IEMS Vol. 10, No. 3, pp. 221-231, September 2011.

An Optimization Algorithm for Minimum Connected Dominating Set Problem in Wireless Sensor Network

Namsu Ahn
Defense Agency for Technology and Quality
275-18, Boksu-dong, Seo-gu, Daejeon, 302-840
Tel: +82-10-6590-0423, E-mail: namsu.ahn@gmail.com

Sungsoo Park†
Department of Industrial and Systems Engineering
KAIST, 373-1, Gusong-dong, Yusong-gu, Daejeon, 305-701,
Tel: +82-42-350-3121, Fax: +82-42-869-3110, E-mail: sspark@kaist.ac.kr

Received, March 12, 2011; Revised, May 18, 2011; Accepted, May 24, 2011

Abstract. One of the critical issues in wireless sensor network is the design of a proper routing protocol. One possible approach is utilizing a virtual infrastructure, which is a subset of sensors to connect all the sensors in the network. Among the many virtual infrastructures, the connected dominating set is widely used. Since a small connected dominating set can help to decrease the protocol overhead and energy consumption, it is preferable to find a small sized connected dominating set. Although many algorithms have been suggested to construct a minimum connected dominating set, there have been few exact approaches. In this paper, we suggest an improved optimal algorithm for the minimum connected dominating set problem, and extensive computational results showed that our algorithm outperformed the previous exact algorithms. Also, we suggest a new heuristic algorithm to find the connected dominating set and computational results show that our algorithm is capable of finding good quality solutions quite fast.

Keywords: Connected Dominating Set, Cutting Plane, Integer Programming, Wireless Sensor Networks

1. INTRODUCTION

Due to the recent rapid advances in digital technologies and the mass production, developments of multi-functional, low-cost and low-power sensors become possible. Generally, sensor is composed of four major units: power unit, sensing unit, processing unit and transceiver unit, and each unit performs a unique function. The power unit provides power to all the other units, and the sensing unit takes charge of the monitoring task and data converting task. Thus, if a certain phenomenon happens, the sensing unit detects the phenomenon changes, such as temperature, light, sound and humidity, and then converts the sensed analog data to digital data. The processing unit can manipulate the collected data and store the collected data in the storage device for transmitting. The transceiver unit can transmit/receive information over a wireless network.

Typically, a large number of sensors collaborate using wireless communication, and the sensors gather, process and transmit information over a wireless network to a remote running application that makes decisions based on this information. Since the deployed sensors are clustered and communicate in wireless channels, we call it wireless sensor network (WSN). Nowadays, WSN has many applications, such as habitat monitoring, forest-fire detection, patient status monitoring, home appliance, inventory tracking and battlefield monitoring.

WSN is featured by no fixed infrastructure, multi-hop communication and limited resources (battery capacity and bandwidth). These characteristics pose new difficulties in designing a routing protocol. Some existing routing protocols for ad hoc networks (Ng and Lu, 1999; Pei *et al*., 2000) are based on flooding mechanism (*i.e.*, upon receiving a packet, transmit the packet to all of its neighbors). Therefore if we use these protocols, it

† : Corresponding Author

not only devastates the resources of the sensors, but gives negative effect on the throughput of the whole network. Furthermore, it probably causes *broadcasting storm problem* as indicated in Ni *et al.* (2002), which results in excessive data redundancy, contention and collision.

In Sinha *et al.* (2001), a new routing protocol based on an overlaying virtual infrastructure, so called virtual backbone, is proposed and the authors showed that, using the backbone can reduce the routing overhead dramatically. Simply, a virtual backbone is defined as a subset of sensors, which can connect all the sensors in the network. This concept is frequently used to simplify the network and improves the efficiency of the routing.

In many existing virtual backbone schemes which can be found in Alzoubi *et al*. (2002), Chen *et al*. (2002), Dai and Wu (2004), Stojmenovic *et al*. (2002) and Wu and Dai 2004, a connected dominating set (CDS) is suggested to use as a backbone. To simplify our discussion, we use a connected graph $G = (V, E)$ to represent the network, where V and E represent the set of vertices and the set of edges, respectively. Each vertex $v \in V$ indicates a sensor, and there is an edge $e(= uv) \in E$ which denotes that sensor u is within sensor v's communication range and vice versa. A set of sensors is called a dominating set if each of the sensors in the network is either in this set or has a neighbor (sensor u is neighbor of sensor v if there exists an edge between the two vertices) in the set. Typically, the sensors in the dominating set are called dominators, and the other sensors are called dominatees. A dominating set is called a CDS if the subgraph induced by the dominating set is connected, and the connectivity among the dominators is required for proper routing of signals. Therefore if any dominatee want to send a message to dominator(or dominatee), it first send a message to connected neighbor dominator. If CDS is used as a backbone, we can obtain the following good characteristics of the network.

- Routing overhead can be reduced as shown in Wu and Li (1999) because only the sensors in CDS need to maintain the routing information. Thus, if a dominatee wants to send a packet to another dominatee, it sends the packet to its dominator. Then the dominator will deliver the packet to the destination dominatee.
- Energy efficient area coverage is possible as indicated in Carle and Simplot (2004) and Chen *et al*. (2002). Since CDS is a good approximation of an area, dominators in CDS can take over the dominatees' sensing task. Thus, if the dominators are actively performing the sensing task, all of its dominatees probably enter into a low-battery sleep state to save energy for future use.

Usefulness of the CDS in WSN has been demonstrated in many communication protocols such as media access coordination, unicast, multicast/broadcast, location-based routing, energy conservation, resource discovery and topology control (More comprehensive review can be found in Blum *et al.* (2004)).

Since the number of sensors forming the virtual backbone needs to be as small as possible to decrease the protocol overhead and energy consumption, it is desirable to form a minimum sized CDS. Finding a minimum sized CDS problem is referred to as the minimum connected dominating set (MCDS) problem. For example, vertex set $\{a, b, c\}$ in Figure 1 forms a CDS, thus the dominators are $\{a, b, c\}$ and the dominatees are $\{d, e, f\}$.

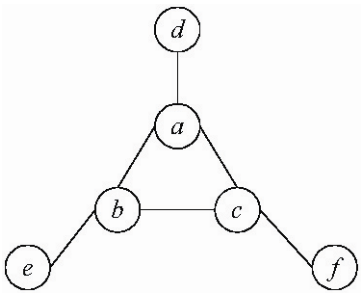

Figure 1. Example of the CDS.

Finding an MCDS is an NP-Hard problem Garey and Johnson (1979), and many researches have been performed on the MCDS problem and a comprehensive review on the algorithms can be found in Blum *et al.* (2004). In this paper, we review the centralized CDS construction algorithms only (Although CDS can be constructed in a distributed manner (Chen *et al.*, 2009; Wang *et al.*, 2009), it is beyond the scope of this paper).

We review the performance guaranteed approximation algorithms first. In Guha and Khuller (1998), two algorithms are suggested. One forms a CDS of size at most $2(1+H(\Delta)) \cdot |MCDS|$ and the other one constructs a CDS of size $(3+ln(\Delta)) \cdot |MCDS|$, where H, $|MCDS|$ and Δ represent the harmonic function, size of the MCDS and the maximum degree of the given graph, respectively. The authors in Li *et al*. (2005), Min *et al*. (2006) and Ruan *et al*. (2004) reported $(4.8+ln5) \cdot |MCDS|$, $6.8 \cdot |MCDS|$ and $(3+ln(\Delta)) \cdot |MCDS|$ approximation algorithms, respectively.

There exist many heuristic algorithms for the MCDS problem. Relatively latest ones can be found in Chen *et al.* (2010) and Morgan and Grout (2008). However, these algorithms cannot guarantee the quality of the obtained solutions because no information on the lower bound can be obtained.

Although solving the MCDS problem exactly is an important research goal, there have been few exact approaches. One simple scheme is enumerating all subset of the vertices, and the algorithm shown in Fomin *et al.* (2008) breaks the $2^{|V|}$ barrier by suggesting $O(1.9407)^{|V|}$ algorithm. However, no implementation and performance test of the algorithm can be found.

Other two optimal approaches used mathematical formulations to obtain MCDS (Morgan and Grout, 2008;

Yuan, 2005). However, the performances of the algorithms reported by the authors were not impressive. In this paper, we propose an improved optimal algorithm using the mathematical formulation in Yuan (2005) and compare the performances with the previous two approaches. The computational results showed that, our algorithm outperforms the previous optimal approaches in terms of the running time to obtain an optimal solution. Also, we propose a new heuristic which use the formulation and the idea of our improved optimal algorithm.

This paper is organized in the following way. In Section 2, we review notation and definition. In Section 3, the existing mathematical formulations and algorithms for the MCDS problem will be discussed, and an improved optimal approach will be introduced in the same section. Section 4.1 shows the performance of the improved optimal approach suggested in this research. Section 4.2 includes description of the heuristic which is suggested in this research for computing the connected dominating set efficiently, and shows the performance of the heuristic. Finally, in Section 5, we conclude this paper.

2. PRELIMINARIES

This section provides some background information to understand the rest part of the paper. As noted before, WSN can be represented by a simple graph $G = (V, E)$ and the following definition and notation from graph theory will be used throughout the paper.

- *Open neighbor set*, $N(u) = \{v \mid (uv) \in E\}$, is the set of vertices adjacent to vertex u.
- *Closed neighbor set*, $N[u] = N(u) \cup \{u\}$, is the set of vertices adjacent to vertex u and u itself.
- *Independent set* is a subset of V such that no two vertices are adjacent in G. For example, $\{d\}$, $\{d, e\}$ and $\{d, e, f\}$ are independent sets in Figure 1.
- *Maximal independent set* is an independent set such that adding any vertex not in the set breaks the property of the independent set. For example, in Figure 1, the independent set $\{d, e, f\}$ is a maximal independent set since addition of any other vertices (a or b or c) makes some vertices in the set are connected.
- *Dominating set* is a subset of V such that each vertex is either in the set or has a neighbor in the set. Note that every maximal independent set is a dominating set, but the converse is not true.
- *Connected dominatig set* is a dominating set whose induced subgraph is connected. For example, $\{a, b, c\}$ is a CDS in Figure 1.
- *Steiner tree* is a tree which, for a given subset T of V, connects the vertices of T possibly using the vertices in $V \backslash T$. For example, in Figure 1, if T is given as $\{d, e, f\}$, a Steiner tree can be constructed by adding the remaining vertices $\{a, b, c\}$ and the edges ab, ac, ad, be and cf.
- Vertex cut is a subset of V whose removal disconnects the graph.

If we use maximal independent set and Steiner tree, the following simple procedure can construct a CDS. We first form a maximal independent set to identify a dominating set, and let the generated dominating set from the first step be the vertex set T in Steiner tree. Then CDS can be constructed by finding a Steiner tree. Many heuristic algorithms and approximation algorithms have been developed using this idea, and the latest one can be found in Min *et al.* (2006).

3. MATHEMATICAL FORMULATIONS AND OPTIMAL ALGORITHMS

To the best of our knowledge, two different mathematical formulations exist for the MCDS problem, first one is shown in Morgan and Grout (2008) and the second one is suggested in Yuan (2005). Two formulations used different modeling techniques. The first one used a $2|V|+2|E|$ number of variables with $3|V|+2|E|$ number of constraints, approximately. The second one used a $|V|$ number of variables, but it required an exponential number of constraints to represent the connectivity of the induced subgraph. At first sight, the first formulation seems to be better. However, after performing extensive computational experiments, we concluded that, if an improved optimal algorithm developed in this paper is used, the second formulation may be used to find an optimal solution much faster than the first formulation.

Generally, representing the connectivity requirement for the graph problems asks for an exponential number of inequalities. To avoid this, the first formulation used the flow variables in the formulation. The main idea of the formulation is, when the vertices are selected to form a CDS, any vertex included in the CDS can send a flow to each of the other vertices in the CDS (using only the selected vertices). Each vertex included in the CDS requires at least one unit of flow, and the root vertex (vertex 1) contains enough flows to supply the required flows in each vertex in the CDS. Thus, if the root vertex is not included in the CDS, it transfers all of its flows to one of its neighboring vertices which is selected to be included in the CDS. We first construct a directed graph $G = (V, A)$ by replacing each edge $e(= ij) \in E$ by two arcs ij and ji, and define the decision variables as the following.

$$x_i = \begin{cases} 1, & \text{if vertex } i \text{ is included in the MCDS,} \\ 0, & \text{otherwise} \end{cases}$$

$$f_{ij} = \text{amount of flow from vertex } i \text{ to } j.$$

$$x_i = \begin{cases} 1, & \text{if flow is permitted from root vertex to vertex} \\ 0, & \text{otherwise} \end{cases}$$

Then the formulation can be given as the following.

$$Minimize \sum_{i \in V} x_i \quad (1)$$

Subjectto

$$x_i + \sum_{\{j \mid ij \in A\}} x_j \geq 1,\ i = 1, \cdots, |V| \quad (2)$$

$$\sum_{\{j \mid 1j \in A\}} f_{1j} - \sum_{\{j \mid j1 \in A\}} f_{j1} = |V| - 1 \quad (3)$$

$$\sum_{\{j \mid ji \in A\}} f_{ji} - \sum_{\{j \mid ij \in A\}} f_{ij} \geq xi,\ i = 2, \cdots, |V| \quad (4)$$

$$f_{1j} \geq |V| \times (x_1 + x_j),\ \forall \{j \mid 1j \in A\} \quad (5)$$

$$f_{ij} \leq |V| \times x_i,\ \forall \{ij \in A \mid i \geq 1 \text{ and } j \geq 2\} \quad (6)$$

$$f_{1j} \leq |V| \times y_i,\ \forall \{j \mid 1j \in A\} \quad (7)$$

$$\sum_{\{j \mid 1j \in A\}} y_j \leq 1 + x_1 \times |V| \quad (8)$$

$$x_i \in \{0, 1\},\ i = 1, \cdots, |V| \quad (9)$$

$$y_j \in \{0, 1\},\ \forall \{j \mid 1j \in A\} \quad (10)$$

Constraints (1) minimize the number of vertices which are selected to be included in the CDS. Constraints (2) guarantee that, if the vertex is not included in the CDS, it has at least one neighboring vertices which are selected to be included in the CDS to satisfy the dominance requirement. Constraints (3) assure that the root vertex in the CDS may produce sufficient flows to provide at least one unit flow to the other vertices which are chosen to be included in the CDS. Constraints (4) state that, if any vertex is chosen to be included in the CDS, it consumes at least one unit flow. Constraints (5) and (6) set upper bounds for the flows. Constraints (7) guarantee that flows from the root vertex to the other vertex j can be sent only if y_j is one. While constraint (8) forces that, if the root vertex is not included in the CDS, at most one of its neighbor vertices can be selected in the CDS to send the flows. Constraints (9) and (10) state the binary integer requirements on the variables.

The authors in Morgan Grout (2008) reported that, when ILOG CPLEX (2008) was used as an optimization software to solve the formulation, they could obtain the optimal solutions for the graphs of up to 100 vertices in 1000 seconds.

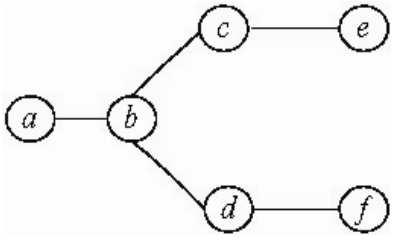

Figure 2. Example of the minimal vertex cut.

Now we review the formulation and the optimal algorithm shown in Yuan (2005). Before giving a detailed explanation, we introduce notation which will be used in the formulation. Let C be a minimal vertex cut (=whose removal disconnects the graph and removing any vertex in the cut fails to form the vertex cut), and $\mathbb{C}$ be a collection of C. For example, in Figure 2, vertex cut $\{a, c, d\}$ is not a minimal vertex cut, but c or d is a minimal vertex cut.

The author in Yuan (2005) proved that, if at least one vertex is selected from C, $\forall C \in \mathbb{C}$, the set of selected vertices form a CDS. Using this characteristic, the authors suggested the following formulation for the MCDS problem (The decision variable x_i indicates that whether the vertex i is included in the CDS or not).

$$Minimize \sum_{i \in V} x_i \quad (11)$$

Subjectto

$$\sum_{i \in C} x_j \geq 1,\ \forall\ C \in \mathbb{C} \quad (12)$$

$$x_i \in \{0, 1\},\ \forall\ i \in V. \quad (13)$$

Since the number of constraints (12) can be huge, representing the formulation in full is impractical. Therefore the author in Yuan (2005) suggested the following constraint generation scheme to identify the violated constraint of (12) only when needed.

Optimal approach in Yuan (2005):

Step 1: Construct the initial formulation using constraints (11) and (13) only.

Step 2: For each vertex $v \in V$, form a subset S of V by setting $S = [v]$. For each S, construct a minimal vertex cut by finding the vertices which has one or several neighbor vertices in $V \backslash S$. Then add the corresponding inequalities of (12) to the formulation.

Step 3: Solve the formulation optimally and let the set of vertices which are selected as D. If D forms a CDS, stop the procedure and output the obtained MCDS. Otherwise, goto *Step* 4.

Step 4: For each vertex $v \in V$, construct a subset S of V using the vertices that v can reach via vertices in D, including v itself. For each S, if we can construct a minimal vertex cut by finding the vertices which has one or several neighbor vertices in $V \backslash S$, store the corresponding inequality of (12).

Step 5: Perform a pair-wise check for the stored inequalities in *Step* 4 to identify the inequalities which are not dominated by the other inequalities (*i.e.*, for given two vertex cuts C_1 and C_2, if C_1 is a subset of C_2, C_2 is dominated by C_1). Only those inequalities are added to the formulation, and goto *Step* 3.

The authors in Yuan (2005) reported that, when the time limit was set to 1 hour and ILOG CPLEX 7.0 was used as the optimization software, they could obtain the optimal solutions for the graph of up to 80 vertices.

Up to this part of the section, we reviewed two different mathematical formulations and the optimal approaches to solve the formulations, but the performances

of the algorithms reported by the authors seem to be not impressive and the approaches have rooms for more improvements. However, since the primary purposes of their researches were developing good heuristic algorithms, no more efforts in optimal approaches could be found. Comparisons of the algorithm in Morgan Grout (2008), original algorithm in Yuan (2005), and our improved algorithm will be given in Section 4.1.

Note that if it is guaranteed that the number of vertices selected in an MCDS is greater than or equal to two, the following additional inequalities can be added to the formulation.

$$\sum_{j \in N(i)} x_j \geq 1, \ \forall \ C \in V \tag{14}$$

Constraints (14) indicate that any selected vertex has at least one or more neighboring vertices to establish the connectivity. Actually, constraints (14) are not required in a correct formulation of the MCDS problem, but adding the constraints to the formulation strengthens the formulation and it reduces the computation time to solve the problem.

Now we discuss our optimal algorithm for the MCDS problem based on the formulation from Yuan (2005). To solve the MCDS problem for a given graph, we used a different constraint generation scheme and this difference results in improvements in the computation times to obtain the optimal solutions.

As noted before, since the number of constraints (12) can be exponential, the constraints (12) need to be dealt implicitly rather than explicitly. We first solve the formulation optimally without constraints (12) and construct the graph using the vertices which are selected. When the current solution is not a CDS, the optimal approach in Yuan (2005) identifies one vertex cut which is violated by the current solution. On the other hand, when there exist several vertex cuts which are violated by the current solution, this approach does not guarantee to identify the smallest sized vertex cut. Since using a minimum vertex cut among the violated vertex cuts may tighten the formulation the most, it probably reduces the computation time to solve the problem.

In this paper, when the constructed subgraph from the current solution consists of several disconnected components, we suggest a procedure to find a minimum vertex cut which separates one component from the other components. We first explain how to find a vertex cut using an example, and then, a procedure which identifies the minimum vertex cut will be discussed.

Assume that, when we solved the formulation without constraints (12) using the example given in Figure 3, two vertices a and c are selected (for this example, suppose that we replaced constraints (14) by $\sum_{v \in N[u]} x_v \geq 1, \ \forall u \in V$, otherwise a and c cannot be chosen). However, since the resulting subgraph is not a CDS, we want to identify a vertex cut which is violated by the current solution a and c.

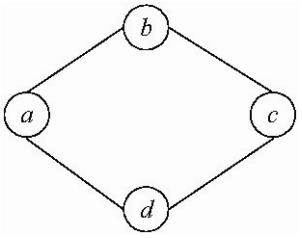

Figure 3. Example of the MCDS problem.

Clearly, since we want to find a vertex cut which separates the two vertices a and c, vertices b and d form a vertex cut which is violated by the current solution. Now we add the corresponding inequality $x_b + x_d \geq 1$ to the formulation. When we solve the enlarged formulation, we can obtain the CDS as $\{a, b\}$ and the procedure stops.

However, identifying the vertex cut probably be a complicating step when the size of the graph is large. Furthermore, it may be harder to find the *minimum* vertex cut among the vertex cuts. Therefore, in this paper, we propose an idea which identifies the vertex cut using the modified graph of G, then a procedure which can be used to find the minimum vertex cut will be illustrated (We used the procedure suggested in Thulasiramanand Swamy (1992) with some modifications).

To identify a vertex cut for a given graph $G = (V, E)$, we first replace each edge by two antiparallel arcs, and then, split a vertex v into two vertices v' and v'', and create an arc directed from v' to v''. Then, replace an arc that is directed from vertex u to other vertex v by an arc directed from u'' to v'. Lastly, assign unit capacities to the generated arcs which are directed from v' to v'', and allocate very large positive (= M) capacities to the generated arcs directed from u'' to v'. The result of applying the procedure to the example in Figure 3 is shown in Figure 4.

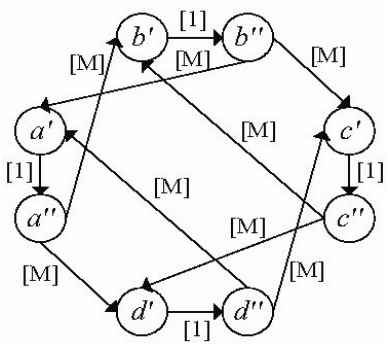

Figure 4. Transformed graph.

Therefore, when we send flows from a source vertex to a sink vertex as much as possible, only the arcs which are directed from v' to v'' limit the amount of flows.

Note that in this example, we have two source ver-

tices a' and a'' (from vertex a) and two sink vertices c' and c'' (from vertex c) instead of a single source vertex and a single sink vertex. Generally, when we want to find a vertex cut using the modified graph, we need to solve the multiple sources and multiple sinks maximum flow problem instead of the ordinary maximum flow problem. However, this problem can be easily reduced to the ordinary single source single sink maximum flow problem. We first add a vertex (= super-source vertex) and create arcs to the source vertices with capacities M. Similarly, we add a vertex (= super-sink vertex) and create arcs from the sink vertices with capacities M. Then, any flow from the super-source vertex to the super-sink vertex corresponds to a flow in the multiple source vertices and multiple sink vertices problem as shown in Cormen *et al*. (1989). If we identified the maximum flow, we also can find the minimum arc cut such that the value of the flow is equal to one (arcs are directed from v' to v''). Then we can obtain a vertex cut of the original graph by restoring the vertices corresponding to the cut arcs.

Suppose that the solution from the formulation without constraints (12) is not a CDS, then the selected vertices form several connected components. Now, for each component, we want to identify the minimum sized vertex cut which separates the current component and the other components, and the corresponding inequalities (12) are added to the formulation. Then, the following procedure can be used to find the minimum sized violated vertex cut.

Procedure to find the minimum sized violated vertex cut:

Step 1: For a given undirected graph $G = (V, E)$, construct a directed graph $G = (V, A)$ by replacing each edge e(= ij)$\in E$ by two arcs ij and ji.

Step 2: Replace every vertex $v \in V$ by two vertices v' and v'' and create an arc directed from v' to v''.

Step 3: Assign unit capacities to the generated arcs in *Step* 2.

Step 4: Replace an arc that is directed from vertex $u \in V$ to the another vertex $v \in V$ by an arc directed from u'' to v'.

Step 5: Assign capacities M to the generated arcs in *Step* 4.

Step 6: Set k as 1.

for all k^{th} component **do**

Step 6.1: Create a super-source vertex and create arcs joining the super-source vertex to the vertices of k^{th} component with capacities M.

Step 6.2: Create a super-sink vertex and create arcs joining the vertices of the other components to the super-sink vertex with capacities M.

Step 6.3: Find a maximum flow from the super-source vertex to the super-sink vertex.

Step 6.4: Identify a minimum arc cut, and obtain a vertex cut of the original graph by restoring the vertices corresponding to the arc cut.

Step 6.5: Drop the generated vertices and arcs in *Step* 6.1 and *Step* 6.2, and increase k by one.

end for

Step 7: Choose the minimum vertex cut among the vertex cuts identified in *Step* 6.

4. DESCRIPTION OF THE HEURISTIC

We suggested an optimal approach and tested the performance of the algorithm in Section 5.1. However, when the size of the problem becomes large, solving the MCDS using the formulation failed to provide an optimal solution within an hour. Therefore, we propose a heuristic which can be used to find the CDS. Major characteristics of the heuristic are linear programming relaxation of the mathematical formulation and minimum vertex cut identification procedure explained in Section 3. The detailed procedure of the heuristic can be given as follows.

Heuristic to compute connected dominating set:

Step 1: Prepare an empty set, and construct the formulation which consists of (11), (13) and (14).

Step 2: Drop the integrality conditions and let the lower bound for the objective value as $\lfloor z \rfloor$

Step 3: Solve the linear programming relaxation.

Step 4: Identify the most fractional variable x_i variable which is close to one, and assign the corresponding vertex to the set.

Step 5: If the set satisfy the dominance requirement and size of the set is greater than or equal to $\lfloor z \rfloor$, go to *Step* 7.

Step 6: Fix the identified x_i variables in *Step* 5 to one in the formulation and return to *Step* 3.

Step 7: Check the connectivity among the vertices which are included in the set. If the set is connected, stop the algorithm and output the vertices in the set. Otherwise, go to *Step* 8.

Step 8: Since the vertices in the set consists of several disconnected components, identify the minimum vertex cut which separates one component from the other components using the procedure described in Section 3.

Step 9: Add the corresponding inequality which is violated by the current solution, and let $\lfloor z \rfloor$:= $\lfloor \text{Set} \rfloor$ + 1 and return to *Step* 3.

In *Step* 4, if the several variables are assigned the same values, we choose the variable whose sum of distances to other vertices included in the set is the minimum.

Note that the number of iterations is finite, and since solving the linear programming relaxation, separating the minimum vertex cut and finding the distances can be done in polynomial times, our heuristic runs in polynomial time.

5. COMPUTATIONAL RESULTS

5.1 Performance of the optimal algorithms

We tested the performances of the optimal approaches for the MCDS problem suggested in Morgan Grout (2008), Yuan (2005) and in this paper. We implemented the three approaches and tested them on randomly generated graphs. For each of the approaches, the algorithm run times are reported. For the approaches given in Yuan (2005) and this research, the number of generated inequalities (12) to obtain the MCDS are also reported. The purpose of the experiment is, to observe the differences in the run times for the three exact approaches.

We used the code obtained from http://www.brandonparker.net/graph_gen.php to generate the connected random graphs. The code can generate three types of connected graphs: sparse, medium and dense graphs for a given number of vertices $|V|$. The graph classification criterion is the number of edges. The number of edges of the sparse graph is less than $|V|\times(|V|-1)/4$, the number of edges of the medium graph is approximately $|V|\times(|V|-1)/4$, and the number of edges of the dense graph is greater than $|V|\times(|V|-1)/4$.

For each type of the graphs, we generated 10 random graphs with the number of vertices ranging from 100 to 1000 with an increment of 100. The three approaches were implemented in C++ and ILOG CPLEX 11.0 was used as optimization software. All experiments were run on an AMD *Athlon*^{TM} 64 X2 Dual Core (2.70 GHz) with 2GB RAM, and running time is given in seconds.

Computational results are given in Table 1, 2 and 3 for each type of the graphs. Note that $|MCDS|$ denotes the size of the minimum connected dominating set and * indicates the problem on which the algorithm failed to obtain the solution within an hour. Also, comparison of the run time is illustrated in Figure 5, 6 and 7.

Table 1. Comparison of the optimal approaches for sparse graph problems.

$\|V\|$	Approach in Morgan Grout (2008)	Approach in Yuan (2005)		Ahn and Park's approach		$\|MCDS\|$
	Run time (sec)	Inequalities	Run time (sec)	Inequalities	Run time (sec)	
100	0.55	15	47.52	4	0.047	31
200	61.08	330	505.55	207	4.71	61
300	12.94	174	1442.94	66	1.42	89
400	56.45	*	*	674	22.15	131
500	1420.75	*	*	501	32.76	155
600	*	*	*	774	52.92	194
700	*	*	*	675	72.31	215
800	*	*	*	1749	446.68	248
900	*	*	*	978	98.26	281
1000	*	*	*	314	122	309

Table 2. Comparison of the optimal approaches for medium graph problems.

$\|V\|$	Approach in Morgan Grout (2008)	Approach in Yuan (2005)		Ahn and Park's approach		$\|MCDS\|$
	Run time (sec)	Inequalities	Run time (sec)	Inequalities	Run time (sec)	
100	11.34	0	4.34	0	0.37	3
200	*	0	31.16	0	6.68	4
300	*	0	92.43	0	23.71	4
400	*	0	233.97	0	108.26	4
500	*	0	1531	0	2228.91	4
600	*	*	*	*	*	*
700	*	*	*	*	*	*
800	*	*	*	*	*	*
900	*	*	*	*	*	*
1000	*	*	*	*	*	*

When we compared the optimal approaches in terms of the algorithm run time, each approach showed different performance depending on the types of the graphs. For sparse graphs, our approach obtained MCDS in the shortest run time, then the approach given in Morgan Grout (2008) follows the next, and lastly, the approach proposed in Yuan (2005) showed the longest run time. On the other hand, for medium and dense graphs, the approach proposed in Morgan Grout (2008) showed the longest run time to obtain the MCDS, and the other two approaches showed equivalent performances. Also, for the given medium and dense graphs, our approach and the approach shown in Yuan (2005) required no additional constraints (12) to obtain the optimal solution. This probably due to the density of the given graphs.

5.2 Performance of heuristic

In Table 4, we use the same problems from Table 1, Table 2 and Table 3 and test the performance of our heuristic. Run time column denotes the running time of the heuristic, $|CDS|$ indicates the size of the connected dominating set obtained from the heuristic and $|MCDS|$ means the size of the minimum connected dominating set.

We can observe that, our heuristic algorithm showed good performance in medium and dense graph problems, but showed relatively poor performance in sparse graph problems.

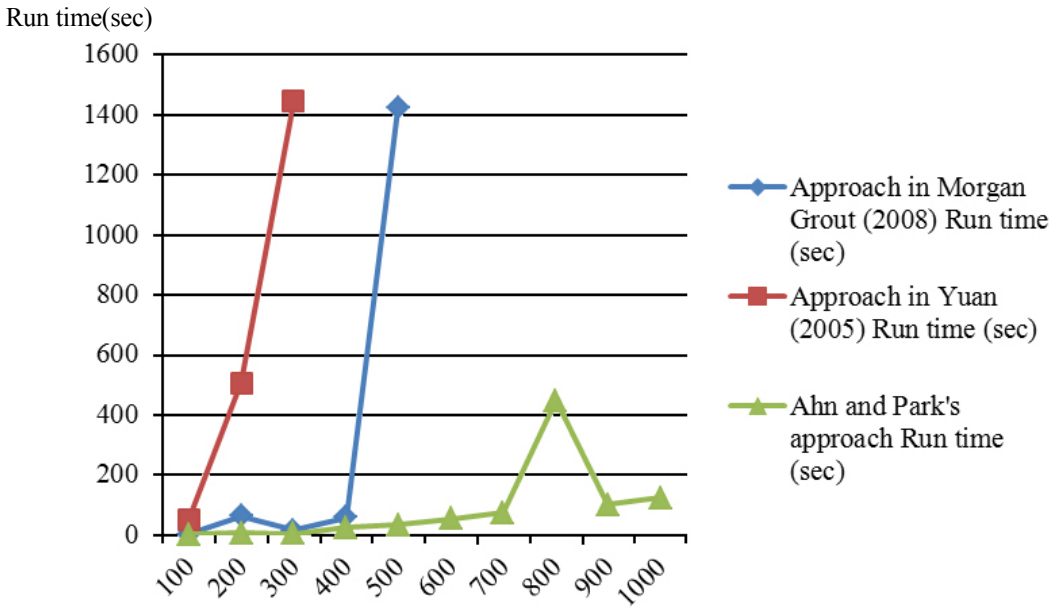

Figure 5. Comparison of run time in medium graph.

Table 3. Comparison of the optimal approaches for dense graph problems.

$\|V\|$	Approach in Morgan Grout (2008)	Approach in Yuan (2005)		Ahn and Park's approach		$\|MCDS\|$
	Run time (sec)	Inequalities	Run time (sec)	Inequalities	Run time (sec)	
100	22.31	0	3.25	0	0.53	2
200	895.913	0	19.27	0	8.64	3
300	*	0	79.2	0	38.31	3
400	*	0	247.97	0	192.35	3
500	*	0	626.33	0	669.75	3
600	*	0	2101.14	0	1989.63	3
700	*	*	*	*	*	*
800	*	*	*	*	*	*
900	*	*	*	*	*	*
1000	*	*	*	*	*	*

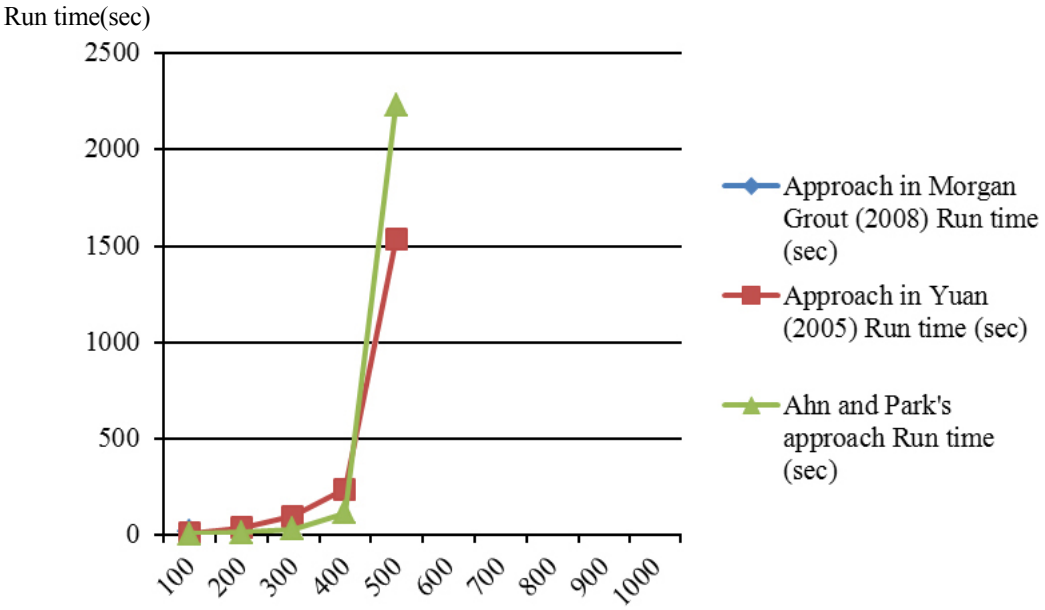

Figure 6. Comparison of run time in medium graph.

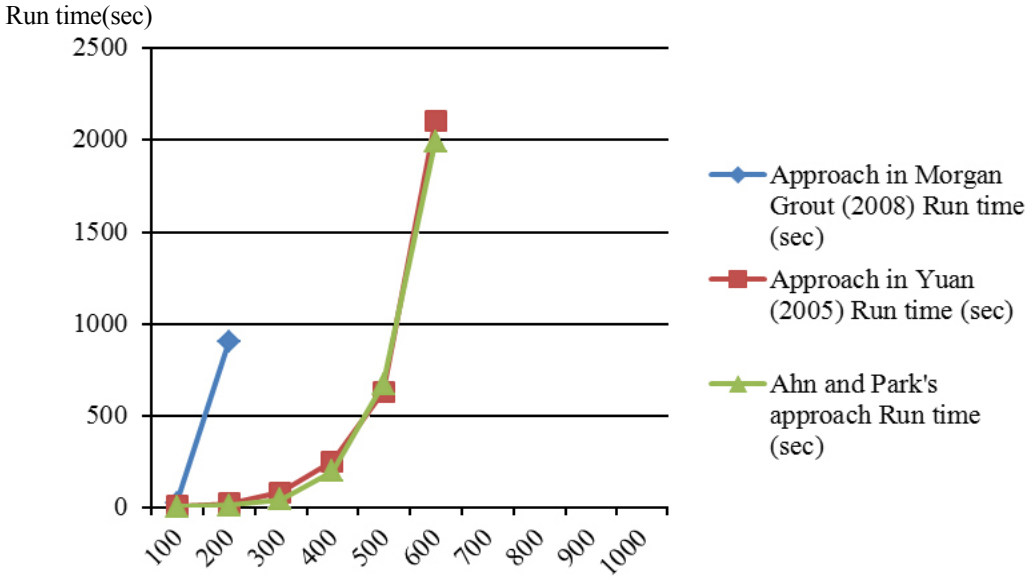

Figure 7. Comparison of run time in dense graph.

6. CONCLUSIONS

Designing a routing protocol is one of the important issues in wireless sensor network. Many virtual infrastructure based protocols have been suggested, and one of them is utilizing the CDS. Since forming a small sized CDS is preferable, many researches have been devoted on this issue.

In this paper, we proposed an improved optimal algorithm for the MCDS problem, and extensive computational experiments showed that our algorithm outperforms the previous approaches in terms of the running time to obtain an optimal solution. Also, we proposed the heuristic in this research which uses the mathematical formulation for the MCDS problem, and the computational experiments show that our algorithm obtains an quite good solution in a reasonable amount of time.Since constructing an MCDS is an important issue in wireless sensor networks, further approach such as branch-and-cut or meta-heuristic methods (genetic algo-

Table 4. Performance of the heuristic in graph problems

$\|V\|$	Sparse Graph			Medium Graph			Dense Graph		
	Run time	$\|CDS\|$	$\|MCDS\|$	Run time	$\|CDS\|$	$\|MCDS\|$	Run time	$\|CDS\|$	$\|MCDS\|$
100	0.016	39	31	0.03	4	3	0.03	3	2
200	0.16	72	61	0.094	4	4	0.16	3	3
300	0.17	98	89	0.35	4	4	0.73	3	3
400	0.75	133	131	1.33	5	4	1.56	3	3
500	1.5	165	155	2.4	5	4	3.6	4	3
600	1.35	195	194	4.6	5	*	6.4	4	3
700	2.07	221	215	5.89	5	*	10.32	4	*
800	3.12	260	248	7.9	5	*	13.21	4	*
900	5.79	314	281	12.8	5	*	20.95	4	*
1000	4.06	341	309	15.77	5	*	21.29	4	*

rithm, simulated annealing, tabu search, ant colony optimization) can be applied to solve the MCDS problem.

REFERENCES

Alzoubi, K. M., Wan, P. J., and Frieder, O. (2002), Distributed Heuristics for Connected Dominating Sets in Wireless Ad hoc Networks, *Journal of Communication Networks*, **4**(1), 22-29.

Blum, J., Ding, M., Thaeler, A., and Cheng, X. (2004), *Handbook of combinatorial optimization*, Kluwer Academic Publishers, Netherlands.

Carle, J. and Simplot-Ryl, D. (2004), Energy-efficient Area Monitoring for Sensor Networks, *IEEE Computer Society*, **37**(2), 40-46.

Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. (2002) Span: An Energy Efficient Coordination Algorithm for Topology Maintenance in Ad hoc Wireless Networks, *ACM Wireless Networks*, **8**(5), 481-494.

Chen, S., Ljubic, I., and Raghavan, S. (2010), The Regenerator Location Problem, *Networks*, **55**, 205-220.

Chen, Q., Fan, W. T., and Zhang, M. (2009), Distributed heuristic approximation algorithm for minimum connected dominating set, *Computer Engineering*, **35**(10), 92-94.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1989), *Introduction to Algorithms*, The MIT Press, New York.

Dai, F. and Wu, J. (2004), An Extended Localized Algorithm for Connected Dominating Set Formation in Ad hoc Wireless Networks, *IEEE Transaxtion on Paralel. Distributing*, **15**(10), 908-920.

Fomin, F. V., Grandoni, F., and Kratsch, D. (2008), Solving Connected Dominating Set Faster Than 2^n, *Algorithmica*, **52**, 153-166.

Garey, M. R. and Johnson, D. S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco.

Guha, S. and Khuller, S. (1998), Approximation algorithms for connected dominating sets, *Algorithmica*, **20**, 374-387.

Li, Y., Thai M. T., Wang, F., Yi, C. W., Wan, P. J., and Du, D. Z. (2005), On Greedy Construction of Connected Dominating Sets in Wireless Networks, *Wireless Communication Mobile Computing*, **5**, 927-932.

Min, M., Du, H., Jia, X., Huang, C. X., Huang, S. C. H., and Wu, W. (2006), Improving Construction for Connected Dominating Set with Steiner Tree in Wireless Sensor Networks, *Journalof Global Optimization*, **35**, 11-119.

Morgan, M. and Grout, V. (2008), Finding Optimal Solutions to Backbone Minimisation Problems using Mixed Integer Programming, 7th *International Network Conference*, Plymouth, United Kingdom, 53-63.

Ng, M. J. and Lu, I. T. (1999), A Peer-to-Peer Zone-based Two-level Link State Routing for Mobile Ad Hoc Networks, *IEEE Journal on Selected Area Communication*, **17**(8), 1415-1425.

Ni, S. Y., Tseng, Y. C., Chen, Y. S., and Sheu, J. P. (2002), The broadcast Storm Problem in a Mobile Ad hoc Network, *Wireless Networks*, **8**, 153-167.

Pei, G., Gerla, M., and Chen, T. W. (2000), Fisheye State Routing: A Routing Scheme for Ad hoc Wireless Networks, *IEEE International Conference on Communication*, New Orleans, LA, 70-74.

Ruan, L., Du, H., Jia, X., Wu, W., Li, Y., and Ko, K. I. (2004), A Greedy Approximation for Minimum Connected Dominating Set, *Theory of Computer*

Science, **329**, 325-330.

Sinha, P., Sivakumar, R., and Bharghavan, V. (2001), Enhancing Ad hoc Routing with Dynamic Virtual Infrastructures, 20th *Annual Joint Conference of the IEEE Computer and Communications Societies*, Anchorage, Alaska, 1763-1772.

Stojmenovic, I., Seddigh, M., Zunic, J. (2002), Dominating Sets and Neighbor Elimination Based Broadcasting Algorithms in Wireless Networks, *IEEE Transaction on Parallel Distributing*, **13**, 14-25.

Thulasiraman, K. and Swamy, M. N. S. (1992), *Graphs: Theory And Algorithms*, Wiley, New York.

Wang, L. Y., Zhang, Q., and Liu, A. M. (2009), Distributed MCDS constructing algorithm in Ad hoc networks, *AppliedComputing*, **26**, 2241-2243.

Wu, J., Li, H. (1999), On Calculating Connected Dominating Set for Efficient Routing in Adhoc Wireless Networks, 3rd *ACM International Workshop on Discrete Algorithms and Methods for MOBILE Computing and Communications*, Dallas, Texas, 7-14.

Wu, J., Dai, F. (2004), A Generic Distributed Broadcast Scheme in Ad hoc Wireless Networks, *IEEE Transaction on Computing*, **53**, 1343-1354.

Yuan, D. (2005), Energy-Efficient Broadcasting in Wireless Ad Hoc Networks: Performance Benchmarking and Distributed Algorithms Based on Network Connectivity Characterizatio, 8th *ACM International Symposium on Modeling*, Analysis and Simulation of Wireless and Mobile Systems. Montreal, Canada, 28-35.