

Recyclable Connected Dominating Set for Large Scale Dynamic Wireless Networks

Donghyun Kim¹, Xianyue Li², Feng Zou¹, Zhao Zhang^{3,*}, and Weili Wu^{1,**}

¹ Department of Computer Science, University of Texas at Dallas,
Richardson, TX, 75080

{donghyunkim,phenix.zou}@student.utdallas.edu, weiliwu@utdallas.edu

² School of Mathematics and Statistics, Lanzhou University,
Lanzhou, Gansu, P.R. China, 730000

lixianyue@lzu.edu.cn

³ College of Mathematics and System Sciences, Xinjiang University, Urumqi,
Xinjiang, P.R. of China, 830046

zhzhao@xju.edu.cn

Abstract. Many people studied the Minimum Connected Dominating Set (MCDS) problem to introduce Virtual Backbone (VB) to wireless networks. However, many existing algorithms assume a static wireless network, and when its topology is changed, compute a new CDS all over again. Since wireless networks are highly dynamic due to many reasons, their approaches can be inefficient in practice. Motivated by this observation, we propose Recyclable CDS Algorithm (RCDSA), an efficient VB maintenance algorithm which can handle the activeness of wireless networks. The RCDSA is built on an approximation algorithm CDS-BD-C1 by Kim et. al. [1]. When a node is added to or deleted from current graph, RCDSA recycles current CDS to get a new one. We prove RCDSA's performance ratio is equal to CDS-BD-C1's. In simulation, we compare RCDSA with CDS-BD-C1. Our results show that the average size of CDS by RCDSA is similar with that by CDS-BD-C1 but RCDSA is at least three times faster than CDS-BD-C1 due to its simplicity. Furthermore, at any case, a new CDS by RCDSA highly resembles to its old version than the one by CDS-BD-C1, which means that using RCDSA, a wireless network labors less to maintain its VB when its topology is dynamically changing.

1 Introduction

Ephremides et. al. first tried to introduce a backbone-like structure in wireless networks [2]. Guha and Kuller firstly used Minimum Connected Dominating Set (MCDS) problem in general graphs to model the problem of computing a minimum size Virtual Backbone (VB) in heterogenous wireless network [3]. Since

* This work is supported in part by the NSFC under grant 60603003 and XJEDU.

** This work is supported in part by the NSF under grant CCF-0514796, CCF-0627233, and CCF-0750992.

a smaller size VB is expected to cause less control messages and suffer less from interference, the size of CDS is served as a major quality factor in many previous works [3,4,5,6,7]. Since computing an MCDS is a well-known NP-hard problem, and thus all of existing work present approximation algorithms.

Apparently, establishing a VB brings additional overheads to wireless networks in terms of time and energy. That is, for a wireless network to utilize a VB, a set of nodes has to be selected as a VB and each node in it has to exchange control messages with others to normalize the VB. Therefore, to benefit from a VB, its lifetime has to be long enough so that the profits from it can compensate the losses to build it. Unfortunately, one remarkable characteristic of wireless networks is that topology can be changed frequently due to the mobility, energy exhaustion, or temporal communication error of wireless nodes, which means that we cannot take advantage of VB without a proper strategy to handle the activeness of wireless networks. This problem can be alleviated by minimizing efforts to fix an incomplete VB or by extending the lifetime of it. However, in many cases, the problem is underestimated and computing a new VB is the only way to deal with it, which becomes more inefficient as the wireless network is getting larger and more dynamic.

In this paper, we introduce a new CDS maintenance algorithm for large scale dynamic wireless networks, namely, Recyclable CDS Algorithm (RCDSA) by exploiting CDS-BD-C1 by us [1] whose approximation ratio is 10.359. When a node is added to or an existing node is deleted from current graph, our centralized algorithm recycles the CDS of current graph to get a new one. For this reason, by implementing our algorithm, the communication of two nodes in the same partition will not be disturbed while current CDS is disconnected by a node failure and reconnected again. RCDSA regenerates a new CDS when a node is added within a linear time. In case that current CDS is partitioned by a node deletion, it quickly repairs the broken CDS by adding at most eight nodes. To evaluate the performance of the proposed algorithm, we define average VB size and computation time as the basic performance metric. In addition, to capture the degree of effort for a wireless network to normalize a VB in it, we introduce a new metric, Resemblance Ratio (RR), which evaluates how much a new VB is similar with its previous version. Our simulation results show that the average size of CDS by RCDSA is almost same with CDS-BD-C1, which coincides with our theoretical analysis, but at least three times faster than that in general. Also, the RR of RCDSA is at least 90%, which indicates that much of current control information (i.e. routing path) in each node in current CDS can be reused even after the CDS is reorganized.

The rest of the paper is organized as follows. In Section 2, we define our problem and introduce performance evaluation metrics. We present RCDSA with its theoretical analysis in Section 3. In Section 4, we describe our simulation results and give their analysis. Finally, we make a conclusion and present several future research directions in Section 5.

2 Problem Definition and Performance Evaluation Metrics

We assume every node in a wireless network is homogeneous and use UDG $G = (V, E)$ to represent the network, where V is a set of nodes in the network and $(u, v) \in E$ is a transmission link between node u and v . We use $V(G)$ and $E(G)$ to represent V and E of G . When a node is added or deleted, G is transformed into G' . $M(G)$ represents a Maximal Independent Set (MIS) of G . $C(G)$ is a CDS of G . $N(G)$ is the set of nodes used to link nodes in $M(G)$. That is, $N(G) = C(G) - M(G)$. At last, $Hopdist(x, y)$ is the hop distance between $x \in V$ and $y \in V$ over the shortest path between them and $Euclidist(x, y)$ is the Euclidean distance between x and y . We mention a node in $N(G)$ **useless** if either it is not used to connect MIS nodes or $C(G)$ is still connected without it. Otherwise, we call it **useful**.

In this paper, our goal is to design an efficient VB maintenance algorithm for large scale dynamic wireless networks, while optimizing the size of VB. This algorithm needs to solve a problem to compute an optimal $C(G')$ by using $C(G)$ with a minimal effort. The problem can be seen as a weak MCDS problem because we have more information than what we do in a pure MCDS problem. This problem has to be NP-hard, otherwise we can solve MCDS by starting from a graph with one node, adding one node and computing a new optimal CDS repeatedly. Naturally, the size of CDS is still an important metric. In addition, computation time is an important performance metric since the faster a VB maintenance algorithm for large scale dynamic wireless networks is, the better it can react against a topology change and is more efficient. At last, as we mentioned, the amount of effort (i.e. number of control messages exchanged, energy consumption, and etc.) for a wireless network to normalize a new CDS is an important metric. Intuitively, such effort is directly related to the degree of the change from an old CDS to its next version. For example, when a node is added to current network, if it is adjacent to one CDS node, we can keep using current CDS without incurring any overhead. On the other hand, if one CDS node is deleted and current CDS is partitioned, more control messages will be generated to handle this situation than the former case. As a result, we define the Resemblance Ratio (RR) of a new CDS over its old version to capture the degree of the difference between them, which is defined as $E_{Old \cap New} / E_{New}$, where $E_{Old \cap New}$ is the number of edges which exist in both old and new CDS and E_{New} is the number of edges in the new CDS.

3 Recyclable CDS Algorithm (RCDSA)

In this section, we introduce our centralized algorithm RCDSA. Generally, RCDSA can be divided into following three sub parts: 1) Initial MIS and CDS construction, 2) Handling a new node insertion, and 3) Handling an existing node deletion.

3.1 Initial MIS and CDS Construction

To get $C(G)$ of any G , we use CDS-BD-C1. This algorithm computes $M(G)$ first. To connect any two nearest nodes in $M(G)$, at most two interconnecting nodes are used. Then, $|N(G)| \leq 2(|M(G)| - 1)$, and $|C(G)| \leq 3|M(G)| - 2$. From [7], we have $|M(G)| \leq 3.453opt + 8.291$ for any G , and thus $|C(G)| \leq 10.359opt + 22.873$. Therefore, the approximation ratio of CDS-BD-C1 is 10.359. Before going further, we introduce some theorems that will be used in the rest of this paper.

Lemma 1. *Suppose we have $C(G)$ and every node in $N(G)$ is useful. Then, $|C(G)| \leq 3|M(G)| - 2$ is always true.*

Proof. The hop distance from an MIS node x to the nearest MIS node y is at most three hops. Therefore, to connect nodes in $M(G)$, we need at most $2(|M(G)| - 1)$ nodes. Since we are assuming all nodes in $N(G)$ are useful, $|C(G)| = |M(G)| + |N(G)| \leq |M(G)| + 2(|M(G)| - 1) = 3|M(G)| - 2$.

Lemma 2. *Suppose an MIS based CDS of G is partitioned into P_1, \dots, P_n by deleting a node in the CDS and $M(G)$ is a valid MIS of G and all nodes in $N(G)$ are useful. Define the distance between two partitions P_i and P_j as $\forall x \in M(P_i) : \forall y \in M(P_j) :: \min(\text{Hopdist}(x, y))$. Then, the distance between two nearest partition is at most three.*

Proof. Now, assume P_i and P_j are two nearest partitions. Then, there have to be $x \in P_i$ and $y \in P_j$ such that $\forall x : \forall y :: \text{Hopdist}(x, y)$ is the minimum. Since the distance between two MIS nodes x and y is at most three hops, the distance between P_i and P_j is at most three.

Lemma 3. *When a node in a CDS is deleted, the CDS can be divided into at most five parts.*

Proof. By [6], in UDG, each node has at most five independent nodes in its neighborhood. Therefore, a CDS can be divided into at most five parts by a node deletion.

3.2 Handling a New Node Insertion

Suppose G becomes G' by a node addition. Then, we may need to compute $C(G')$. Now, we divide every possibility into following two cases and present algorithms to handle them.

- **Case 1 - none of x 's neighbors is in C :** set $M(G') = M(G) \cup \{x\}$. Select one of x 's neighbor y and set $N(G') = N(G) \cup \{y\}$.
- **Case 2 - at least one of x 's neighbors is in C :** if none of x 's neighbors is in $M(G)$, set $M(G') = M(G) \cup \{x\}$.

Theorem 1. *In Case 1, $M(G')$ is an MIS. $C(G') = M(G') \cup N(G')$ is a CDS and $|C(G')| \leq 3|M(G')| - 2$.*

Proof. Since x is not adjacent to any node in $M(G)$, $M(G') = M(G) \cup \{x\}$ is still a valid MIS. Because x , a new MIS node is connected with $|C(G)|$ through y , $C(G') = C(G) \cup \{x, y\}$ is a CDS. Now assume $|C(G)| \leq 3|M(G)| - 2$ before adding x to $C(G)$. Then, $|C(G')| = |M(G')| + |N(G')| = (|M(G)| + 1) + (|N(G)| + 1) \leq 3|M(G)| - 2 + 2 = 3|M(G')| - 3$ and thus the theorem is true.

Theorem 2. *In Case 2, $M(G')$ is an MIS. $C(G') = M(G') \cup N(G')$ is a CDS and $|C(G')| \leq 3|M(G')| - 2$.*

Proof. Assume x does not have any neighbor in $M(G)$, since otherwise the proof is trivial. Then, $M(G') = M(G) \cup \{x\}$ is a MIS of G . Since x is adjacent to $y \in N(G)$, $C(G') = C(G) \cup \{x\}$ is a CDS. Now assume that $|C(G)| \leq 3|M(G)| - 2$ was true before x was added to $M(G)$. Then, $|C(G')| = |M(G')| + |N(G')| = |M(G)| + 1 + |N(G)| \leq 3|M(G)| - 2 + 1 = 3|M(G')| - 4$ and thus the theorem is true.

3.3 Handling an Existing Node Deletion

Suppose G becomes G' by a node deletion. Then, we may need to compute $C(G')$. That is, for any $x \in C(G)$, $C(G') = C(G) - \{x\}$ may not be a good CDS of G' , which means that $M(G') = M(G)$ may not be an MIS for G' or $C(G) - \{x\}$ is partitioned. To resolve those problems, we proceed following three steps. First, we recover $M(G')$. Then, we execute Algorithm 1 to recognize the number of CDS partitions and nodes included in each partition as well as remove useless CDS nodes in each partition. If we have more than one CDS partition, we choose some non-CDS nodes and add them to $N(G')$ so that $C(G') = M(G') \cup N(G')$ becomes a CDS.

MIS reconstruction. this part of RCDSA is initiated when a node $x \in C(G)$ is deleted. If $x \in M(G)$, set $M(G') = M(G) - \{x\}$ and $N(G') = N(G)$. Now denote Y be the set of x 's neighbor nodes. For each $y \in Y$, if y is not adjacent to any node in $M(G')$, then make $M(G') = M(G') \cup \{y\}$. If $x \in N(G)$, set $M(G') = M(G)$ and $N(G') = N(G) - \{x\}$.

Theorem 3. *After the MIS reconstruction phase, $M(G')$ is an MIS of G' .*

Proof. If $x \in N(G)$, then $M(G)$ is a good MIS for G' . Now, suppose $x \in M(G)$. Since by the definition of MIS, every nodes in G has to be in $M(G)$ or adjacent to a node in $M(G)$, and thus the neighbors of x has to be included in one of following two separate sets, $S_1(G)$, and $S_2(G)$. That is, a node s is in $S_1(G)$, if s is adjacent to a node in $M(G) - \{x\}$. Otherwise, s is in $S_2(G)$. Clearly, $M(G) - \{x\}$ is an MIS of $V(G) - x - S_2(G)$. Assume after MIS reconstruction algorithm is executed, we obtain an MIS $M(S_2)$ of $S_2(G)$. Then, obviously, $M(G') = (M(G) - \{x\}) \cup M(S_2)$. Since $M(G) - \{x\}$ and $M(S_2)$ are MISs of $V(G) - x - S_2(G)$ and $S_2(G)$, respectively, and there is no edges between $M(G) - \{x\}$ and $M(S_2)$, $M(G')$ is an MIS of G' .

Algorithm 1. Sweeper - $(G(V, E), COLOR, v_i)$

```

1: color  $v_i$  in  $COLOR$ 
2: if  $v_i$  is an MIS node then
3:   for each CDS neighbor of  $v_i, v_j$  do
4:     if  $v_j$  is not colored yet then
5:       Sweeper ( $G, COLOR, v_j$ )
6:     end if
7:   end for
8:   return TRUE
9: else
10:  for each CDS neighbor of  $v_i, v_j$  do
11:    if  $v_j$  is not colored yet then
12:       $T_j =$  Sweeper ( $G, COLOR, v_j$ )
13:    end if
14:  end for
15:  if any of  $T_j$  is TRUE then
16:    return TRUE
17:  else
18:    switch  $v_i$  to non-MIS node
19:    return FALSE
20:  end if
21: end if

```

Sweeper algorithm. after $M(G')$ is recovered by processing all neighbor of $x \in G'$, we execute Sweeper, a variation of depth first search algorithm. The goals of this algorithm are 1) recognize each CDS partition and 2) remove useless CDS node in each partition. Sweeper assigns the same color number to a set of CDS nodes only if they are in the same CDS partition. It also sweeps useless CDS nodes in each CDS partition. In detail, for each v_i , we execute **Sweeper** (G, W, v_i) if v_i is a CDS node, but not colored yet, where W is the smallest number of unused color.

Theorem 4. *After Sweeper is executed, every CDS in each partition is useful and any two CDS nodes have same color if and only if they are in the same partition.*

Proof. Since Sweeper is a tree traversal algorithm and it visits only CDS nodes, it cannot visit two disconnected CDS nodes within one search trial. Therefore, two nodes in separate partitions will be colored differently. Now assume after Sweep is executed, a subtree of P_i is two connected, where P_i is a partition in a broken CDS. This can happen only if Sweep visits one MIS nodes two times. However, Sweep prevents this from happening by line 4 and line 11. At last, by line 18~19 Sweep deletes any useless CDS node. Therefore, the theorem holds true.

Reconnection algorithm. now, we have at most five partitions and by Lemma 3 and the distance between any two nearest partition is three hops by Lemma 2. Therefore, we need to add at most eight nodes to current $N(G')$ to

make $C(G') = M(G') \cup N(G')$ as a good CDS of G' . To select these new interconnecting nodes, we first reduce each CDS partition into one node and put an edge between them only if they are connected in the original graph through at most two non-MIS nodes. Denote this new graph as \hat{G} . Then, we can create an adjacency matrix of \hat{G} . Now, our problem to select the interconnecting nodes in G' is reduced to computing a spanning tree in \hat{G} , and then a simple tree traversal algorithm like depth first search is good enough to solve this problem. Now, from this solution, we can decide which non-MIS nodes has to be added to $N(G')$ to make $M(G') \cup N(G')$ connected.

Theorem 5. *Assume RCDSA reconnects a broken CDS $C(G)$ and $C(G')$ is a resulting CDS. Then, $|C(G')| \leq 3|M(G')| - 2$ is still true.*

Proof. By MIS construction phase, we have an MIS $M(G')$ of G' . Using Sweeper algorithm, we removed every useless node in each partition. Then, by Lemma 1, $|P_i| \leq 3|M(P_i)| - 2$, where $M(P_i)$ is the set of MIS nodes in P_i . Now assume we have n partitions. By Reconnection algorithm, we add at most $2(n-1)$ nodes to CDS. Therefore, $|C(G')| = \sum_{1 \leq i \leq n} |P_i| + 2(n-1) \leq 3 \sum_{1 \leq i \leq n} |M(P_i)| - 2n + 2(n-1) = 3 \sum_{1 \leq i \leq n} |M(P_i)| - 2 = 3|M(G')| - 2$, and the theorem holds true.

Theorem 6. *The time complexity of RCDSA is $O(n^2)$.*

Proof. To handle a node addition, it takes $O(n)$ time. For a node deletion, MIS recovery requires $O(n^2)$ time. Since Sweeper is a variation of depth first search, it requires $O(E+V)$ time. To recognize the edges between any two partitions, it takes $O(n^2)$ time and to reconnect them, it requires a constant time. Therefore, the time complexity of RCDSA is $O(n^2)$.

4 Simulation Results and Analysis

To evaluate the performance of our algorithm, we compare our centralized algorithm RCDSA with CDS-BD-C1. Since we incorporated a mechanism to handle the activeness of wireless network to CDS-BD-C1, these simulation results will show the effectiveness of this approach. For the simulation, we used a desktop computer with 2.4Ghz Intel Core2 CPU and 2GB memory. We installed Fedora Core 7.0 by Rad Hat as an operating system. For precise and fair running time comparison, we executed each algorithm one by one while only basic demon processes are running on the computer. The implementation of each algorithm exactly follows its description in the literature in terms of procedures and time complexity. We use a 100 by 100 virtual space and randomly deploy wireless nodes. The number of nodes in a network varies from 10 to 100 by increasing 10 per each parameter setting. We set the maximum transmission range of node to 15 and 25. Per each parameter setting, we randomly generated 100 connected graph instances and calculated average CDS size, computation time, and RR. To make our discussion more brief, in Figure 1, we put a tag on each parameter setting. Its interpretation is as follows: DA represents we use RCDSA to handle a node addition. DD and SD mean we use RCDSA and CDS-BD-C1 to handle a node deletion, respectively.

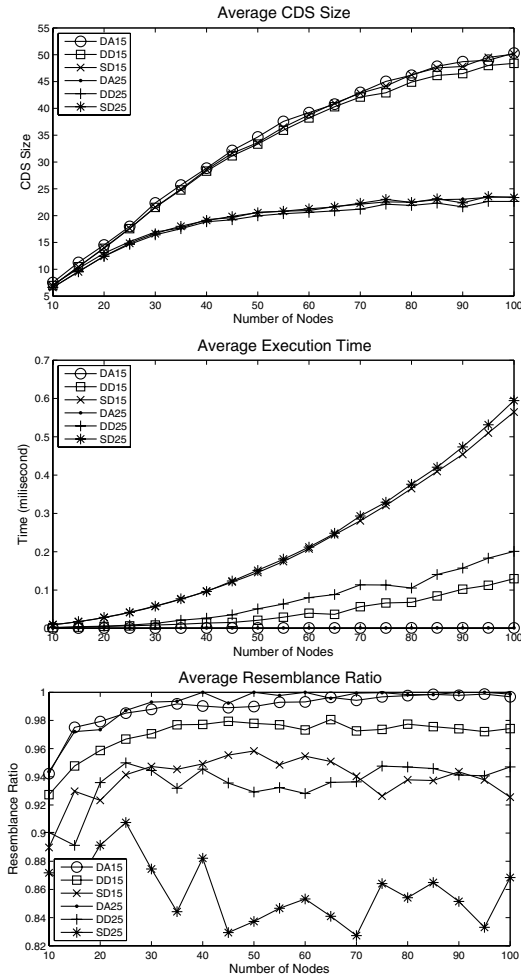


Fig. 1. Each figure shows the comparison result in terms of average CDS size, execution time, and resemblance ratio

The last number 15 and 25 represent the maximum transmission range of nodes in a wireless network in which the simulation is done.

The first graph in Figure 1 shows average size of CDS generated by each algorithm. All curves look almost similar, which coincides with our theoretical analysis that RCDSA has the same approximation ratio with CDS-BD-C1. When the maximum transmission range is longer, we usually have bigger complete subgraph in a given graph G , and thus we have smaller MIS. Therefore, it is natural that the curves in the graph are divide into two groups. The reason that the size of CDS by RCDSA is smaller than CDS-BD-C1 is that when a node is deleted and RCDSA is executed, Sweeper deletes every useless CDS nodes in each CDS partition and this optimizes the final CDS.

The second graph in Figure 1 shows the average running time for each parameter setting. The time complexity of RCDSA to handle a node addition is $O(n)$ and even its structure is so simple. Thus, the shape of the curves for DA15 and DA25 resembles a line. Since CDS-BD-C1 does not have any explicit way to handle a node addition, we do not compare RCDSA with CDS-BD-C1 in this situation. Now let us consider node deletion. If a non-CDS is deleted or CDS is still connected after one CDS node is removed, then we do not need to compute a new CDS, but can use current one. Therefore, in this simulation, we deleted a node such that the remaining CDS nodes are disconnected. The reason that DD is faster than SD is that even though the running time of both algorithms are $O(n^2)$ in theory, DD spends less time due to its structural simplicity. Interestingly, RCDSA shows different performance under DD15 and DD25. In DD, RCDSA identifies every link between CDS partitions, and such links consist of non-MIS nodes. Therefore, when the maximum transmission range is 15, the size of MIS increases and the number of non-MIS nodes decreases and as a result, DD15 is faster than DD25.

Now, let us take a look at the RR of each algorithm. In case of DA, it perfectly maintains current CDS structure and add one node to CDS if necessary. Therefore, naturally, the RR of DA is extremely high as we can see in Figure 1. Based on its property, if CDS-BD-C1 starts from a randomly selected root, its reusability can be very low. To make our comparison fair, we fix the root for CDS-BD-C1 and perform the simulation. In general, RCDSA has higher RR than CDS-BD-C1. This is because in contrast to RCDSA, which uses the most of current MIS to compute a new MIS, CDS-BD-C1 computes a new MIS all over again, and when one node is deleted, the choice of a new MIS over the remaining node can be quite different. At last, when the maximum transmission range is longer, the deletion of one MIS node affects less the rest of CDS, and thus RCDSA has higher reusability ratio in DD25 than DD15. So far, we learn that the execution time of RCDSA increases relatively slower and the RR of RCDSA remains higher than those of CDS-BD-C1's. We also saw that when the maximum transmission range is shorter, the performance of RCDSA is better. From those results, we can conclude that RCDSA is a good VB maintenance algorithm for large size sparse dynamic wireless networks.

5 Conclusion and Future Work

In this paper, we introduced Recyclable CDS Algorithm (RCDSA), 10.359-approximation algorithm to compute and maintain CDS for large scale dynamic wireless networks. By recycling current CDS to build a new one, RCDSA becomes fast and efficient. However, still, RCDSA can be improved. Since RCDSA is a centralized algorithm with limited applications, designing RCDSA as a distributed algorithm can be a good future work. We also consider designing a k -connected RCDSA, where a node can communication with others without being interrupted while current CDS is broken and repaired.

References

1. Kim, D., Wu, Y., Li, Y., Zou, F., Du, D.-Z.: Constructing minimum connected dominating sets with bounded diameters in wireless networks. *IEEE Transactions on Parallel and Distributed Systems* (2008)
2. Ephremides, A., Wieselthier, J., Baker, D.: A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceeding of IEEE* 75(1), 56–73 (1987)
3. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. *Algorithmica* 20, 374–387 (1998)
4. Cardei, M., Cheng, X., Cheng, X., Du, D.-Z.: Connected domination in multihop ad hoc wireless networks. In: *Proc. of International Conference on Computer Science and Informatics* (March 2002)
5. Ruan, L., Du, H., Jia, X., Wu, W., Li, Y., Ko, K.-I.: A greedy approximation for minimum connected dominating sets. *Journal of Theoretical Computer Science* 329, 325–330 (2004)
6. Wu, W., Du, H., Jia, X., Li, Y., Huang, S.C.-H.: Minimum connected dominating sets and maximal independent sets in unit disk graphs. *Theoretical Computer Science* 352 (2006)
7. Funke, S., Kesselman, A., Meyer, U., Segal, M.: A simple improved distributed algorithm for minimum CDS in unit disk graphs. *ACM Transactions on Sensor Network* 2(3), 444–453 (2006)