Journal of Global Optimization (2006) 35: 111–119
DOI 10.1007/s10898-005-8466-1

Improving Construction for Connected Dominating Set with Steiner Tree in Wireless Sensor Networks

MANKI MIN[1], HONGWEI DU[2], XIAOHUA JIA[2], CHRISTINA XIAO HUANG[3], SCOTT C.-H. HUANG[1] and WEILI WU[4]
[1]*Department of Computer Science and Engineering University of Minnesota, Minneapolis MN 55455, USA (e-mail: {min, huang}@cs.umn.edu)*
[2]*Department of Computer Science, City University of Hong Kong, Kowloon Tong, Hong Kong (e-mail: {hongwei, jia}@cs.cityu.edu.hk)*
[3]*3M Center, Building 0235-03-F-08, St. Paul MN 55144, USA (e-mail: xchuang@mmm.com)*
[4]*Department of Computer Science, University of Texas at Dallas, Richardson, Tx 75083 (e-mail: weiliwu@utdallas.edu)*

(Received 6 June 2005; accepted in revised from 6 June 2005)

Abstract. The connected dominating set plays an important role in *ad hoc* wireless networking. Many constructions for approximating the minimum connected dominating set have been proposed in the literature. In this paper, we propose a new one with Steiner tree, which produces approximation solution within a factor of 6.8 from optimal. This approximation algorithm can also be implemented distributedly.

1. Introduction

Wireless Sensor Network is widely applied in the healthcare industry, food industry, and agriculture. The sensors in the network are incorporated with integrated circuits to provide networking capability, so they are also referred as smart sensors [14]. Wireless sensor network essentially is an *ad hoc* wireless network which is composed of many sensors. It inherits the characteristics of *ad hoc* wireless networking, It is an autonomous system consisting of mobile hosts connected by wireless links. There is no central administration in the network and the network is hardware-infrastructureless. In the wireless sensor network, each sensor is not only a mobile host but also a router. In other words, the sensors are able to forward the received data packages according to routing protocols. In this paper, we assume that all sensors have the same power, that is, every sensor can communicate with others within a unit distance. Under this assumption, the topology of the sensor network can be formulated as a unit disk graph.

A unit disk is a disk with radius one. A *unit disk graph* is associated with a set of unit disks in the Euclidean plane. Each node is the center of a unit disk. An edge exists between two nodes u and v if and only if $|uv| \leqslant 1$ where $|uv|$ is the Euclidean distance between u and v. This means that two nodes connecting with an edge if and only if u's disk covers v and v's disk covers u.

Multicasting is to send messages to a group of receivers at the same time. For example, when one sensor wants to send out topology update information to a group of other sensors in the network, it will use multicasting. Multicasting reduces the network traffic by combining multiple unicast data stream into one. The advancement of multicasting in the network is driven by emerging applications, such as net meeting and video conference. Currently, IP multicast group, where each multicast group uses one IP address, and MBone, which is the multicast backbone, are employed in the wired network. [7, 9]. Because of the different transmission media between the wired network and wireless network, the multicasting protocols in the wired network can not be applied in the wireless network.

One of efficient ways to support multicasting is to use virtual backbone in wireless networks. A virtual backbone is a *connected dominating set* in the network, that is, it is a subset of sensors such that they form a connected sub-network and every sensor is either in the subset or adjacent to a sensor in the subset.

Since multicasting can be performed first within virtual backbone and then to others, it has been recommended to manage and updating the topology of virtual backbone instead of the topology of the whole network, which reduces both storage and message complexities.

Clearly, the smaller virtual backbone gives the better performance. However, computing the minimum connected dominating set is NP-hard even in unit disk graphs. Therefore, many efforts [2, 15, 16, 18, 17, 1, 4] have been made to design approximations or heuristics for the minimum connected dominating set.

Guha and Khuller [10] showed a two-stage greedy ($\ln \Delta + 3$)-approximation for the minimum connected dominating set in general graphs where Δ is the maximum degree in the graph. They also gave a lower bound ($\ln \Delta + 1$) for any polynomial-time approximation for the minimum connected dominating set provided $NP \not\subseteq 2^{polylog(n)}$, Ruan et al. [13] found a one-stage greedy ($\ln\Delta + 2$)-approximation.

Cheng et al. [6] showed the existence of a polynomial-time approximation scheme for the minimum connected dominating set in unit disk graphs. This means that theoretically, the performance ratio for polynomial-time approximation can be as small as $1+\varepsilon$ for any positive number ε. However, its running time grows rapidly as ε goes to 0 and hence is not worth implementing in practice.

Among implemented approximation for the minimum connected dominating set in unit disk graphs, the best previously known performance ratio is 8 [17, 5] . In this paper, we will present a 6.8-approximation, which can also be implemented distributedly.

It is a popular idea to construct a connected dominating set in two steps: In the first step a dominating set is constructed; in the second step, connect the dominating set into a connected dominating set. Our main idea is to employ the Steiner tree to do the job in the second step, A *Steiner tree* for a given subset of nodes, called *terminals*, in a graph is a tree interconnecting all terminals such that every leaf is a terminal. Every node other than terminals in the Steiner tree is called a *Steiner node*. Clearly, we prefer the smaller number of Steiner nodes in order to obtain smaller connected dominating set. Therefore, we will study the following Steiner tree problem in the unit disk graphs.

Steiner Tree with Minimum Number of Steiner Nodes (ST-MSN): Given a unit disk graph G and a subset P of nodes, compute a Steiner tree for P with the minimum number of Steiner nodes.

The ST-MSN problem in unit disk graph has not been studied very much in the literature. However, its geometric version in the Euclidean plane has been studied extensively [11, 3, 8]. While some results in the Euclidean plane can be extended to unit disk graph, some cannot be done. For example, two points with distance 2 can be connected with a Steiner point in the Euclidean plane. But, two nodes with distance 2 may not be able to be connected by a Steiner node since such a node may not exist. Fortunately, a 3-approximation for ST-MSN can be extended from the the Euclidean plane to unit disk graphs with a quite different proof, which becomes a fundamental part in our approximation algorithm.

2. Approximation with Steiner Trees

Our algorithm consists of two steps. At the first step, we construct a maximal independent set. It is well known that every maximal independent set is also a dominating set. The following is a recent result in (WeiliWu et al., submitted) about relation between the size of the maximal independent set and the minimum connected dominating set in unit disk graphs.

LEMMA 1. *In any unit disk graph, the size of every maximal independent set is upperbounded by $3.8opt+1.2$ where opt is the size of minimum connected dominating set.*

Especially, Wan [17] and Cheng [5] constructed maximal independent set having the following property.

LEMMA 2. *Every subset of the maximal independent set is two hops away from its complement.*

We assume throughout this paper that the maximal independent set we talk about satisfies Lemma 2.

At the second step, we employ a 3-approximation for the ST-MSN to interconnect the maximal independent set. Note that the size of optimal solution for the ST-MSN cannot exceed the size of the minimum connected dominating set since the latter can also interconnect the maximal independent set. Therefore, we spend at most $3opt$ Steiner nodes in the second step, By Lemma 1, the resulting connected dominating set would have size bounded by $6.8opt$.

THEOREM 1. *The two step algorithm with Steiner tree produces 6.8-approximation for the minimum connected dominating set.*

Now, let us describe algorithm in the second step.

ALGORITHM A. Input a maximal independent set and mark all its nodes in black and others in grey. In the following, we will change some grey nodes to black in certain rules. A *black component* is a connected component of the subgraph induced by black nodes.

Stage 1. **while** there exists a grey node adjacent to at least three
black components **do**
change its color from grey to black;
end-while;
Stage 2. **while** there exists a grey node adjacent to at least two
black components **do**
change its color from grey to black;
end-while;
return all black nodes.

We know that Theorem 1 follows immediately from the following.

THEOREM 2. *Let T^* be art optimal tree for the ST-MSN problem on an input maximal independent set. Then the number of grey nodes changed their color to black is at most . $C(T^*)$, the number of Steiner nodes in T^*.*

The remainder of this section is contributed to the proof of Theorem 2. First, we show some properties of optimal trees for the ST-MSN problem.

Since a unit disk graph is placed in the Euclidean plane, its edges have Euclidean lengths. An optimal solution for the ST-MSN is said to be *shortest* if its total edge-length reaches the minimum among all optimal solution.

Any shortest optimal solution T for the ST-MSN in the unit disk graph must have the following properties.

(a1) No two edges cross each other.
(a2) Two edges meet at a node with an angle of at least 60°.
(a3) If two edges meet with an angle of exactly 60°, then they have the same length.

Indeed, if anyone of the above three condition does not hold, then we can easily to find another optimal tree with shorter length.

Note that if a Steiner tree has a terminal with degree more than one, then we can decompose it into smaller trees. Resulting subtrees from decomposition at all terminals with degree more than one are called *full components*.

LEMMA 3. *There exists an optimal tree for the ST-MSN on maximal independent set with the following properties:*

(b1) Each Steiner node having degree at most five, and
(b2) Every full component contains either only one Steiner node of degree two or no Steiner node of degree two.

Proof. First, we show that actually, there exists a shortest optimal tree for the ST-MSN problem such that every node has degree at most five. In fact, consider a shortest optimal tree T for the ST-MSN problem. By (a2) , every node has degree at most six. Suppose T has a node u with degree exactly six. By (a2), every angle at u equals 60°. By (a3), all edges incident to u have the equal length. Consider two adjacent nodes v, w of u such that $\angle vuw = 60^\circ$. Replacing edge vu by vw results in still a shortest optimal solution. Now, the degree of u is reduced. But, the degree of v is increased. Can the degree of v still be at most five? We next show that the answer is yes.

Consider any node v with degree d. We claim that if v is adjacent to k node with degree six, then $d \leqslant 6-2k$. In fact, suppose u with degree six is adjacent to v. Then u has two edges uw and ux such that $\angle wuv = \angle vux = 60^\circ$ and $|uv| = |uw| = |ux|$. Thus, $|vw| = |uw|$ and $|vx| = |ux|$. Replacing uw and ux by vw and vx results in still a shortest optimal tree for ST-MSN. But, v gets two more edges. For all nodes with degree six and adjacent to v, perform the same operation. We will obtain a shortest optimal tree for ST-MSN such that v has degree $d+2k$. Hence, $d+2k \leqslant 6$.

This claim guarantees that moving only one edge from u with degree six to its adjacent node v would still keep v to have degree at most four, Therefore, there exists an optimal tree T having all nodes with degree at most five.

Suppose T has a full component containing both Steiner node with degree two and Steiner node with degree more than two. Removal of a Steiner node with degree two breaks T into two part. By Lemma 2, we can add a Steiner node between two terminals to reconnect the two part. Resulting tree is still optimal, but the number of full components is increased.

Moreover, an optimal tree cannot have a full component containing only Steiner nodes with degree two and more than one Steiner nodes with degree two. Indeed, if such case Occurs, then removal of all Steiner nodes in the full component breaks the tree into two connected parts. By Lemma 2, we need to add only one Steiner node to reconnect the two parts, contradicting the optimality.

Therefore, among optimal trees of nodes with degree at most five, the one with maximum number of full components must have properties (b1) and (b2). □

A Steiner tree is called a *steinerized spanning tree* if all Steiner nodes have degree two. A steinerized spanning tree is minimum if the number of Steiner nodes reaches the minimum. By Lemma 2, in a minimum steinerized spanning tree, each Steiner node is between two terminals. If the number of terminals is n, then the number of Steiner nodes in any minimum steinerized spanning tree is $n-1$.

Now, we are ready to prove Theorem 2.

Proof of Theorem 2. Let us add a tree T_A in Algorithm A. Initially, T_A is an empty tree with all black nodes. When a grey node v becomes black, we add to T_A a star with center v and edges connecting adjacent black connected components.

Denote by $T^{(i)}$ the T_A at the beginning of Step i in the algorithm A. Suppose $T^{(2)} - T^{(1)}$ contains k_3 3-stars, k_4 4-stars, and k_5 5-stars. Then

$$C(T_A) \leqslant C(T) - 2k_3 - 3k_4 - 4k_5$$

where T can be any minimum steinerized spanning tree on all terminals, especially, can be the one constructed as follows. Let T^* be an optimal tree for ST-MSN with properties (b1) and (b2). Suppose T^* has g full components $T_1, T_2, \ldots, T_g$ that each contains a Steiner node with degree more than two. For every T_j, $j = 1, 2, \ldots, g$ we remove an edge between a terminal and a Steiner node. If the tree is broken into two parts, then we add a Steiner node to reconnect them by Lemma 2, Let s_i denote the number of Steiner nodes with degree i in T_j. Then the number of terminals in T_j is

$$3s_5 + 2s_4 + s_3 + 2.$$

Note that removal of the last edge between a terminal and a Steiner node would not break the tree into two parts. Therefore, this operation would add $3s_5+2s_4+s_3+1$ Steiner nodes to replace T_j. If $s_4>0$ or $s_3>0$, then $3s_5+2s_4+s_3 \leqslant 3(s_5+s_4+s_3)=3C(T_j)$ where $C(T_j)$ is the number of Steiner nodes in T_j. Therefore, if T^* has h full components each containing only Steiner nodes with degree five, then

$$C(T) \leqslant 3C(T^*)+h.$$

Hence,

$$C(T_A) \leqslant 3C(T^*)+h-k_3-2k_4-3k_5.$$

It suffices to show $h \leqslant k_3+2k_4+3k_5$.

Let n be the number of terminals. Note that $T^{(1)}$ is the empty graph on all terminals. Hence, $T^{(2)}$ has $q(=n-2k_3-3k_4-4k_5)$ connected components $C_1, C_2, \ldots, C_q$. Now, we construct a graph H in the following ways: Initially, H is the empty graph on all terminals. Let $T_1, T_2, \ldots, T_h$ be the h full components of T^*; each contains only Steiner nodes with degree five. If T_j, $j=1,2,\ldots,h$, has only one Steiner node, then this Steiner node connects to five terminals which must lie in at most two C_i's. Hence, among them there are three pairs of terminals; each pair lie in the same C_i. Connect the three pairs into three edges and put them into H. If T_j has at least two Steiner nodes, then there must exist at least two Steiner nodes each connecting to four terminals. We can also find three pairs of terminals among them such that each pair lies in the same C_i. Connect the three pairs into three edges and put them into H. An important observation is that H cannot contain a cycle because, if it does, so does T^*, a contradiction. Therefore, H has exactly $n-3h$ connected components. Since every connected component of H is contained by a C_i, we have $n-2k_3-3k_4-4k_5 \leqslant n-3h$. Therefore, $h \leqslant (2k_3+3k_4+4k_5)/3 \leqslant k_3+2k_4+3k_5$. □

3. Distributed Implementation

There already exist several distributed algorithms for computing a maximal independent set in the literature [17, 5]. Therefore, we only describe a distributed implementation of Algorithm A.

Each black node carries a z-value which is an identification for black component, that is, all black nodes with the same z-value form a black component. Initially, the z-value of each black node equals its ID.

Grey nodes are ranked based on two values. The first one is y-value which is the number of black components adjacent to it. The second one is its ID. The node with larger y-value is ranked higher. If two grey nodes with the same y-value, then the one with smaller ID is ranked higher.

A grey node u is a *competitor* of another grey node v if u and v are either adjacent each other or adjacent to a same black component. A grey node u is going to change its color to black if and only if u is ranked higher than every competitor of u.

Every grey node keeps two lists, a black list and a competitor list. The black list contains all adjacent black nodes with their z-values, which enable the grey node to compute its y-value.

The competitor list contains all its competitors and their black lists so that each grey node can also compute the y-value of every competitor of it, which enable the grey node to make a decision on whether it should change color nor not.

When a grey node u changes its color to black, all its adjacent black components are connected into one and hence their z-value should be updated to the same one, say the smallest one among them. Meanwhile, all competitors of u become competitors of every competitor of u. Therefore, the competitor list of each competitor of u should also be updated. So, after u changed its color, u would send an UPDATE(u) message to all its neighbors. The message contains u' ID and its two lists.

When a black node v receives UPDATE(u) message, it will update z, send out a COMPLETE(u) and pass UPDATE(u) to its neighbors other than nodes which already sent to v UPDATE(u) or COMPLETE(u).

When a grey node receives UPDATE(u), it updates both black and competitor lists and sends out COMPLETE(u) to its neighbors.

Acknowledgement

Support in part by National Science Foundation under grant ACI0305567.

References

1. Alzoubi, K. M., Wan, P. -J., and Frieder, O. (2002) Message-optimal connected dominating sets in mobile *ad hoc* networks, *Proceedings of the 3rd ACM International Symposium on Mobile ad hoc Networking and Computing*, June 09–11, Lausanne, Switzerland.
2. Bharghavan, V. and Das, B. Routing in *ad hoc* networks using minimum connected dominating sets, *International Conference on Communication*, Montreal, Canada, June 1997.
3. Chen, D., Du, D.-Z., Hu, X., Lin, G.-H., Wang, L. and Xue, G. (2000), Approximations for Steiner trees with minimum number of Steiner points, *Journal of Global Optimization* 18, 17–33.
4. Chen, Y. P. and Liestman, A. L. (2002), Approximating minimum size weakly connected dominating sets for clustering mobile *ad hoc* networks, *Proceedings of the third ACM International Symposium on Mobile ad hoc Networking and Computing,* June 09–11, Lausanne, Switzerland.
5. Cadei, M., Cheng, M.X., Cheng, X. and Du, D.-Z. (2002), Connected domination in *ad hoc* wireless networks, *Proceedings the Sixth International Conference on Computer Science and Informatics* (CS&I'2002).

6. Cheng, X., Huang, X., Li, D., Wu, W. and Du, D.-Z, (2003), A polynomial-time approximation scheme for minimum connected dominating set in *ad hoc* wireless networks, *Networks* 42, 202–208.
7. Deering, S., Farinacci, D., Jacobson, V., Lui, C. and Wei, L. (1994), An Architecture for Wide Area Multicast Routing, *Proceedings of ACM SIGCOMM 1994,* September 1994.
8. Du, D. -Z., Wang, L. and Xu, B. (2001), The Euclidean Bottleneck Steiner Tree and Steiner Tree with Minimum Number of Steiner Points, *COCOON' 2001*, pp. 509–518
9. Eriksson, H. (1994), MBone: The Multicast Backbone, *Communications of the ACM,* August 1994.
10. Guha, S. and Khuller, S. (1998), Approximation algorithms for connected dominating sets. *Algorithmica* 20(4), 374–387.
11. Lin, G.-H. and Xue, G.L. (1999), Steiner tree problem with minimum number of Steiner points and bounded edge-length, *Information Processing Letters* 69, 53–57.
12. Ramamurthy, B., Iness J. and Mukherjee, B. (1997), Minimizing the number of optical amplifiers needed to support a multi-wavelength optical LAN/MAN, *Proceeding IEEE INFOCOM'97*, pp. 261–268, April 1997.
13. Ruan, L., Du, H., Jia, X., Wu, W., Li, Y., and Ko, K. -I. (2004), A greedy approximation for minimum connected dominating set, *Theoretical Computer Science* 329(1–3), 325–330.
14. Salhieh, A., Weinmann, J., Kochha, M. and Schwiebert, L. (2001), Power Efficient topologies for wireless sensor networks, *ICPP'2001*, pp. 156–163.
15. Sivakumar, R., Das, B. and Bharghavan, V. (1998), An improved spine-based infrastructure for routing in *ad hoc* networks, *IEEE Symposium on Computer and Communications*, Athens, Greece, June, 1998.
16. Stojmenovic, I., Seddigh, M. and Zunic, J. (2001), Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks, *Proceeding IEEE Hawaii International Conference on System Sciences*, January 2001.
17. Wan, P.-J., Alzoubi, K.M. and Frieder, O. (2002), Distributed construction of connected dominating set in wireless *ad hoc* networks, *Proceedings of Infocom'2002*.
18. Wu J. and Li, H.L. (1991), 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*, pp. 7–14.