Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks

Khaled M. Alzoubi Peng-Jun Wan Ophir Frieder
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
Email: {alzoubi, wan, ophir }@cs.iit.edu

Abstract

Connected dominating set (CDS) has been proposed as virtual backbone or spine of wireless ad hoc networks. Three distributed approximation algorithms have been proposed in the literature for minimum CDS. In this paper, we first reinvestigate their performances. None of these algorithms have constant approximation factors. Thus these algorithms can not guarantee to generate a CDS of small size. Their message complexities can be as high as $O\left(n^2\right)$, and their time complexities may also be as large as $O\left(n^2\right)$ and $O\left(n^3\right)$. We then present our own distributed algorithm that outperforms the existing algorithms. This algorithm has an approximation factor of at most 8, $O\left(n\right)$ time complexity and $O\left(n\log n\right)$ message complexity. By establishing the $\Omega\left(n\log n\right)$ lower bound on the message complexity of any distributed algorithm for nontrivial CDS, our algorithm is thus message-optimal.

Keywords: wireless ad hoc networks, distributed algorithm, connected dominating set, independent set, leader election, spanning tree.

1 Introduction

Wireless ad hoc networks can be flexibly and quickly deployed for many applications such as automated battlefield, search and rescue, and disaster relief. Unlike wired networks or cellular networks, no physical backbone infrastructure is installed in wireless ad hoc networks. A communication session is achieved either through a single-hop radio transmission if the communication parties are close enough, or through relaying by intermediate nodes otherwise. In this paper, we assume that all nodes in a wireless ad hoc network are distributed in a two-dimensional plane and have an equal maximum transmission range of one unit. The topology of such wireless ad hoc network can be modeled as a *unit-disk graph* [6], a geometric graph in which there is an edge between two nodes if and only if their distance is at most one (see Figure 1).

Although a wireless ad hoc network has no *physical* backbone infrastructure, a *virtual* backbone can be formed by nodes in a connected dominating set of the corresponding unit-disk graph [1][7][10]. Such virtual backbone, also referred to as *spine*, plays a very important role in routing, broadcasting and connectivity management in wireless ad hoc networks [1]. In general, a *dominating set* (DS) of a graph $G = (V, E)$ is a subset $V' \subset V$ such that each node in $V - V'$ is adjacent to some node in V', and a *connected dominating set* (CDS) is a dominating set which also induces a connected

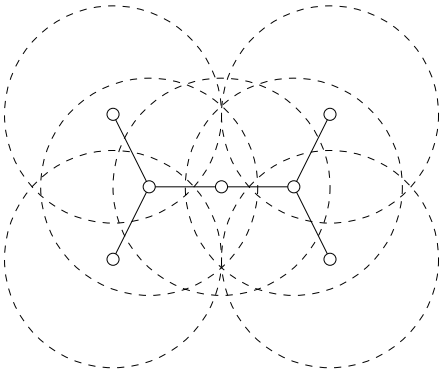

Figure 1: Model the topology of wireless ad hoc networks by unit-disk graphs.

subgraph. A (connected) dominating set of a wireless ad hoc network is a (connected) dominating set of the corresponding unit-disk graph. To simplify the connectivity management, it is desirable to find a minimum connected dominating set (MCDS) of a given set of nodes. However, finding an MCDS in unit-disk graphs is NP-hard [6], and thus only distributed approximation algorithms in polynomial time are practical for wireless ad hoc networks. In this paper, we further show that any distributed algorithm for *nontrivial* CDS requires at least $O(n \log n)$ messages, where n is the number of nodes and the message size is a constant multiple of the number of bits representing the node IDs (a CDS is said to be *trivial* if it consists of all nodes).

Since the networking nodes in wireless ad hoc networks are very limited in resources, a virtual backbone should not only be "thinner", but should also be constructed with low communication and computation costs. In addition, the communication and computation costs should be scalable as the wireless ad hoc networks are typically deployed with large network size. In this paper, we first reinvestigate the performance of the three known distributed approximation algorithms for MCDS, proposed by Das et al. in [1][7][10], by Wu and Li in [12], and by Stojmenovic et al. in [11], respectively. While the first one has a $\Theta(\log n)$ approximation factor, the other two both have $\Theta(n)$ approximation factors. Thus none of them can guarantee to generate a CDS of small size. The algorithms also have very high implementation cost in terms of message complexity and/or time complexity. We thus present our own distributed algorithm that always outputs a nontrivial CDS. This algorithm has an approximation factor of at most 8, $O(n)$ time complexity and $O(n \log n)$ message complexity. As $\Omega(n \log n)$ is a lower bound on the message complexity of any distributed algorithm for nontrivial CDS, our algorithm is thus a message-optimal distributed algorithm for nontrivial CDS.

The remaining of this paper is organized as follows. In Section 2, we establish a $\Omega(n \log n)$ lower bound on the message complexity of any distributed algorithm for nontrivial CDS. In Section 3, Section 4 and Section 5, we analyzes the performances of the three existing algorithms by Das et al. in [1][7][10], by Wu and Li in [12], and by Stojmenovic et al. in [11], respectively. In Section 6, we present a better distributed algorithm and analyze its performance. Finally, we conclude this paper in Section 7.

2 Lower Bound on Message Complexity

In this section, we establish the $\Omega(n \log n)$ lower bound on the message complexity for distributed algorithms for leader election, spanning tree and nontrivial CDS in wireless ad hoc networks. The reduction is made from the following well-known bound on the message complexity of distributed leader election in asynchronous ring networks with point-to-point transmission. A leader election is a process to elect a unique node as the leader by all nodes.

Theorem 1 *[2] In asynchronous rings with point-to-point transmission, any distributed algorithm for leader election sends at least* $\Omega(n \log n)$ *messages.*

Theorem 2 *In asynchronous wireless ad hoc networks whose unit-disk graph is a ring, any distributed algorithm for leader election sends at least* $\Omega(n \log n)$ *messages.*

Proof. Let $\mathcal{A}$ be any distributed algorithm for leader election in wireless ad hoc networks whose unit-disk graph is a ring. Let $\mathcal{A}^*$ be the algorithm by replacing each wireless transmission by two point-to-point transmissions. Then $\mathcal{A}^*$ is a distributed algorithm for leader election in asynchronous rings with point-to-point transmission. Note that the algorithm $\mathcal{A}^*$ sends twice messages of that sent by $\mathcal{A}$. Thus from Theorem 1, $\mathcal{A}$ must also send at least $\Omega(n \log n)$ messages. ■

Theorem 3 *In asynchronous wireless ad hoc networks whose unit-disk graph is a ring, any distributed algorithm for spanning tree sends at least* $\Omega(n \log n)$ *messages.*

Proof. Let $\mathcal{A}$ be any distributed algorithm for spanning tree in wireless ad hoc networks whose unit-disk graph is a ring. Note that any spanning tree of a ring consists of all edges in the ring except one. Thus it has exactly two leaves which are also neighbors. Thus after an spanning tree is completed, the two leaves can exchange a message to select the leader between them according to some symmetry-breaking criterion, for example by their IDs. After the leader is identified, it then notifies all other nodes in linear number of message. Thus from algorithm $\mathcal{A}$, we can derive a distributed algorithm for leader election whose message complexity is $\Theta(n)$ more than the number of messages sent by $\mathcal{A}$. From Theorem 2, the message complexity of $\mathcal{A}$ is at least $\Omega(n \log n)$. ■

A distributed algorithm for leader election in wireless ad hoc networks has been proposed in [5]. This algorithm has message complexity $O(n \log n)$ and therefore is message-efficient. Its actual implementation also constructs a spanning tree rooted at the leader.

Theorem 4 *In asynchronous wireless ad hoc networks whose unit-disk graph is a ring, any distributed algorithm for nontrivial CDS sends at least* $\Omega(n \log n)$ *messages.*

Proof. Let $\mathcal{A}$ be any distributed algorithm for CDS in wireless ad hoc networks whose unit-disk graph is a ring. Note that for any nontrivial CDS of a ring consists of all nodes except either a unique node or two neighboring nodes. So after an nontrivial CDS is completed, the leader can be elected as follows. A dominatee declares itself as the leader if both its neighbors are dominators, or one of its neighbor is a dominatee but has larger ID. The leader then notifies all other nodes in linear number of message. Thus from algorithm $\mathcal{A}$, we can derive a distributed algorithm for leader election whose message complexity is $\Theta(n)$ more than the number of messages sent by $\mathcal{A}$. From Theorem 2, the message complexity of $\mathcal{A}$ is at least $\Omega(n \log n)$. ■

3 Das et al.'s Algorithm

The centralized version of the distributed algorithm proposed by Das et al. consists of three stages. The first stage finds an approximation to Minimum Dominating Set, which is essentially the well-studied Set Cover problem. Not surprisingly, the heuristic proposed by Das et al. in [1][7][10] is a translation of Chvatal's greedy algorithm [4] for Set Cover, and thus guarantees an approximation factor of $H(\Delta)$, where Δ is the maximum degree and H is the harmonic function. Let U denote the dominating set output in this stage. The second stage constructs a spanning forest F. Each tree component in F is a union of stars centered at the nodes in U. The stars are generated by letting each dominatee node pick up an arbitrary neighbor in U. The third stage expands the spanning forest F to a spanning tree T. All internal nodes in T form a CDS. It is easy to show that the CDS generated in this way contains at most $3|U|$ nodes, and therefore is a $3H(\Delta)$-approximation of MCDS.

Figure 2 shows a family of instances for which the size of the solution computed by the above greedy algorithm is larger than the optimum solution by a logarithm factor. All points lie in a rectangle whose horizontal side has length one and whose vertical side has length $2\sqrt{1-\left(\frac{1}{2(k-1)}\right)^2}$ (the selection of the value $2\sqrt{1-\left(\frac{1}{2(k-1)}\right)^2}$ is based on the fact for any two points whose y-coordinates are differed by $\sqrt{1-\left(\frac{1}{2(k-1)}\right)^2}$, they are connected if and only if their x-coordinates are differed by at most $\frac{1}{2(k-1)}$). The two nodes v_1 and v_k are the centers of the left and right vertical sides respectively. The $k-2$ nodes $v_2, v_3, \cdots, v_{k-1}$ are evenly distributed within the line segment between $v_1$and v_k from left to right. The two nodes $u_1$and u_2 are the centers of the two sub-rectangles above and below the segment between $v_1$and v_k respectively. The rest points lie in the two horizontal sides. In each horizontal side, $2^0 = 1$ node lies to the left of (and excluding) the perpendicular bisector of v_1v_2, 2^{k-1} nodes lie to the right of (and excluding) the perpendicular bisector of $v_{k-1}v_k$, and 2^{i-1} nodes lie between (and excluding) the perpendicular bisector of $v_{i-1}v_i$ and the perpendicular bisector of v_iv_{i+1}. Thus, the total number of nodes is

$$n = k + 2 + 2\sum_{i=1}^{k} 2^{i-1} = k + 2^{k+1}.$$

Note that u_1 is adjacent to all nodes lying in the top sub-rectangle, u_2 is adjacent to all nodes lying in the bottom sub-rectangle, and they are adjacent to each other. Thus, $\{u_1, u_2\}$ forms an MCDS. On the other hand, the above greedy algorithm would add $v_k, v_{k-1}, \cdots, v_1$ sequentially to

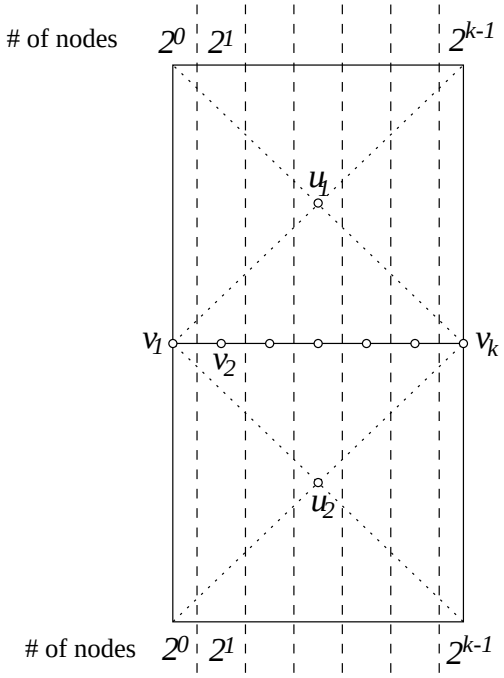

Figure 2: Instance for which the size of the solution output by Das et al.'s algorithm, $\{v_1, v_2, \cdots, v_k\}$, is larger than the optimum solution, $\{u_1, u_2\}$, by a logarithm factor.

the dominating set in the first stage and output the set $\{v_1, v_2, \cdots, v_k\}$ as the CDS at the end of the second stage. This can be proven by induction as follows.

Initially, the degree of node v_i is

$$2 \cdot 2^{i-1} + (k-1) + 2 = 2^i + k + 1;$$

the degrees of the node u_1 and u_2 are both

$$\sum_{i=1}^{k} 2^{i-1} + k + 1 = 2^k + k;$$

and the degree of any other node is

$$\sum_{i=1}^{k} 2^{i-1} - 1 + 1 + 1 = 2^k.$$

So v_k is the first node to be selected. Now we assume that the nodes $v_k, v_{k-1}, \cdots, v_j$ have been added to the dominating set. For any node v_i with $i < j$, the number of its neighbors that have not been dominated yet is $2 \cdot 2^{i-1} = 2^i$; for the node u_1 or u_2, the number of its neighbors that have not been dominated yet is

$$\sum_{i=1}^{j-1} 2^{i-1} = 2^{j-1} - 1;$$

and for any other rest node, the number of its neighbors that have not been dominated yet is

$$\sum_{i=1}^{j-1} 2^{i-1} - 1 = 2^{j-1} - 2.$$

So the node v_{j-1} is then added to the dominating set. Therefore, by induction, the nodes $v_k, v_{k-1}, \cdots, v_1$ are added sequentially to the dominating set. Note that $\{v_1, v_2, \cdots, v_k\}$ is a CDS. The first stage will stop after v_1 is added, and the second stage would add no more nodes.

Since $n = k + 2^{k+1}$ and $\Delta = 2^k + k + 1$, we have $k > \log n - 2$ and $k > \log \Delta - 1$. Therefore, the instance shown in Figure 2 implies the lower bounds $\frac{\log n}{2} - 1$ and $\frac{\log \Delta}{2} - \frac{1}{2}$ on the approximation factor of the greedy algorithm.

The distributed implementation of the above greedy algorithm proposed in [1][7][10] has very high time complexity and message complexity. Indeed, both time complexity and message complexity can be as high as $\Theta\left(n^2\right)$. We also notice that such distributed implementation is technically incomplete. For example, the distributed implementation consists of multiple stages, but the implementation lacks mechanisms to bridge two consecutive stages. Thus, individual nodes have no way to tell when the next stage should begin. While these technical incompleteness are possibly to be fixed, we will not take such effort here as the approximation factor of the greedy algorithm is intrinsically poor.

In summary, we have the following performance results of the distributed algorithm in [1][7][10].

Theorem 5 *The approximation factor of the distributed algorithm proposed by Das et al. in [1][7][10] is between* $\frac{\log \Delta}{2} - \frac{1}{2}$ *and* $3H(\Delta)$. *Both its message complexity and time complexity are* $O\left(n^2\right)$.

4 Wu and Li's Algorithm

While the algorithm proposed by Das et al. first finds a DS and then grow this DS into a CDS, the algorithm proposed by Wu and Li in [12] takes an opposite approach. The algorithm in [12] first finds a CDS and then prune certain redundant nodes from the CDS. The initial CDS U consists of all nodes which have at least two non-adjacent neighbors. A node u in U is considered as *locally redundant* if it has either a neighbor in U with larger ID which dominates all other neighbors of u, or two adjacent neighbors with larger IDs which together dominate all other neighbors of u. The algorithm then removes all locally redundant nodes from U. This algorithm applies only to wireless ad hoc networks whose unit-disk graph is not a complete graph. As indicated in [12], the approximation factor of this algorithm remains unspecified. Obviously, the MCDS of any wireless ad hoc network whose unit-disk graph is not complete graph consists of at least two nodes. On the other hand, any CDS contains at most n nodes. Thus, the approximation factor of the above algorithm is at most $\frac{n}{2}$ where n is the number of nodes. Next, we show that the approximation factor of the above algorithm is exactly $\frac{n}{2}$. This means that the above algorithm does perform extremely poorly over certain instances.

When n is even, we consider the instance illustrated in Figure 3(a). These nodes are evenly distributed over the two horizontal sides of a unit-square. Each node has exactly m neighbors, one in the opposite horizontal side and the rest in the same horizontal side. The two nodes at the left two corners of the unit-square form an MCDS. However, the CDS output by the algorithm in [12] consists of all nodes. Indeed, for each node u, the unique neighbor lying in the opposite horizontal side is not adjacent to all other neighbors of u. Thus, the initial CDS U consists of all nodes. In addition, no single neighbor of a node u can dominate all other neighbors of u. Furthermore, if a pair of neighbors of u are adjacent, they must lie in the same horizontal side as u, and therefore neither of them is adjacent to the unique neighbor of u lying in the opposite horizontal side. So no node is locally redundant. Consequently the output CDS still consists of all nodes.

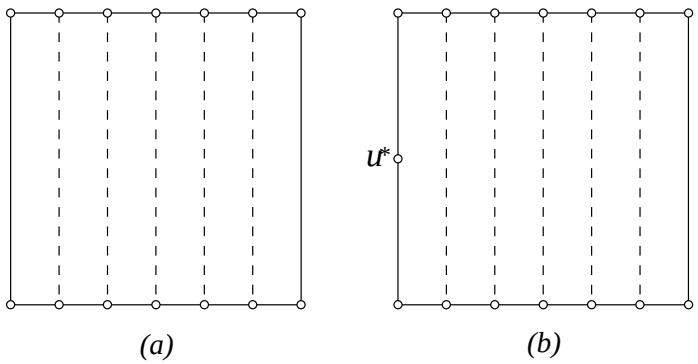

Figure 3: Instance for which the CDS output by Wu and Li's algorithm consists of all nodes but the MCDS consists of only two nodes.

When n is odd, we consider the instance illustrated in Figure 3(b). The node with the largest ID, denoted by u^*, is the center of the left vertical side of a unit-square, and all other $n-1$ nodes are evenly distributed over the two horizontal sides of the unit-square. The two nodes at the left two corners of the unit-square form an MCDS. On the other hand, the CDS output by the algorithm in [12] also consists of all nodes. In fact, following the same argument as in the even case, all nodes other than u^* are in the initial CDS U. The node u^* is also in the initial CDS U as well. Since u^* is not adjacent to the two nodes at the right corners of the unit-square, all nodes other than u^* are not locally redundant. The u^* itself is also not locally redundant as it has the maximum ID. Therefore, the output CDS still consists of all nodes.

The distributed implementation of the above algorithm given in [12] runs in two phases. In the first phase, each node first broadcasts to its neighbors the *entire* set of its neighbors, and after receiving this adjacency information from all neighbors it declares itself as dominator if and only if it has two nonadjacent neighbors. These dominators form the initial CDS. In the second phase, a dominator declares itself as a dominatee if it is locally redundant. Note a dominator can find whether it is locally redundant from the adjacency information of all its neighbors. It is claimed in [12] that the total message complexity is $O(n\Delta)$ and the time complexity at each node is $O(\Delta^2)$. A more accurate message complexity is $\Theta(m)$ where m is the number of edges in the unit-disk graph, as each edge contributes two messages in the first phase. The $O(\Delta^2)$ time complexity, however, is not correct. In fact, in order to decide whether it is locally redundant in the second phase, a node u in the initial CDS may have to examine as many as $O(\Delta^2)$ pairs of neighbors, and for each pair of neighbors, as much as $O(\Delta)$ time may be taken to find out whether such pair

of neighbors together dominates all other neighbors of u. Therefore, the time complexity at each node may be as high as $O\left(\Delta^3\right)$, instead of $O\left(\Delta^2\right)$. Note that m and Δ can be as many as $O\left(n^2\right)$ and $O\left(n\right)$ respectively. Thus, the message complexity and the time complexity of the distributed algorithm in [12] are $O\left(n^2\right)$ and $O\left(n^3\right)$ respectively. The instances shown in Figure 3 do require such complexities.

In summary, we have the following performance results of the distributed algorithm in [12].

Theorem 6 *The approximation factor of the distributed algorithm proposed by Wu and Li in [12] is exactly* $\frac{n}{2}$. *Its message complexity is* $\Theta\left(m\right)$ *and its time complexity is* $O\left(\Delta^3\right)$.

5 Stojmenovic et al.'s Algorithm

In the context of clustering and broadcasting, Stojmenovic et al. [11] presented three *synchronized* distributed constructions of CDS. In each of the three constructions, the CDS consists of two types of nodes: the cluster-heads and the border-nodes. The cluster-heads form a maximal independent set (MIS), i.e., a dominating set in which any pair nodes are non-adjacent. A node is a border-node if it is not a cluster-head and there are at least two cluster-heads within its 2-hop neighborhood. The set of cluster-heads is induced by a ranking of nodes which give rise to a total ordering of all nodes. Three rankings are used: the ID only [8][9], an ordered pair of degree and ID [3], and an order pair of degree and location [11]. The selection of the cluster-heads is given by a *synchronized* distributed algorithm, which can be generalized to the following framework. Initially all nodes are colored white. In each stage of the *synchronized* distributed algorithm, all white nodes which have the lowest rank among all white neighbors are colored black; then all white nodes adjacent to the these black nodes are colored gray; finally the ranks of the remaining white nodes are updated. The algorithm ends when all nodes are colored either black or gray. All black nodes then form the cluster-heads.

Regardless of the choice of the ranking, the algorithms in [11] have a $\Theta\left(n\right)$approximation factor. Such inefficiency stems from the non-selective inclusion of all border-nodes. In fact, if the rank is ID only, Figure 4 shows a family of instances which would imply the approximation factor to be exactly n, the worst possible. In these instances, the node with the largest ID is located at the center of a unit-disk and all other nodes are evenly distributed in the boundary of the unit-disk. After the cluster-heads are selected, all other nodes become border-nodes. Thus the CDS would consist of all nodes. On the other hand, the node at the center dominates all other nodes. If the rank is an ordered pair of degree and ID or an order pair of degree and location, the instances shown in Figure 3 imply that their approximation factors are at least $\frac{n}{2}$.

All algorithms in [11] have $O\left(n^2\right)$ message complexity and $\Omega\left(n\right)$ time complexity. This can be illustrated in the following instance: All n nodes are evenly distributed in an interval of length $n-1$ with two nodes being the endpoints of the interval. The i^{th} node from the left endpoint of interval has ID i (i.e., the IDs increase from left to right).

In summary, we have the following performance results of the distributed algorithm in [11].

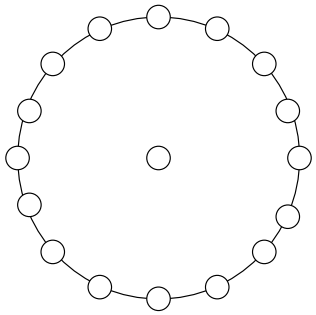

Figure 4: Instance for which the CDS output by Stojmenovic et al.'s algorithm consists of all nodes but the MCDS consists of only one node.

Theorem 7 *The distributed algorithms proposed by Stojmenovic et al. in [11] have an approximation factor of* $\frac{n}{2}$ *or* n*,* $O\left(n^2\right)$ *message complexity, and* $\Omega\left(n\right)$ *time complexity.*

6 A Better Distributed Algorithm

Our distributed algorithm for CDS consists of two phases. These two phases construct a maximal independent set (MIS), and a dominating tree, respectively. They are described and analyzed in the next three subsections.

6.1 MIS Construction

By definition, any pair of nodes in an MIS are separated by at least two hops. However, a subset of nodes in an MIS may be three hops away from the subset of the rest nodes in this MIS. The MIS constructed in this section guarantees that the distance between any pair of its complementary subsets is *exactly two* hops. Our construction uses a carefully chosen rank definition. The ranking is induced by an *arbitrary* rooted spanning tree T, which can be constructed by the distributed leader-election algorithm in [5] with $O\left(n\right)$ time complexity and $O\left(n \log n\right)$ message complexity. Given a rooted spanning tree T, the (tree) level of a node is the number of hops in T between itself and the root of T. (Thus the level of the root is 0.) The rank of a node is then given by the *ordered* pair of its level and its ID. Such ranking gives rise to a total ordering of the nodes in the lexicographic order. The following distributed process enables each node to calculate its own rank and the number of lower-ranked neighbors.

Each node maintains two local metering variables x_1 and x_2. The variable x_1 counts the number of neighbors whose levels have not yet been identified and is thus initialized to the number of neighbors. The variable x_2 counts the number of children who have not yet reported the completion and is thus initialized to the number of children. Each node also maintains a *levelList* that records the levels of its neighbors and is initially empty, and a local variable y which stores the number of lower-ranked neighbors. After the rooted spanning tree T is constructed, the root announces its level 0 by broadcasting a LEVEL message. Upon receiving a LEVEL message, a node appends

an entry consisting of the sender's ID and level to *levelList* and then decrements x_1 by 1. If the sender is its parent in T, it sets its own level to one plus the sender's level, and then announce this level by broadcasting a LEVEL message. If $x_1 = 0$, it sets y to the number of lower-ranked neighbors which can be calculated from *levelList*. If it is a leaf in T (i.e. $x_2 = 0$ initially) and its own level has been determined, it transmits a LEVEL-COMPLETE message to its parent. Upon receiving a LEVEL-COMPLETE message towards itself, a node decrements x_2 by 1; if $x_2 = 0$ after the update and it is not the root, a node transmits a LEVEL-COMPLETE message to its parent and then resets x_2 to the number of children. When the local variable $x_2 = 0$ at the root, the root simply resets x_2 to the number of children. By this time, all nodes know the ranks of its own and all its neighbors and thus the root will move on the construction of the MIS by a color-marking process.

All nodes are initially marked with white color and will be marked with either gray or black eventually. Each node also maintains a *blackList* which records the IDs of its black neighbors. Note that the *blackList* can contain at most five black nodes. The root first marks itself black and broadcasts a BLACK message. Upon receiving a BLACK message, a node adds the sender's ID to *blackList*, and if its color is still white, it marks itself gray and broadcasts a GRAY message which contains its level. Upon receiving a GRAY message, if the rank of the sender is lower than its own, a white node decrements y by 1; if $y = 0$ after the update, it marks itself black and broadcasts a BLACK message. When a leaf node is marked with either gray or black, it transmits a MARK-COMPLETE message to its parent. Upon receiving a MARK-COMPLETE message towards itself, a node decrements x_2 by 1; if $x_2 = 0$ after the update and it is not the root, a node transmits a MARK-COMPLETE message to its parent. By the time when the local variable $x_2 = 0$ at the root, all nodes have been marked with either gray of black and thus the root will move on the construction of the CDS.

Figure 5 illustrates the algorithm for color marking in this phase. In the graph, the IDs of the nodes are labelled beside the nodes, and node 0 is the leader elected in the first phase. The solid lines represent the edges in the rooted spanning tree T, and the dashed lines represents other edges in the unit-disk graph. The ordering of the nodes by rank is given by 0, 4, 12, 2, 5, 8, 10, 3, 6, 9, 11, 1, 7. A possible execution scenario is shown in Figure 5(a)–(g). The nodes 0, 5, 3, and 7 are the black nodes and form a CDS.

The construction of the CDS in the next phase relies on the following property of the black nodes.

Theorem 8 *All black nodes form an MIS and any pair of complementary black subsets are separate by exactly two hops.*

Proof. Let $U = \{u_i : 1 \leq i \leq k\}$ where u_i is the i^{th} node which is marked black. From the construction, any pair of black nodes are not adjacent to each other and thus U is an MIS. For any $1 \leq j \leq k$, let H_j be the graph over $\{u_i : 1 \leq i \leq j\}$ in which a pair of nodes is connected by an edge if and only if their graph distance in G is two. We prove by induction on j that H_j is connected. Since H_1 consists of a single vertex, it is connected trivially. Assume that H_{j-1} is connected for some $j \geq 2$. When the node u_j is marked black, its parent in T must be already marked gray. Thus, there is some node u_i with $1 \leq i < j$ which is adjacent to u_j's parent in T. So (u_i, u_j) is an edge in H_j. As H_{j-1} is connected, so must be H_j. Therefore, H_j is connected for any $1 \leq j \leq k$. ■

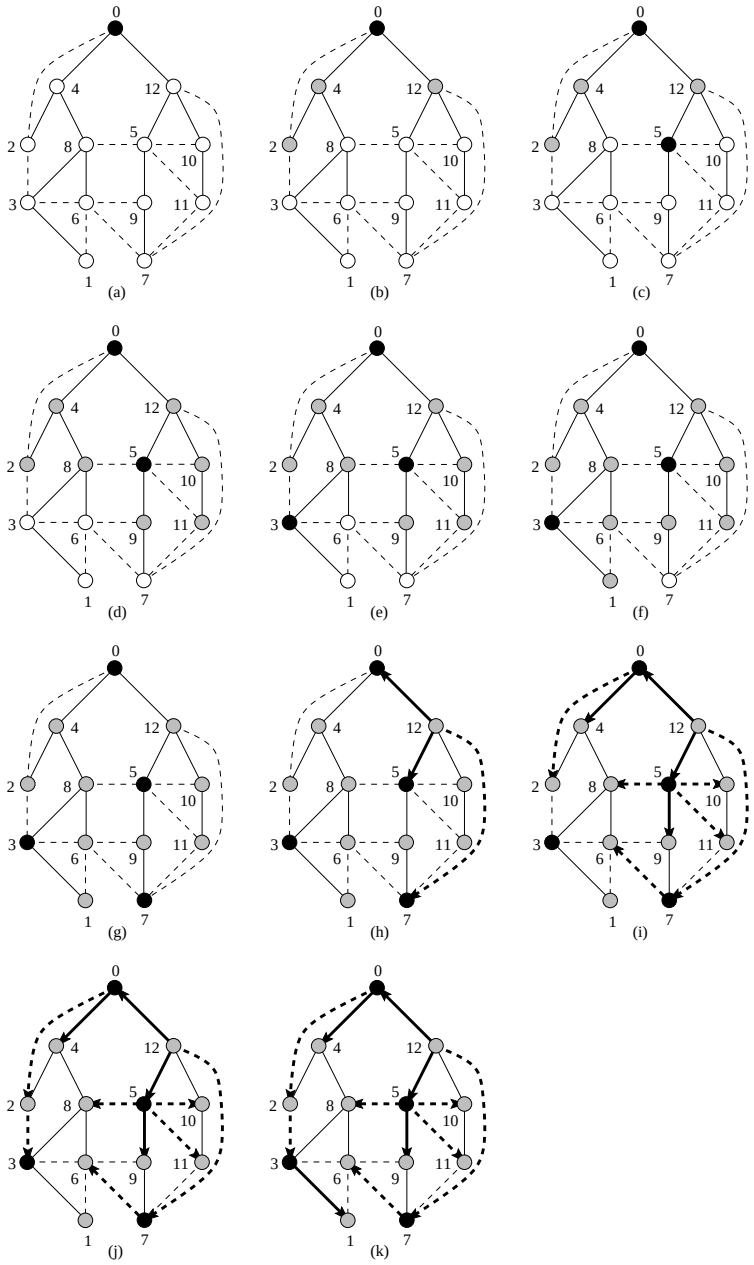

Figure 5: An example of the MIS construction (a)-(g) and dominating tree construction (h)-(k).

6.2 Dominating Tree Construction

The second phase constructs a *dominating tree* T^* whose internal nodes would become a CDS. Each node maintains a local boolean variable z which is initialized to 0 and set to 1 after the node joins the tree T^*. Each node also maintains a local variable *parent* which stores the ID of its parent in T^* and is initially empty, and a *chidrenList* which records the IDs of its children in T^* and is initially empty. The root of T^* is a (gray) neighbor of the root of T which has the largest number of black neighbors. To select the root for T^*, the root of T also maintains a variable *root* and a variable *degree* which is initialized to 0.

The root of T first resets the local variable x_1 to the number of its neighbors and then broadcasts a QUERY message. Upon receiving a QUERY message, a (gray) node transmits to the sender a REPORT message which contains the number of its black neighbors. Upon receiving a REPORT message towards itself, the root of T decrements x_1 by one, and if the number of the black neighbors of the sender is greater than the value of *degree*, it resets *degree* to the number of the black neighbors of the sender and also resets the variable *root* to the ID of the sender. If $x_1 = 0$ after the update, the root of T transmits a ROOT message to the node whose ID is stored in the local variable *root*. Upon receiving the ROOT message towards itself, a node becomes the root of T^*. It sets $z = 1$ and then broadcasts an INVITE2 message. All other nodes join the tree T^* according to the following principle.

- Upon receiving an INVITE2 message, a black node with $z = 0$ sets $z = 1$ and *parent* to the ID of the sender, transmits a JOIN message towards the sender, and then broadcasts an INVITE1 message.
- Upon receiving an INVITE1 message, a gray node with $z = 0$ sets $z = 1$ and *parent* to the ID of the sender, transmits a JOIN message towards the sender, and then broadcasts an INVITE2 message.
- Upon receiving a JOIN message towards itself, a node adds the ID of the sender to *chidrenList*.

Theorem 8 guarantees that whenever there is any black node outside the current T^*, at least one black node would join T^*. Thus eventually all black nodes will join T^*. Consequently, all gray nodes will join T^* eventually. The internal nodes of T^* are then output as the CDS.

Figure 5 (h)-(k) illustrates a possible scenario of the dominating tree construction. The thick links are edges in the dominating tree. The internal nodes 12, 0, 5, 7, 2, 3 form a CDS.

6.3 Performance Analysis

We first analyze the message complexity and time complexity of our distributed algorithm. After the rooted spanning tree T is constructed, the MIS construction in the first phase additionally uses linear messages and takes at most linear time. The construction of the dominating tree T^* also uses linear messages and takes at most linear time. Thus besides the construction of the tree T, our algorithm uses $O(n)$ messages and takes $O(n)$ time. Since the algorithm in [5] used for the construction of T has $O(n \log n)$ message complexity and $O(n)$ time complexity, our algorithm

has $O(n \log n)$ message complexity and $O(n)$ time complexity in overall. Note that the message complexity of our algorithm is dominated by the construction of a rooted spanning tree.

Next we analyze the size of the out CDS, which is the number of internal nodes in T^*. Let OPT be any minimum CDS and let *opt* denote the size of OPT. We begin with the following property of the independent sets.

Lemma 9 *The size of any independent set in a unit-disk graph $G = (V, E)$ is at most $4opt + 1$.*

Proof. Let U be any independent set of V, and let T' be any spanning tree of OPT. Consider an arbitrary preorder traversal of T' given by $v_1, v_2, \cdots, v_{opt}$. Let U_1 be the set of nodes in U that are adjacent to v_1. For any $2 \leq i \leq opt$, let U_i be the set of nodes in U that are adjacent to v_i but none of $v_1, v_2, \cdots, v_{i-1}$. Then $U_1, U_2, \cdots, U_{opt}$ form a partition of U. As v_1 can be adjacent to at most five independent nodes, $|U_1| \leq 5$. For any $2 \leq i \leq opt$, at least one node in $v_1, v_2, \cdots, v_{i-1}$ is adjacent to v_i. Thus U_i lie in a sector of at most 240 degree within the coverage range of node v_i (see Figure 6). This implies that $|U_i| \leq 4$. Therefore,

$$|U| = \sum_{i=1}^{opt} |U_i| \leq 5 + 4\,(opt - 1) = 4opt + 1.$$

This completes the proof. ■

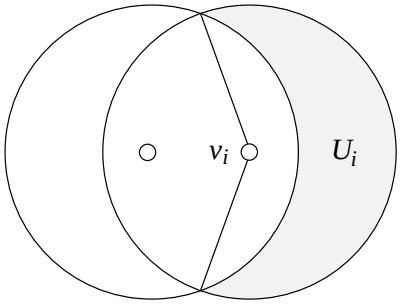

Figure 6: U_i lie in a sector of at most 240 degree within the coverage range of node v_i.

Lemma 9 and its proof implies the following upper-bound on the size of the CDS generated by our algorithm.

Lemma 10 *The number of internal nodes in T^* is at most $8opt - 2$.*

Proof. If there is a black node in OPT, then following the similar proof to Lemma 9 we can show that the total number of black nodes is at most

$$1 + 4\,(opt - 1) = 4opt - 3.$$

Since each internal gray node in T^* has at least one black child, the total number of internal gray nodes in T^* is no more than the number of black nodes. Thus the total number of internal nodes in T^* is at most

$$2\,(4opt - 3) = 8opt - 6.$$

Now we assume that no black node is in OPT. Let k be the number of black nodes adjacent to the root of T^*. Then $k \leq 5$, and following the similar proof to Lemma 9 we can show that the total number of black nodes is at most $k + 4\,(opt - 1)$. Note that the root of T^* has exactly k black children and any internal gray node other than the root of T^* has at least one black child. Thus the total number of internal gray nodes in T^* other than the root of T^* is at most $4\,(opt - 1)$. So the number of internal nodes in T^* is at most

$$\begin{aligned} &4\,(opt - 1) + k + 1 + 4\,(opt - 1) = 8opt - 7 + k \\ &\leq 8opt - 7 + 5 = 8opt - 2. \end{aligned}$$

Thus the lemma is true in either case. ■

In summary, we have the following performance results of our distributed algorithm.

Theorem 11 *Our distributed algorithm has an approximation factor of at most* 8, $O\,(n)$ *time complexity, and* $O\,(n \log n)$ *message complexity.*

7 Conclusion

In this paper, we have established a $\Omega\,(n \log n)$ lower bound on message complexity of any distributed algorithm for nontrivial CDS. We then reinvestigated three known distributed approximation algorithms for MCDS. After that we presented our own algorithm. The performance comparison of these four algorithms is listed in Table 1. From this table, we can conclude that our algorithm outperforms the existing algorithms.

	[1][7][10]	[12]	[11]	This paper
Approx. factor	$\Theta\,(\log n)$	n	$\frac{n}{2}, n$	≤ 8
Msg. complexity	$O\left(n^2\right)$	$\Theta\,(m)$	$O\left(n^2\right)$	$O\,(n \log n)$
Time complexity	$O\left(n^2\right)$	$O\left(\Delta^3\right)$	$\Omega\,(n)$	$O\,(n)$
Nontrivial	Yes	No	No	Yes

Table 1: Performance Comparison.

References

[1] V. Bharghavan and B. Das, "Routing in Ad Hoc Networks Using Minimum Connected Dominating Sets", *International Conference on Communications'97*, Montreal, Canada, June 1997, pp. 376-380.

[2] J. Burns, "A Formal Model for Message Passing Systems", *Technical Report TR-91*, Computer Science Department, Indiana University, May 1980.

[3] G. Chen and I. Stojmenovic, "Clustering and routing in wireless ad hoc networks", *Technical Report TR-99-05*, Computer Science, SITE, University of Ottawa, June 1999.

[4] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operation Research*, 4(3):233–235, 1979.

[5] I. Cidon and O. Mokryn, "Propagation and Leader Election in Multihop Broadcast Environment", *12th International Symposium on DIStributed Computing (DISC98)*, September 1998, Greece. pp.104–119.

[6] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit Disk Graphs", *Discrete Mathematics*, 86:165–177, 1990.

[7] B. Das, R. Sivakumar, and V. Bharghavan, "Routing in Ad-Hoc Networks Using a Spine", *International Conference on Computers and Communications Networks '97*, Las Vegas, NV. September 1997.

[8] M. Gerla, and J. Tsai, "Multicluster, mobile, multimedia radio network", *ACM-Baltzer Journal of Wireless Networks*, Vol.1, No.3, pp.255-265(1995).

[9] C.R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 7, Sept. 1997, pp. 1265-1275

[10] R. Sivakumar, B. Das, and V. Bharghavan, "An Improved Spine-based Infrastructure for Routing in Ad Hoc Networks", *IEEE Symposium on Computers and Communications '98*, Athens, Greece. June 1998.

[11] I. Stojmenovic, M. Seddigh, J. Zunic, "Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks", *Proc. IEEE Hawaii Int. Conf. on System Sciences*, January 2001.

[12] J. Wu, and H.L. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks", *Proceedings of the 3rd ACM international workshop on Discrete algorithms and methods for mobile computing and communications*, 1999, Pages 7–14.