event that a particular network node is selected as a dominator at that point of the execution. As mentioned above, the path from root to any vertex in T_b, say x, signifies a particular execution of LRG, say E. We assign x a *vertex value* of p_x, which is the probability that the network node corresponding to vertex x is selected as a dominator at the last step of the execution E. The left (resp., right) branch from x is assigned an *edge value* of $1 - p_x$ (resp., $-p_x$).

Furthermore, we transform T_b into a full binary tree T_f. We first add extra vertices to make the total of the vertex values along any path of T_f equal to the maximum of them. Note that for each path, adding at most n vertices is sufficient, because a new vertex could be assigned vertex value 1. Next, by adding vertices with vertex value 0, we make the binary tree a full binary tree.

We now analyze T_f. With each vertex x in T_f with vertex value p_x, we associate a random variable X such that $\Pr[X = 1 - p_x] = p_x$ and $\Pr[X = -p_x] = 1 - p_x$. Thus, $X = 1 - p_x$ corresponds to the left branch of x, and $X = -p_x$ corresponds to the right branch. We also associate with each non-leaf vertex x a random variable Z_x, which denotes the sum of the edge values along any path from x to a leaf vertex. We call Z_x the *sum variable* of x. Specifically, let Z be the sum variable of the root vertex of T_f. For vertex x with two leaf children, it is easy to see that $E[Z_x] = p_x(1-p_x) - p_x(1-p_x) = 0$. By induction, let x_1, x_2 be the left and right children of x in T_f, with $E[Z_{x_1}] = E[Z_{x_2}] = 0$. We have $E[Z_x] = p_x(1 - p_x + E[Z_{x_1}]) + (1-p_x)(-p_x + E[Z_{x_2}]) = 0$. Thus, we have shown that for any vertex x in T_f, $E[Z_x] = 0$. Let $\bar{p}_x$ be the *average vertex value* of x which is the average of the vertex values along any path from x to a leaf vertex. (Note that the average of the vertex values along any path down from x is the same.) Specifically, let $\bar{p}$ be the average vertex value of the root vertex in T_f. Let N be the depth of T_f, and define *height* k for vertex x to be the number of vertices below x on any path of T_f. Let S be the size fo the dominating set computed by LRG. It is easy to see that $S = Z + \bar{p}N$. The remainder of the proof rests on the inequality

$$\Pr[Z \geq (\beta - 1)\bar{p}N] < (e^{\beta-1}\beta^{-\beta})^{\bar{p}N}, \tag{3}$$

where $\beta \geq 1$. We will prove Equation 3 shortly. Note that $E[S] \leq 4b(\ln\Delta)|\mathrm{OPT}|$ and $E[Z] = 0$. By setting $\beta = 1 + (\ln n)/\ln\Delta$ in Equation 3, we have

$$\begin{aligned}
\Pr[Z \geq 4b(\ln n)|\mathrm{OPT}|] &\leq (e^{\beta-1}\beta^{-\beta})^{4b(\ln\Delta)|\mathrm{OPT}|} \\
&= \left(e^{\frac{\ln n}{\ln\Delta}}\big(1 + \frac{\ln n}{\ln\Delta}\big)^{-(\frac{\ln n}{\ln\Delta})}\right)^{4b(\ln\Delta)|\mathrm{OPT}|} \\
&\leq \big(\frac{1}{4^{\ln 4 - 1}}\big)^{4b|\mathrm{OPT}|} \\
&< \frac{1}{n^{1.5}}.
\end{aligned}$$

Thus,

$$\Pr[S \geq 8b(\ln n)|\mathrm{OPT}|] \leq \Pr[Z \geq 4b(\ln n)|\mathrm{OPT}|] \quad < \quad \frac{1}{n^{1.5}},$$

and the desired high probability bound on the approximation ratio follows. It thus remains to prove Equation 3. Our proof follows the lines of the derivation of Chernoff bounds as described in [1, Appendix A].

We will show by induction that for any vertex x with height k, average vertex value $\bar{p}_x$ and sum variable Z_x,

$$E[e^{\lambda Z_x}] \leq e^{-\lambda k \bar{p}_x}(\bar{p}_x e^{\lambda} + (1-\bar{p}_x))^k$$

holds for any $\lambda > 0$.

For the base case, where $k = 1$ and x has two leaf children, it is easy to see that

$$\begin{aligned} E[e^{\lambda Z_x}] &= p_x e^{\lambda(1-p_x)} + (1-p_x)e^{\lambda(-p_x)} \\ &\leq e^{-\lambda k \bar{p}_x}(\bar{p}_x e^{\lambda} + (1-\bar{p}_x))^k, \end{aligned}$$

since $p_x = \bar{p}_x$. Consider induction step k. Let x_1, x_2 be the left and right children of vertex x, with

$$E[e^{\lambda Z_{x_1}}] \leq e^{-\lambda k \bar{p}_{x_1}}(\bar{p}_{x_1} e^{\lambda} + (1-\bar{p}_{x_1}))^k$$

and

$$E[e^{\lambda Z_{x_2}}] \leq e^{-\lambda k \bar{p}_{x_2}}(\bar{p}_{x_2} e^{\lambda} + (1-\bar{p}_{x_2}))^k.$$

Note that $\bar{p}_{x_1} = \bar{p}_{x_2} = p'$. Thus, we have

$$\begin{aligned} E[e^{\lambda Z_x}] &= p_x e^{\lambda(1-p_x)} E[e^{\lambda Z_{x_1}}] + (1-p_x)e^{-\lambda p_x} E[e^{\lambda Z_{x_2}}] \\ &\leq e^{-\lambda k p'}(p' e^{\lambda} + (1-p'))^k e^{-\lambda p_x}(p_x e^{\lambda} + (1-p_x)) \\ &\leq e^{-\lambda(k+1)\bar{p}_x}(\bar{p}_x e^{\lambda} + (1-\bar{p}_x))^{k+1}. \end{aligned}$$

The last step in the above inequality is by the fact that function $f(x) = \ln(e^{\lambda}x + 1 - x)$ is concave, which leads to the inequality $kf(p') + f(p_x) \leq (k+1)f(\bar{p}_x)$. Exponentiating both sides, the above inequality follows. By induction, we now obtain

$$E[e^{\lambda Z}] \leq e^{-\lambda \bar{p} N}(\bar{p} e^{\lambda} + (1-\bar{p}))^N.$$

Invoking Markov's Inequality, we obtain

$$\Pr[Z \geq a] = \Pr[e^{\lambda Z} \geq e^{\lambda a}] < e^{-\lambda \bar{p} N}(\bar{p} e^{\lambda} + (1-\bar{p}))^N e^{-\lambda a}.$$

Substituting $\lambda = \ln(1 + a/(\bar{p}N))$ and $a = (\beta - 1)\bar{p}N$, Equation 3 follows, which completes the proof of Theorem 2. ■

We now consider a tradeoff between the time complexity and the expected approximation ratio achieved by the algorithm. Theorems 1 and 2 show that LRG achieves an expected $4bH_{\Delta}$-approximation and terminates in $O(\log n \log \Delta)$ rounds whp. A slight variant of LRG yields an approximation arbitrarily close to H_{Δ} by giving up an appropriate constant factor in the number of rounds. Let ε be any positive real constant. We modify the candidate selection phase of LRG as follows. Instead of selecting a candidate v with probability $1/med(v)$, we select a candidate with probability $\varepsilon/r_{\varepsilon}(v)$, where $r_{\varepsilon}(v)$ is the support of rank $\lceil \varepsilon d(v) \rceil$ among the supports of all nodes in $C(v)$ (recall that $d(v)$ is the span of v). Finally, we set b to $1 + \varepsilon$. We now establish the following bounds on the new algorithm, which we refer to as ε-LRG.

Theorem 3 *Algorithm ε-LRG terminates in $O(\log n \log \Delta)$ rounds whp and achieves an expected approximation ratio of $(1+3\varepsilon)H_\Delta$, for any sufficiently small constant $\varepsilon > 0$.*

Theorem 3 implies that algorithm ε-LRG achieves an expected approximation ratio arbitrarily close to H_Δ. The analysis of ε-LRG follows the lines of the analysis of LRG. In the following, we focus on the points at which the two analyses differ.

We first note that Lemma 3.1 holds as before. We next modify the definitions of the sets $T(v)$ and $B(v)$ as follows. Let $T(v)$ and $B(v)$ denote the set of the top $\lceil \varepsilon d(v) \rceil$ and bottom $d(v) - \lfloor \varepsilon d(v) \rfloor$ entries, respectively. (Note that $T(v)$ and $B(v)$ may have one element in common.) As before, we say that candidate v is *good* for u if $u \in T(v)$. We say that a node u is nice if at least $\varepsilon s(u)/2$ candidates covering u are good for u. We then obtain the following lower bound on $P_c(u)$.

Lemma 3.7 *If u is nice, then $P_c(u) > 1 - \frac{1}{e^{\varepsilon^2/2}}$.*

Proof: For any candidate v that is good for u, we have $P_s(v) \geq 1/s(u)$. Then, for a nice node u, we have

$$\begin{aligned} P_c(u) &= 1 - \prod_{u \in C(v)} (1 - P_s(v)) \\ &\geq 1 - \left(1 - \frac{\varepsilon}{s(u)}\right)^{\varepsilon s(u)/2} \\ &> 1 - \frac{1}{e^{\varepsilon^2/2}}. \end{aligned}$$

■

Analogous to Lemma 3.4, we obtain the following lemma.

Lemma 3.8 *At least an $(\varepsilon/2)$-fraction of the total entries are nice top entries.*

Proof: Fix a connected component S in which the maximum rounded degree is m. By Lemma 3.1, all candidates in S have the same rounded degree. Let x (resp., y) denote the total number of entries (resp., number of non-nice top entries) in S. By the definition of nice nodes, each nice node u has at least $\varepsilon s(u)/2$ entries in $T(v)$, for each candidate v. Therefore, there exist at least $2(1-\varepsilon/2)y/\varepsilon$ non-nice node entries in $B(v)$. Clearly, $(1-\varepsilon)x - 2(1-\varepsilon/2)y/\varepsilon \geq 0$, from which we get $y \leq \varepsilon x/2$. So, the total number of nice node entries in $T(v)$ is at least $\varepsilon x - \varepsilon x/2 = \varepsilon x/2$. Adding over all connected components with rounded degree m, we obtain the desired claim. ■

Lemma 3.7 and 3.8 imply Lemma 3.2, with the constant $d = 1 - (\varepsilon/2)(1 - 1/e^{\varepsilon^2/2})$, which is at least $1 - \varepsilon^3/4$. Using the Chernoff-bound argument as in the proof of Theorem 1 (with $\alpha = 1 - \varepsilon^3/8$, and $p = \varepsilon^3/8$), we obtain that the probability that ε-LRG does not terminate in $2(\log_b \Delta)(\ln_{1/\alpha}(bn))/p$ rounds is at most $1/n^2$, assuming that $\varepsilon > 0$ is sufficiently small. Thus, ε-LRG terminates in $O(\log \Delta \log n)$ rounds whp, where the hidden constant in O-notation is proportional to $1/\varepsilon^7$.

We next consider the approximation ratio. Lemma 3.5 holds as before. We now describe how to modify the proof of Lemma 3.6 to obtain the following lemma.

Lemma 3.9 *If S is the set of candidates added to D in any round and Z is the total cost assigned in the round, then $E[|S|] \leq (1+3\varepsilon)bE[Z]$.*

Proof: We define $c(u)$ as before. We obtain

$$|S| \leq \sum_{v \in S} |C(v)| \frac{b}{\widehat{d}(v)} = b \sum_{v \in S} \sum_{u \in C(v)} c(u) \leq (1+2\varepsilon) b \sum_{v \in S} \sum_{u \in B(v)} c(u),$$

since $|C(v)| \leq (1+2\varepsilon)|B(v)|$ for all v, assuming $\varepsilon > 0$ is sufficiently small. Defining $t(u)$ as in the proof of Lemma 3.6, we obtain

$$|S| < (1+2\varepsilon)b \sum_{v \in S} \sum_{u \in B(v)} c(u) = (1+2\varepsilon)b \sum_{u \in V'} c(u)t(u),$$

which yields

$$E[|S|] \leq (1+2\varepsilon)b \sum_{u \in V'} c(u) \Pr[t(u) > 0] E[t(u)|t(u) > 0]. \tag{4}$$

Our final step of the proof is to show that $E[t(u)|t(u) > 0]$ is $1 + O(\varepsilon)$. For this, we define, as in Lemma 3.6, $W = \{v : u \in B(v)\}$ and $p(v)$ to be the probability that v is added to D for a given $v \in W$. As before, we obtain the inequality

$$E[t(u)|t(u) > 0] \leq \frac{\sum_{v \in W} p(v)}{\sum_{v \in W} p(v) + \sum_{x,y \in W, x \neq y} p(x)p(y)}. \tag{5}$$

For $v \in W$, since $u \in B(v)$, we have $p(v) \leq \varepsilon/s(u) \leq \varepsilon/|W|$. Noting that $\sum_{x,y \in W, x \neq y} p(x)p(y)$ is at most $(\sum_{v \in W} p(v))^2/2$ and that $\sum_{v \in W} p(v)$ is at most ε, we obtain from Equation 5 that $E[t(u)|t(u) > 0]$ is at most $1/(1 - \varepsilon/2)$, which is at most $1 + \varepsilon$. Substituting the bound on $E[t(u)|t(u) > 0]$ in Equation 4 yields the desired claim. ■

Applying Lemma 3.9 over all rounds and invoking Lemma 3.5, we obtain the desired bound on the expected approximation ratio.

3.3 Tightness of the analysis

We now show that our analysis of LRG is tight to within constant factors. It is easy to see that LRG can do no better than the greedy algorithm. Therefore, the approximation ratio of LRG is at least that of the greedy algorithm, which is H_Δ (for a lower bound instance, see [16]). In this section, we present a network for which LRG takes $\Omega(\log n \log \Delta)$ rounds whp before termination. We assume for simplicity that the base b in LRG equals 2. However, our lower bound network can be generalized to any $b > 1$.

The lower bound network G is illustrated in Figure 3. The network consists of $\log m$ *levels* labeled from 1 to $\log m$, where m is a power of 2. In level i, there are 2^i *core nodes*, and those core nodes are divided into 2^{i-1} *pairs*. The two core nodes in each pair have links with a distinct set of 2^{2i} *fringe nodes*, which we call the pair's *cluster*. Besides links within a certain level, there exist cross-level links. Each core node in level $i-1$ has a link with exactly one fringe node from each of the clusters in level i. The total number of nodes in this network is $n = (4m^3 - 18)/7 + 2m$.

Now consider the running time of LRG for this network. In level i, the span of each core node is $2^{2i} + 2^i$ if $1 \leq i < \log m$, or 2^{2i} if i is $\log m$. Since $(2^{2i} + 2^i)/(2^{2i-2} + 2^{i-1})$ and $2^{2i}/(2^{2i-2} + 2^{i-1})$ are both at least 2, we know that the core nodes in level $i-1$ will not be selected as candidates until all fringe nodes in level i have been covered.

Consider any round at the start of which all nodes in levels $i+1, \ldots, \log m$ have been covered, and where there exists at least one uncovered node in level i. In this round, all the candidates are core nodes at level i, and the support of each uncovered fringe node is 2. Thus, each candidate adds itself to the dominating set with probability $1/2$. For each pair, with probability $3/4$ at least one of the core nodes is selected to be a dominator, and with probability $1/4$ neither is selected. Also, if at least one core node in a pair is selected, then all the fringe nodes in its cluster are covered. If we number the clusters in level i from 1 to 2^{i-1}, then we have the following inequality for $i \geq (\log m)/2$,

$$\Pr[\text{level } i \text{ is covered in } r \text{ rounds}] = \prod_{j=1}^{j=2^{i-1}} (1 - \frac{1}{4^r}) \leq \prod_{j=1}^{j=\sqrt{m}/2} (1 - \frac{1}{4^r}) = (1 - \frac{1}{4^r})^{\sqrt{m}/2}.$$

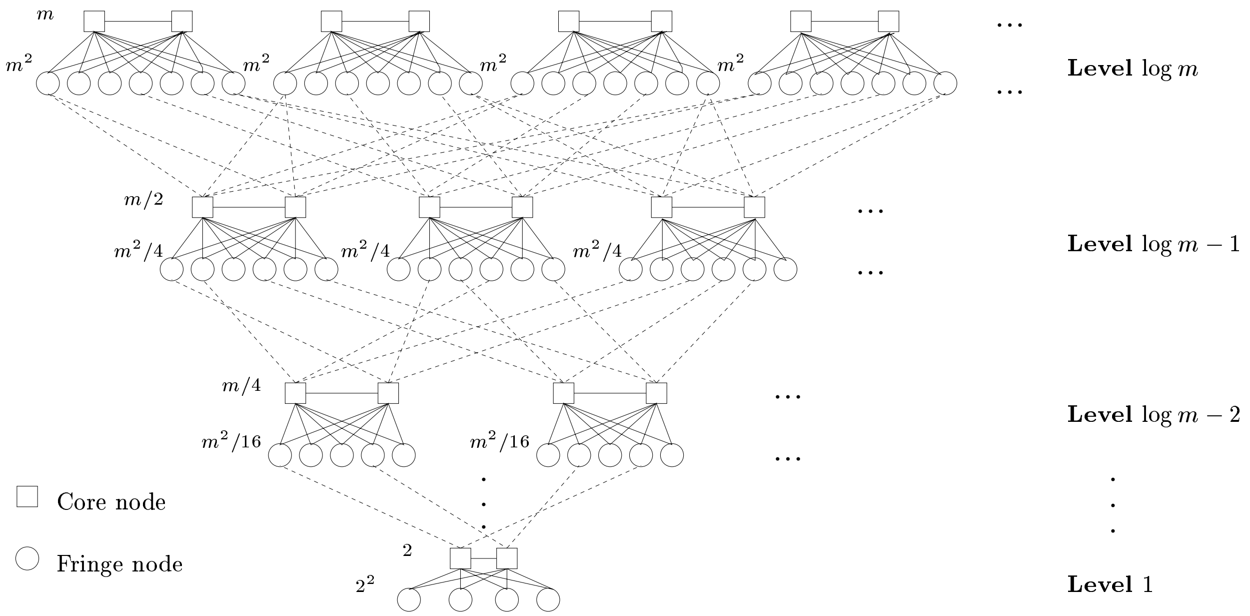

Figure 3: A network for which LRG takes $\Omega(\log n \log \Delta)$ rounds whp.

Thus, the probability that level i, for $i \geq (\log m)/2$, is covered in $r = (\log m)/8$ rounds is at most $(1 - 1/4^{(\log m)/8})^{\sqrt{m}/2}$, which is less than $1/e^{\sqrt[4]{m}/2}$. Therefore, the probability that level i takes at least $(\log m)/8$ rounds to be covered is at least $1 - 1/e^{\sqrt[4]{m}/2}$. Adding over levels $(\log m)/2$ through $\log m$, we obtain that the number of rounds LRG takes for this network is $\Omega(\log^2 m)$ whp. Since the total number of nodes in G is $n = O(m^3)$, and Δ is m^2, the running time of LRG for this network is $\Omega(\log n \log \Delta)$ whp.

4 Generalizations

In this section, we extend the analysis of LRG in three directions. We first analyze, in Section 4.1, a natural variant of LRG that may be of independent interest. Sections 4.2 and 4.3 consider the MMDS problem and MDS with weights, respectively.

4.1 A variant of LRG

In the dominator selection step of each round of LRG, each candidate v adds itself to the dominating set with probability $1/med(v)$, where $med(v)$ is the median support among all nodes in $C(v)$. Another, perhaps more natural, choice for the probability is the average of the inverse of the supports. We consider a modification of LRG, which we refer to as AVE, in which the dominator selection step of each round is the following: each candidate adds itself to the dominating set with probability $(\sum_{u \in C(v)} 1/s(u))/d(v)$. Interestingly, the behavior of AVE is different than that of LRG. A crucial aspect of the proof of Theorem 2 is that the expected ratio of the number of nodes covered to the number of entries in the dominators selected in any round is constant. This property does not hold for AVE. Nevertheless, we can establish an $O(\log n \log \Delta)$ high probability bound on the approximation ratio of AVE.

Theorem 4 AVE *computes a dominating set of size within $O(\log n \log \Delta)$ of the optimal in $O(\log n \log \Delta)$ rounds whp.*

We place a bound on the time complexity by analyzing the progress of the algorithm in any given round as in the analysis of LRG. We define the subgraph H of G as before, and it is easy to see that Lemma 3.1 holds for AVE as well. For a candidate v and an uncovered node u, we define $C(v)$, $T(v)$, $B(v)$, $P_s(v)$, $P_c(u)$, and the notions of good and nice as in Section 3.1. The following lemma is analogous to Lemma 3.3.

Lemma 4.1 *If u is nice, then $P_c(u) > 1 - \frac{1}{\sqrt[8]{e}}$.*

Proof: For any candidate v that is good for u, we have

$$P_s(v) = (\sum_{u \in C(v)} \frac{1}{s(u)})/d(v) \geq (\sum_{u \in B(v)} \frac{1}{s(u)})/d(v) \geq \frac{1}{2med(v)} \geq \frac{1}{2s(u)}.$$

For a nice node u, we thus have

$$\begin{aligned}
P_c(u) &= 1 - \prod_{u \in C(v)} (1 - P_s(v)) \\
&\geq 1 - \prod_{v \text{ is good for } u} (1 - P_s(v)) \\
&\geq 1 - \left(1 - \frac{1}{2s(u)}\right)^{s(u)/4} \\
&> 1 - \frac{1}{\sqrt[8]{e}}.
\end{aligned}$$

■

We then define the potential function Φ as in Section 3.1. Note that Lemma 3.4 holds for AVE; thus, by invoking an argument similar to the one in the proof of Lemma 3.2, we have $E[\Phi'] \leq \Phi(1 - \frac{1}{3}(1 - \frac{1}{\sqrt[8]{e}}))$, which justifies Lemma 3.2 for AVE.

Now we are ready to establish the bound on the running time and approximation ratio of AVE.

Proof of Theorem 4: Given the validity of Lemma 3.2 for AVE, it follows that AVE terminates in $O(\log n \log \Delta)$ rounds whp, in the same manner as LRG.

We now consider the approximation ratio. Fix one round and let $S_1, S_2, ..., S_k$ be the connected components in H with corresponding rounded spans $m_1, m_2, ...m_k$. Let OPT denote the minimum dominating set for the given instance, and let $v_1, v_2, ..., v_{|\text{OPT}|}$ denote the dominators in OPT.

Consider any component S_i for $1 \leq i \leq k$. By the definition of AVE, candidate v adds itself to the dominating set with probability $(\sum_{u \in C(v)} 1/s(u))/d(v)$. We thus obtain that the expected number of candidates in S_i added to the dominating set at the end of the round is

$$\begin{aligned}
\sum_{\text{candidate } v \in S_i} \Big(\sum_{u \in C(v)} \frac{1}{s(u)}\Big)/d(v) &\leq \frac{b}{m_i}\Big(\sum_{\text{candidate } v \in S_i} \sum_{u \in C(v)} \frac{1}{s(u)}\Big) \\
&= \frac{b}{m_i} \sum_{u \in S_i} s(u) \frac{1}{s(u)} \\
&\leq b \cdot \frac{|S_i|}{m_i}.
\end{aligned}$$

We now show that the expected number of dominators selected in any round is $O(|\text{OPT}|)$. For $1 \leq i \leq k$, we associate a cost $1/m_i$ with each element in S_i, and let $W_{ij} = \sum_{u \in S_j, (u, v_i) \in E} 1/m_j$ denote the total cost of all nodes in S_j that have a link with dominator v_i in OPT. We have $\sum_{1 \leq i \leq |\text{OPT}|} W_{ij} \geq |S_j|/m_j$, because OPT is a dominating set. Since the span of v_i, i.e., the number of uncovered nodes it covers at the beginning of this round, is at most the rounded span of any candidate that overlaps with the coverage of v_i, we have $\sum_{1\le j\le k} W_{ij} \le 1$. Thus, we obtain $\sum_{1\le i\le k} |S_i|/m_i \le \sum_{1\le i\le|\text{OPT}|,1\le j\le k} W_{ij} \le |\text{OPT}|$, which gives an upper bound of $b\cdot|\text{OPT}|$ on the expected number of candidates added to the dominating set at the end of any round. Since the number of rounds taken by AVE is $O(\log n \log \Delta)$ whp, it follows that the approximation ratio is $O(b \log n \log \Delta) = O(\log n \log \Delta)$ whp. ■

We note that the lower bound argument on the time complexity of LRG established in Section 3.3 also applies to AVE. A tight bound on the approximation ratio of AVE, however, is still open.

4.2 Dominating sets with multiple coverage

A dominating set of G guarantees that each node in G is covered by at least one node in the dominating set. Certain application domains that are prone to faults and dynamic changes may demand a degree of fault-tolerance by requiring each node to be covered by multiple dominators. Such a requirement can be modeled by the MMDS problem, in which we are given a coverage requirement $r(u)$ for each node u, and the goal is to select a subset D of V such that each node u is covered by $r(u)$ distinct nodes in D. We refer to the desired set D as a *multi-dominating set*. We now describe a distributed algorithm that computes a multi-dominating set of expected size within $O(H_\Delta)$ of the optimal in $O(\log n \log R \log \Delta)$ rounds whp, where R equals $\max_u r(u)$. Our result also extends to a slight generalization of MDS that is equivalent to the Set Multicover problem [28].

The algorithm for MMDS follows the framework of LRG and proceeds in rounds. Let $q(u)$ denote the coverage requirement at the start of a given round, i.e., $q(u)$ equals $r(u)$ minus the number of dominators selected thus far that cover u. We say that a node u is uncovered at the start of a round if $q(u) > 0$. In each round, each node v first calculates its span $d(v)$, where $d(v)$ is defined, as in LRG, as the number of uncovered nodes adjacent to v. The candidate selection step is identical to LRG. We next calculate the *normalized support* $\widehat{s}(u)$ for each node u: $\widehat{s}(u)$ equals $s(u)/q(u)$. Finally, we select a candidate v to be a dominator independently with probability $1/\widehat{med}(v)$, where $\widehat{med}(v)$ is the median of the normalized support among the nodes in $C(v)$.

Theorem 5 *There exists a randomized distributed algorithm for* MMDS *that achieves an approximation ratio of $O(\log \Delta)$ in expectation and $O(\log n)$ whp and a time complexity of $O(\log n \log \Delta \log R)$ whp.*

We define the notions of good nodes and nice nodes as in the analysis of LRG, and also define $C(v)$, $T(v)$, and $B(v)$ as before. Let $q(u)$ (resp., $q'(u)$) denote the coverage requirement at the beginning (resp., end) of a round.

Lemma 4.2 *If u is nice, then $E[q'(u)] \le 3\cdot q(u)/4$.*

Proof: For any candidate v that is good for u, we have $P_s(v) = 1/\widehat{med}(v) \ge 1/\widehat{s}(u)$. Thus, for a nice node u, we have

$$\begin{aligned}
E[q'(u)] &= q(u) - \sum_{u\in C(v)} P_s(v) \\
&\le q(u) - \sum_{v \text{ is good for } u} P_s(v) \\
&\le q(u) - \sum_{v \text{ is good for } u} 1/\widehat{s}(u) \\
&\le q(u) - s(u)/(4\cdot \widehat{s}(u)) \\
&= 3\cdot q(u)/4.
\end{aligned}$$

■

Let the potential Φ be the sum of the terms $\sum_{u \in C(v)} \log(q(u))$, taken over all candidates v with rounded span m, where m is the maximum rounded span. We use the definitions of entry, top entry and nice top entry as in Section 3.1. Lemma 3.4 applies to MMDS in a straightforward manner. We now analyze the progress in any round.

Lemma 4.3 *If Φ and Φ' are the potentials at the start and end of a round, then $E[\Phi'] \leq (1 - \Omega(1/\log R)) \cdot \Phi$.*

Proof: From Lemma 4.2 and Lemma A.1, we have for any nice top entry u, $E[\log q'(u)] \leq \log q(u) - c$ holds for some constant $c > 0$. Combined with Lemma 3.4, we obtain that the expected decrease in Φ is at least $cX/3$, where X is the total number of entries. Thus, we expect a decrease in Φ by a factor of at least $cX/(3X \log R) = c/(3 \log R)$, where R is the maximum coverage requirement. This completes the proof. ■

We are now ready to establish the bound on the running time and approximation ratio of the algorithm.

Proof of Theorem 5: As in the proof of Theorem 1, we divide the running time of the algorithm into phases. A phase consists of a maximal sequence of rounds with the same maximum rounded span. Consider a phase with maximum rounded span m. From Lemma 4.3, $E[\Phi'] \leq d\Phi$, where $d = 1 - c/(3 \log R)$. Let $T(P)$ denote the number of rounds remaining in the phase when the potential at the start of a round is P. We have the following probabilistic recurrence for $T(\cdot)$. For $P > 1$, we have

$$T(P) = a(P) + T(P'), \tag{6}$$

where $a(P) = 1$ for all P, and P' is the random variable denoting the potential at the end of the round. For $P \leq 1$, we have $T(P) = 0$. Note that P is at most $n \cdot \Delta \log R$. Invoking [18, Theorem 1.3], we obtain that the number of rounds with $\Phi \geq 1$ is $O(\log R \log(n\Delta \log R)) = O(\log n \log R)$ whp. When Φ drops to below 1, it follows from the proof of Theorem 1 that the number of rounds with $\Phi < 1$ is $O(\log n)$ whp. Thus, we obtain that the number of rounds in a phase is $O(\log n \log R)$ whp.

The total number of phases is at most the total number of distinct values for the rounded span, which is at most $\log_b \Delta$. Therefore, the running time of the algorithm is $O(\log n \log \Delta \log R)$ whp.

We now consider the approximation ratio. As in Section 3.2, we assign a cost to a node u every time u gets covered, as long as the coverage requirement is unfulfilled. (If node u is covered more than $r(u)$ times in a round, then an arbitrary $r(u)$ of them are assigned costs.) Let $cost_i(u)$, $1 \leq i \leq r(u)$, denote the cost assigned to u each time u is covered. Following a generalization of the argument used in the proof of Lemma 3.5, it can be shown that Lemma 3.5 holds for MMDS, which means that $\sum_{u \in V} \sum_{i=1}^{r(u)} cost_i(u) \leq H_\Delta \cdot |\mathrm{OPT}|$.

Let $c(u) = 1/\widehat{d}(v)$, where $u \in C(v)$. Let $t(u)$ denote the number of candidates v that are added to D at the end of this round such that $u \in B(v)$, and let Z denote the total cost assigned in the round. Following the proof of Lemma 3.6, we obtain that

$$E[|S|] \leq 2b \sum_{u \in V'} c(u) \Pr[t(u) > 0] \cdot E[t(u) | t(u) > 0].$$

Consequently, we have

$$E[|S|] \leq 2b \sum_{u \in V'} c(u) \Big(\sum_{i=1}^{q(u)-1} \Pr[t(u) = i] \cdot E[t(u) | t(u) = i] + \Pr[t(u) \geq q(u)] \cdot E[t(u) | t(u) \geq q(u)] \Big).$$

The dominator selection can be viewed as a sequence of Bernoulli trials, one trial for each candidate; the probability associated with the Bernoulli trial for candidate v is simply $1/\widehat{med}(v)$. Using elementary probability theory, one can show that if X is the number of successes in a sequence of Bernoulli trials, then $E[X|X \geq \ell]$ is at most $\ell + E[X]$. (For the sake of completeness, the preceding claim is established in Lemma A.2.) We thus obtain

$$E[|S|] \leq 2b \sum_{u \in V'} c(u) \Big(\sum_{i=1}^{q(u)-1} \Pr[t(u) = i] \cdot E[t(u)|t(u) = i] + \Pr[t(u) \geq q(u)] \cdot (q(u) + E[t(u)]) \Big)$$

For $E[t(u)]$, we have

$$E[t(u)] = \sum_{u \in B(v)} P_s(v) = \sum_{u \in B(v)} \frac{1}{\widehat{med}(v)} \leq \sum_{u \in B(v)} \frac{q(u)}{s(u)} \leq q(u).$$

Substituting the bound on $E[t(u)]$ in Equation 7, we have

$$\begin{aligned} E[|S|] &\leq 4b \sum_{u \in V'} c(u) \Big(\sum_{i=1}^{q(u)-1} \Pr[t(u) = i] \cdot E[t(u)|t(u) = i] + \Pr[t(u) \geq q(u)] \cdot q(u) \Big) \\ &\leq 4b \cdot E[Z]. \end{aligned}$$

With an argument similar to that in the proof of Theorem 2, Theorem 5 follows. ■

4.3 Weights on the nodes

A natural and useful generalization of MDS associates a weight $w(u)$ with every node u and seeks a dominating set of minimum total weight. LRG generalizes to this weighted version of MDS using the same technique used in the greedy algorithm for the weighted set cover problem. Instead of comparing the rounded span of the nodes, we compare the ratio of the span to the weight of the node. We again round this value, which we refer to as the *normalized span*, to a nearest power of a constant $b > 1$ (allowing negative powers). In the candidate selection phase, a node selects itself as a candidate only if the rounded normalized span is the maximum among all the nodes within a distance of two. The remaining phases in each round are identical to LRG. The same analysis shows that the expected approximation ratio of the algorithm is still $O(\log \Delta)$. Since the number of different values for the rounded normalized span is $O(\log(W\Delta))$, where W is the ratio of the maximum weight to the minimum weight, we obtain that the time complexity of the algorithm is $O(\log(W\Delta)\log n)$ whp.

5 Concluding remarks

In this paper, we have studied simple local algorithms for constructing small dominating sets that we hope will have applications in practical network scenarios. In ongoing experimental work, we are evaluating different dominating set algorithms for the particular application of locating centers in mobile ad-hoc networks. We plan to explore the impact of mobility on the quality of the dominating sets, and to consider algorithms for updating these sets in response to changes in the network.

There are several interesting directions for future theoretical research. In Section 3.3, we have established a bound of $\Omega(\log n \log \Delta)$ on the number of rounds taken by LRG, and this bound also extends to other variants such as AVE. An important open problem is whether there exists a distributed $O(\log n)$-time $O(\log \Delta)$-approximation algorithm for MDS. It will also be interesting to determine the best approximation-time tradeoff achieveable by a *deterministic* distributed algorithm. More specific to our results, we would also like to resolve the asymptotic performance of algorithm AVE.

The dominating set problem and its variants, most notably, the k-dominating set problem, are closely related to facility location problems such as the k-median [24] and k-center [14] problems, which also consider the task of locating nodes within a network. The constraints and the associated objective functions are different, and these problems appear to require more global coordination. We plan to explore the distributed complexity of facility location problems. In this regard, a recent primal-dual approximation algorithm of [15] may provide some leads for an efficient parallel or distributed implementation.

Acknowledgments

We would like to thank Alessandro Panconesi for his valuable comments on an earlier version of the paper. We would also like to thank Aravind Srinivasan for helpful discussions, and the anonymous referees for their comments.

References

[1] N. Alon and J. H. Spencer. *The Probabilistic Method.* Wiley, New York, NY, 1991.

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

[3] B. Awerbuch. Optimal distributed algorithms for minimum-weight spanning tree, counting, leader election and related problems. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 230–240, May 1987.

[4] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network decomposition and locality in distributed computation. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 364–369, October 1989.

[5] J. Bar-Ilan, G. Kortsarz, and D. Peleg. How to allocate network centers. *Journal of Algorithms*, 15:385–415, 1993.

[6] Y. Bartal, J. Byers, and D. Raz. Global optimization using local information with applications to flow control. In *Proceedings of the 38th IEEE Symposium on the Foundations of Computer Science*, pages 303–312, October 1997.

[7] 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, December 1994.

[8] A. Broder, M. Charikar, and M. Mitzenmacher. A derandomization using min-wise independent permutations. In M. Luby, J. Rolim, and M. Serna, editors, *Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science,* Lecture Notes in Computer Science, volume 1518, pages 15–24. Springer-Verlag, 1998.

[9] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[10] R. Cole and U. Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70:32–53, 1986.

[11] U. Feige. A threshold of $\ln n$ for approximating set cover. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 314–318, May 1996.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York, 1979.

[13] A. V. Goldberg, S. A. Plotkin, and G. E. Shannon. Parallel symmetry-breaking in sparse graphs. *SIAM J. Discrete Math.*, 1:434–446, 1989.

[14] D. Hochbaum and D. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10:180–184, 1985.

[15] K. Jain and V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 1–10, October 1999.

[16] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[17] R. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Communications*, pages 85–103. Plenum Press, New York, 1972.

[18] R. Karp. Probabilistic recurrence relations. *Journal of The ACM*, 41:1136–1150, November 1994.

[19] S. Kutten and D. Peleg. Fast distributed construction of k-dominating sets and applications. *Journal of Algorithms*, 28:40–66, 1998.

[20] B. Liang and Z. Haas. Virtual backbone generation and maintenance in ad hoc network mobility management. In *Proceedings of the 2000 IEEE INFOCOM*, March 2000.

[21] N. Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21:193–201, 1992.

[22] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[23] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 448–457, May 1993.

[24] P. Mirchandani. The p-median problem and generalizations. In P. Mirchandani and R. Francis, editors, *Discrete Location Theory*, pages 55–118. Wiley, New York, NY, 1990.

[25] A. Panconesi and A. Srinivasan. The local nature of Δ-coloring and its algorithmic applications. *Combinatorica*, 15:255–280, 1995.

[26] D. Peleg. Distributed data structures: A complexity oriented view. *Proceedings of the 4th International Workshop on Distributed Algorithms*, pages 71–89, September 1990.

[27] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.

[28] 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.

A Technical lemmas

Lemma A.1 *Let X' and X be two random variables. If $0 \leq X' \leq X$ and $E[X'] \leq dX$, then $E[\log X'] \leq \log X - c$, for some constant $c > 0$.*

Proof: From Markov's inequality, we have $\Pr[X' \geq d'X] \leq E[X']/(d'X) \leq dX/(d'X) = d/d'$, where $d < d' < 1$. We thus have

$$\begin{aligned} E[\log X'] &= \Pr[X' \leq d'X]E[\log X' | X' \leq d'X] + \Pr[X' > d'X]E[\log X' | X' > d'X] \\ &\leq (1 - \Pr[X' > d'X])\log(d'X) + \Pr[X' > d'X]\log X \\ &= \log X + (1 - \Pr[X' > d'X])\log d' \\ &\leq \log X + (1 - d/d')\log d' \\ &= \log X - (1 - d/d')\log(1/d'), \end{aligned}$$

which completes the proof of this lemma. ■

Lemma A.2 *If $X_1, X_2, ..., X_n$ are n independent Bernoulli trials and $X = \sum_{i=1}^{n} X_i$, then $E[X | X \geq \ell] \leq \ell + E[X]$ for any nonnegative integer $\ell \leq n$.*

Proof: Let A_i denote one instance of the $N = \binom{n}{\ell}$ combinations of the n results, such that $X_{i_1} = X_{i_2} = ... = X_{i_\ell} = 1$. It is obvious that event $A = \bigcup_{i=1}^{N} A_i$ is identical to the event $X \geq \ell$. Thus, we have

$$E[X|X \geq \ell] = E[X|A] = \sum_{i=1}^{N} \Pr\big[\bigcap_{k=1}^{i-1} \overline{A_k} \cap A_i | A\big] E\big[X| \bigcap_{k=1}^{i-1} \overline{A_k} \cap A_i\big] \tag{7}$$

Now consider $E[X| \bigcap_{k=1}^{i-1} \overline{A_k} \cap A_i]$ for $1 \leq i \leq N$. First, we show that $\Pr[\bigcap_{k=1}^{i-1} \overline{A_k} | X_j = 0] \leq \Pr[\bigcap_{k=1}^{i-1} \overline{A_k} | X_j = 1]$, where $j \neq i_1, i_2, ..., i_\ell$. Let $B_1, B_2, ..., B_k$ denote $\{A_k : 1 \leq k \leq i-1, j \neq k_1, k_2, ..., k_\ell\}$. We thus have

$$\Pr\big[\bigcap_{k=1}^{i-1} \overline{A_k} | X_j = 1\big] \leq \Pr\big[\bigcap_{i=1}^{k} \overline{B_i}\big] = \Pr\big[\bigcap_{k=1}^{i-1} \overline{A_k} | X_j = 0\big].$$

Consequently,

$$\begin{aligned}
\Pr\big[\bigcap_{k=1}^{i-1} \overline{A_k}\big] &= \Pr[X_j = 0] \Pr\big[\bigcap_{k=1}^{i-1} \overline{A_k} | X_j = 0\big] + \Pr[X_j = 1] \Pr\big[\bigcap_{k=1}^{i-1} \overline{A_k} | X_j = 1\big] \\
&\geq (1 - \Pr[X_j = 1]) \Pr\big[\bigcap_{k=1}^{i-1} \overline{A_k} | X_j = 1\big] + \Pr[X_j = 1] \Pr\big[\bigcap_{k=1}^{i-1} \overline{A_k} | X_j = 1\big] \\
&= \Pr\big[\bigcap_{k=1}^{i-1} \overline{A_k} | X_j = 1\big] \qquad (8)
\end{aligned}$$

We then have

$$\begin{aligned}
E\big[X| \bigcap_{k=1}^{i-1} \overline{A_k} \cap A_i\big] &= \ell + \sum_{j \neq i_1, i_2, ..., i_\ell} \Pr\big[X_j = 1 | \bigcap_{k=1}^{i-1} \overline{A_k}\big] \\
&= \ell + \sum_{j \neq i_1, i_2, ..., i_\ell} \frac{\Pr[X_j = 1] \Pr\big[\bigcap_{k=1}^{i-1} \overline{A_k} | X_j = 1\big]}{\Pr\big[\bigcap_{k=1}^{i-1} \overline{A_k}\big]} \\
&\leq \ell + \sum_{j \neq i_1, i_2, ..., i_\ell} \Pr[X_j = 1] \\
&\leq \ell + E[X], \qquad (9)
\end{aligned}$$

where the last step is by inequality 8.

Substituting inequality 9 in equation 7, we have

$$\begin{aligned}
E[X|X \geq \ell] &\leq \big(\sum_{i=1}^{N} \Pr\big[\bigcap_{k=1}^{i-1} \overline{A_k} \cap A_i | A\big]\big)(\ell + E[X]) \\
&= \ell + E[X],
\end{aligned}$$

which completes the proof of Lemma A.2. ■