On constructing minimum *k*-connected *m*-dominating set for robust backbone in wireless sensor networks.

안남수, 박성수
한국과학기술원 산업 및 시스템 공학과

Abstract

In wireless sensor networks, virtual backbone has been proposed as the routing infra-structure and connected dominating set has been widely adopted as virtual backbone. If we can find a connected dominating set composed of small number of sensors, more efficient routing and area monitoring can be executed. However, since the sensors in wireless sensor networks are prone to failures, recent studies suggest that, it is also important to maintain a certain degree of redundancy in the backbone.

In this paper, we both consider the routing flexibility and the fault tolerance which are hard to achieve in connected dominating set. Since the objective is constructing the backbone with a small number of sensors, this problem is referred to as the minimum *k*-connected *m*-dominating set problem. In this research, we use an integer programming approach to find an optimal solution to the problem, and to the best of our knowledge, this is the first exact approach. Some preliminary computational results are included, and extension on the research will be discussed.

1. Introduction

Recent advances in digital technologies have enabled the development of low-cost, low-power and multi-functional sensor. When the sensors are connected in a wireless manner, a new wide range of applications, such as environmental monitoring, military surveillance application, healthcare, inventory tracking, traffic control and *etc*., became possible. Generally, sensors gather environmental information within their sensing ranges and transmit the processed information over wireless networks to the other sensors or a remote running application.

Typically, wireless sensor networks (WSN) has no network infrastructure. Therefore, to send a message, although the simplest routing method is a flooding, this not only devastates batteries of the sensors but gives negative effect on the throughput of the whole network as indicated in [25].

Furthermore, if we use the flooding for broadcasting purpose, it probably causes *broadcasting storm problem* as shown in [15]. Therefore recent broadcasting, activity scheduling and area monitoring algorithms use the concept of the virtual network infrastructure, so called backbone. Backbone can be defined as a subset of sensors that can perform data communication tasks and to serve nodes that are not in the backbone [20]. Previous work [18] has shown that the backbone can reduce the routing overhead dramatically. Also, there is a vast literature about the backbone construction, and it can be found in [19].

A primary backbone concept used in the literature is a connected dominating set (CDS). The sensors in the CDS are called dominators, and the other sensors are called dominatees. The CDS have the following properties. First, each sensor in the networks is either in the dominating set or has a neighboring sensor that is in the dominating set. Second, connectivity is required among the dominators for proper protocol functioning. 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 and topology control (More comprehensive review can be found in [4].).

However, since the sensors in WSN are prone to failure, recent researches [1, 5, 12, 13] indicate it is also important to maintain a certain degree of redundancy in the CDS. Assume that we have a dominating set such that, every dominatee has at least m adjacent dominators and between any pair of dominators, there exist at least k different paths. Then, even m-1 adjacent dominators are dead, dominatee sensor still can be connected with the dominating set. Also, even k-1 dominators are failed, dominating sets still can form a CDS. This type of dominating set can enhance the fault tolerance and the routing flexibility, and it is referred to as the k-connected m-dominating set (kmCDS).

The quality of the kmCDS can be measured by the size with respect to the minimum possible size for the same network. Although many algorithms for the minimum kmCDS were suggested in [8, 16, 17, 21, 24], none of them guarantees the exact solution. To the best of our knowledge, no exact algorithm for the minimum kmCDS problem has been developed.

The main purpose of this research is, when a graph is given with k and m, designing an optimal algorithm to find a minimum kmCDS. (Note that, when k=1 and m=1, the problem is a minimum CDS problem which is one of the well-known NP-hard problems [10]).

The remainder of this paper is organized as follows. In Section 2, we discuss a mathematical formulation for the minimum *km*CDS problem. In Section 3, we show an optimal algorithm based on a combinatorial Benders' cut, and additional constraints for the minimum *km*CDS will be introduced. Section 4 contains computation results of our optimal algorithm and the following Section 5 introduces how the optimal algorithm for the *km*CDS can be used to find a maximum network lifetime. Some preliminary computational results are also included. Finally, in Section 6, we conclude this paper and discuss some future research directions.

2. Mathematical formulation

This section presents a mathematical formulation for the minimum *km*CDS problem. Before introduce our formulation, we review the objective and restrictions of the problem more in detail. The objective is clear. When a graph is given with k and m, we want to minimize the number of vertices in the dominating set. Only two restrictions exist. First restriction is, if a certain vertex is a dominatee, then it has at least m dominators. Second restriction is, between any pair of dominators there exist at least k vertex disjoint paths (Note that, since the sensors are prone to failures, we want to establish the vertex disjoint path instead of the edge disjoint path.).

To formulate the problem, decision binary variables d_i are defined as follows.

$$d_i = \begin{cases} 1, \text{if vertex } i \text{ is a dominator,} \\ 0, \text{otherwise(= vertex } i \text{ is a dominatee).} \end{cases}$$

Then the objective can be represented as follows.

$$\text{minimize} \sum_{i \in V} d_i$$

To represent the first restriction, assume that vertex i is set to dominatee (d_i=0). Then, there exist at least m adjacent dominators to the vertex i. Let the set of neighbors of a vertex i as $N(i)$.

Then,

$$\sum_{j \in N(i)} d_j \geq m(1-d_i), \forall i \in V.$$

holds.

Now the second restriction remains. However, constructing the corresponding inequalities seems challenge because the establishment of the vertex disjoint path requires prior identification of the dominators, which are the decision variables we need to find. In other words, after the dominating vertices are chosen, vertex disjoint path inequality can be established as follows.

Let S be the set of such dominators and x_e for each edge $e(=ij)\in E$. After then,

$$\sum_{i \in W} \sum_{j \in S \setminus (Z \cup W)} x_{ij} \geq 1, \quad \text{for all } s,t \in S, s \neq t \text{ and}$$
$$\text{for all } Z \subseteq S \setminus \{s,t\} \text{ with } |Z| = k-1 \text{ and}$$
$$\text{for all } W \subseteq S \setminus Z \text{ with } s \in W, t \notin W.$$

can be constructed as shown in [11].

In this research, instead of putting an effort on the direct representation of the vertex disjoint path inequality, indirect method will be adopted and discussed in Section 3. However, the following additional inequalities can be obtained due to the k-connectivity requirement.

$$\sum_{j \in N(i)} d_j \geq k d_i, \forall i \in V.$$

These inequalities show that, to form a k vertex disjoint paths, if a certain vertex i is set to dominator (d_i=1), then vertex i is adjacent to at least k different dominators.

Let the mathematical formulation composed of the listed inequalities as P and it is shown as follows.

$$P = \begin{cases} \text{minimize} \sum_{i \in V} d_i \\ \text{subject to } \sum_{j \in N(i)} d_j \geq m(1-d_i), \forall i \in V \\ \sum_{j \in N(i)} d_j \geq k d_i, \forall i \in V \\ d_i \in \{0,1\}, \forall i \in V \end{cases}$$

3. Optimal algorithm

In this section, we propose an optimal algorithm for the minimum *km*CDS problem. Our optimal algorithm asks to use combinatorial Benders' cuts, K^{th} best minimum cut identification algorithm, vertex connectivity finding algorithm and edge connectivity finding algorithm. We first review the each sub-algorithm, and then provide the optimal algorithm later.

3.1. Combinatorial Benders' cut

Combinatorial Benders' (CB) cut was first shown in [7] to handle the constraints with big-M coefficients efficiently. It divides the original problem into two problems, so called *master* problem and *slave* problem. Each problem contains the part of the constraints of the original problem. For convenience, we made a small modification on the original problem. Let *Original* be a mixed integer programming problem with the following structure:

$$Original = \begin{cases} \text{minimize } c^{\mathrm{T}} x \\ \text{subject to } Fx \le g \\ \qquad Mx + Ay \ge b \\ \qquad Dy \le e \\ \qquad x \in \{0,1\} \\ \qquad y \ge 0 \end{cases}$$

Now we split the *Original* into two sub-problems as follows:

$$Master = \begin{cases} \text{minimize } c^{\mathrm{T}} x \\ \text{subject to } Fx \le g \\ \qquad x \in \{0,1\} \end{cases}$$

$$Slave = \begin{cases} Ay \ge b\text{-}M\tilde{x} \\ Dy \le e \\ y \ge 0 \end{cases}$$

At first, we solve *Master* optimally and let *x** be the generated solution from *Master*. Then *x** be an input to *Slave*. When we solve *Slave*, two cases can happen, whether *Slave* has a feasible solution or not. If *Slave* has a feasible solution *y**, then clearly, (*x**, *y**) is an optimal solution of *Original*. If *Slave* is infeasible, *x** is infeasible for *Original*, too. In order to break the infeasibility, at least one of *x** variables need to be changed. To derive the inequality in the *x* space, let *I* and *O* be the sets of indices of *x* variables which are set to one and zero in *Master*, respectively. Then the following CB cut can be established:

$$\sum_{j \in I} (1 - x_j^*) + \sum_{j \in O} x_j^* \ge 1$$

This CB cut is added to *Master* and iterating this procedure will produce an exact solution. Interested reader can find more detailed procedure of the CB cut in [7].

Now we discuss how the CB cut can be used to solve the minimum *km*CDS problem. Since the formulation *P* guarantees the minimum *m*-dominating set, after solve *P* optimally, we can construct the graph *G'* as the following.

INPUT: *d** from *P*

Step 1. Let *V'* be the set of *i* such that $d_i^* = 1, \forall i \in V$.

Step 2. Let *E'* be the set of *e*=(*ij*) such that *i*∈*V'*, *j*∈*V'* and *e*(=*ij*) ∈*E*

OUTPUT: *G'*=(*V'*, *E'*)

According to *Menger's* theorem, if the vertex connectivity of the graph *G'* is *k*, there exist at least *k* vertex disjoint paths between any pairs of the vertices in *G'*. Thus, if the vertex connectivity of *G'* is greater than or equal to *k*, we obtained minimum *km*CDS. Otherwise, if the vertex connectivity of *G'* is less than *k*, at least one of *d** variables need to be changed in order to break the infeasibility. Note that for a given graph *G*=(*V*,*E*), the complexities of computing the vertex connectivity and edge connectivity are $O(|E|^{1/2}|V|^2)$ and $O(\min\{m^{3/2}n, n^{5/3}m\})$, respectively from [23]. Therefore, if we apply the decomposition technique to our minimum *km*CDS problem, the problem can be decomposed into two problems such that, *Master* obtains the minimum *m*-dominating set and *Slave* guarantees the *k*-connectivity.

Although *Slave* cannot be represented by the linear inequality, note that the vertex connectivity can be checked in polynomial time. Therefore, the following Figure 1 shows how the CB cut can be applied to solve the minimum *km*CDS problem.

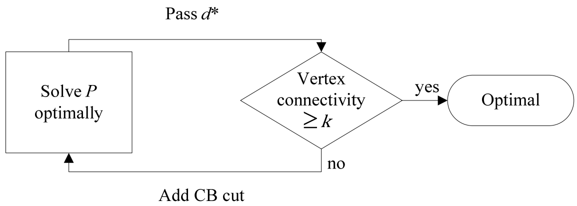

Figure 1. Application of CB cut to minimum *km*CDS problem.

3.2. *K*^th best minimum cut

In this section, we address the detailed procedure to find the K^{th} best minimum cut, and explain how it can be used to derive the stronger inequality to solve the minimum *km*CDS problem.

Before giving a detailed explanation about the algorithm for the K^{th} best minimum cut, we briefly review the classical minimum *source-sink* cut formulation suggested in [9] as follows.

For a given directed graph *G*=(*V*, *A*), there is a variable u_i for each vertex v_i∈*V* and a variable y_{ij} for each arc (v_i,v_j) ∈*A* (let *A'*={(ij):(v_i,v_j) ∈*A* }).

$$Mincut = \begin{cases} \text{minimize } \sum_{(ij) \in A'} c_{ij} y_{ij} \\ \text{subject to } u_i - u_j + y_{ij} \ge 0, \forall (ij) \in A' \\ \qquad u_{\text{sink}} - u_{source} = 1 \end{cases}$$

For a given graph with the source and sink vertices, if we solve *Mincut* optimally, minimum cut arcs which separate *V* into *S* and *V\S* can be found. Also, if the vertex *i* belong to *S*, corresponding vertex variable u_i takes zero, contrarily, if vertex *i* belong to *V\S*, corresponding vertex variable u_i takes one. Then the following simple procedure using the CB cut can be applied to find the K^{th} best minimum cut.

INPUT: *G*=(*V*, *A*) and *k*

Step 1. Construct *Mincut* and set *i*=1

Step 2. Solve *Mincut* optimally.

Step 3. If *i* is reached to *K*, stop. Otherwise, increase *i* by one.

Step 4. Add CB cut to *Mincut* to exclude out the current solution and return to Step 2.

OUTPUT: K^{th} best minimum cut

3.3. Additional constraints

Now we change our attention to the case such that, when we solve *P*(=*Master*), vertex connectivity of *G'* is strictly less than *k* (If the vertex connectivity of *G'* is greater than or equal to *k*, we obtained optimal.). Therefore, there exist two vertices *s* and *t* such that the number of vertex disjoint paths is strictly less than *k*. Clearly, for a given two vertices, the number of edge disjoint paths is always greater than or equal to the number of vertex disjoint paths. Therefore, the following two sub-cases are possible. First, the number of edge disjoint paths is equal to the number of vertex disjoint paths. Second, the number of edge disjoint paths is greater than the number of vertex disjoint paths. Figure 2 shows the second case.

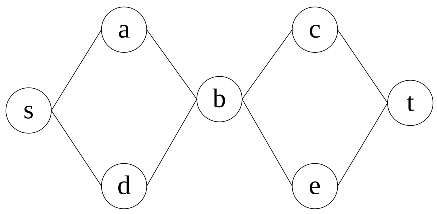

Figure 2. one s-t vertex disjoint path: sabct, two s-t edge disjoint paths: sabct, sdbet.

In any cases, if there exist *k* vertex disjoint paths, corresponding *k* edge disjoint paths can be constructed. In other words, if we fail to construct the *k* edge disjoint paths, we cannot build *k* vertex disjoint paths. In this section, we will use this property to identify the additional constraints.

Let *S* be a proper subset of *V*, such that *S* contains at least one vertex *u* that does not have any neighbor in *V**S*. Similarly, resulting *V**S* contains at least one vertex *v* that does not have any neighbor in *V*. Among the vertices in *S*, let *C*(*S*) and *C**S* be the sets of vertices which have neighbor in *V**S* and *V*, respectively.

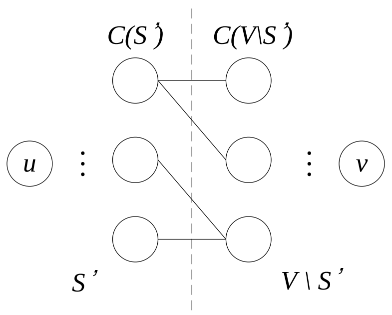

Figure 3. Illustration of condition such that *u* and *v* are in *S* and *V**S*, respectively.

The Figure 3 represents the condition. Then, clearly, at least one vertex in each *C*(*S*) and *C*(*V**S*) must be selected as a dominator. Otherwise, *u* and *v* cannot be dominated by any neighboring dominators. Thus between such *S* and *V**S*, at least *k* vertex disjoint paths exist. Therefore, corresponding *k* edge disjoint paths exist.

Now the following additional constraints can be added to *P*. Let *S'* be the proper subset of *V* such that *S'* and *V**S'* contain at least one vertex *u* and *v* with the aforementioned property.

Also, let $k_{s'}$ and $k_{v\setminus s'}$ be the minimum numbers of vertices in *C*(*S'*) and *C*(*V**S'*) to form *k* different edges. Then,

$$\sum_{i \in C(S')} d_i \geq k_{s'}, \forall S' \in V$$

$$\sum_{j \in C(V \setminus S')} d_j \geq k_{c \setminus s'}, \forall S' \in V$$

can be established. To guarantee the connectivity between the vertices in *C*(*S'*) and *C*(*V**S'*), the following constraints can also be added (let $E' = \{(ij) : (v_i, v_j) \in E\}$).

$$\sum_{j \in C(V \setminus S') \text{ and } (ij) \in E'} d_j \geq d_i, \forall S' \in V \text{ with } i \in C(S')$$

$$\sum_{i \in C(V \setminus S') \text{ and } (ij) \in E'} d_i \geq d_j, \forall S' \in V \text{ with } j \in C(V \setminus S')$$

However, since the number of constraints can be huge, adding these constraints to *P* all at once seems impractical. Therefore, in this research, we propose a scheme which generates the violated constraints by the *d** of *P*. Now the scheme follows.

INPUT: *G'*=(*V'*,*E'*) generated from *d** of P

Step 1. Identify the components of *G'* such that each component has *k* edge connectivity.

Step 2. Construct the *Mincut* and set *i* as one.

Step 3.

For all i^{th} component **do**

Step 3.1. Set u_i variables of i^{th} component as 0.

Step 3.2. Set u_i variables of the other components as 1.

Step 3.3. Solve *Mincut* optimally.

while Generated *S* and *V**S* do not contain aforementioned *u* and *v* **do**

Step 3.3.1. Add CB cut to *Mincut* to exclude out the current solution.

Step 3.3.2. Solve *Mincut* optimally.

end while

Step 3.4. Drop the fixed u_i variables and CB cuts.

Step 3.5. Identify the corresponding additional constraints and increase *i* by one.

end for

OUTPUT: Violated constraints

3.4. Flowchart of an optimal algorithm

In the previous sections, we discussed several sub-problems and sub-algorithms to design the optimal algorithm for the minimum *km*CDS problem. Now, the overall process of the optimal algorithm can be summarized as shown in Figure 4.

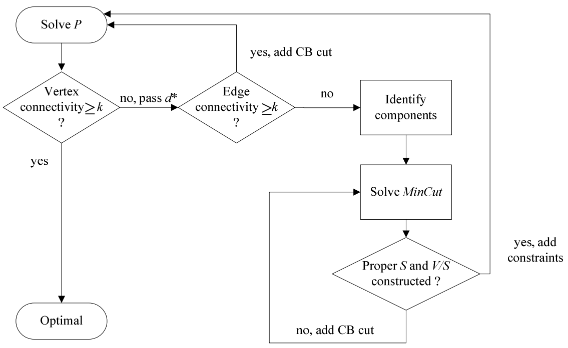

Figure 4. Flow chart of an optimal algorithm for the minimum *km*CDS problem.

5. Computational results

In this section, we evaluated the performance of the optimization algorithm, which is suggested in this research. All graphs in this section obtained by running the graph generator code in the web, http://www.brandonparker.net/graph_gen.php. The code takes the number of vertices as an input, and then it generates four types of connected graphs: sparse, medium, dense and complete graph. Performance of the algorithm was tested in these graphs, and the algorithm was implemented in C++ and ILOG CPLEX 11.0 was used as optimization software. All experiments were run on AMD Athlon™ 64 X2 Dual Core (2.70 GHz) with 2GB Ram, and measurement unit of the algorithm runtime is second.

For each type of the graphs, we change the number of vertices from 10 to 50 with an increment 10. Two numbers in each cell of the table represent the number of vertices and the consumed algorithm runtime in second, respectively. In some cases, *km*CDS does not exist for a given graph and these cases are denoted by N/A sign.

Now the results are following as shown in Tables 1, 2, 3 and 4.

Table 1. Minimum *km*CDS in sparse graph.

$\|V\|$	k=1, m=2	k=1, m=3	k=2, m=1
10	6, 0.0	7, 0.0	4, 0.0
20	12, 0.0	16, 0.0	9, 0.0
30	19, 0.0	23, 0.0	N/A
40	25, 2.0	30, 0.0	17, 0.0
50	31, 1.0	39, 0.0	N/A

Table 2. Minimum *km*CDS in medium graph.

$\|V\|$	k=1, m=2	k=1, m=3	k=2, m=1
10	3, 0.0	4, 0.0	3, 0.0
20	3, 0.0	5, 0.0	3, 0.0
30	2, 0.0	5, 0.0	3, 0.0
40	4, 0.0	5, 0.0	3, 0.0
50	5, 0.0	6, 0.0	3, 0.0

Table 3. Minimum *km*CDS in dense graph.

$\|V\|$	k=1, m=2	k=1, m=3	k=2, m=1
10	3, 0.0	4, 0.0	3, 0.0
20	3, 0.0	4, 0.0	3, 0.0
30	3, 0.0	4, 0.0	3, 0.0
40	3, 0.0	4, 0.0	3, 0.0
50	3, 0.0	5, 0.0	3, 0.0

Table 4. Minimum *km*CDS in complete graph.

$\|V\|$	k=1, m=2	k=1, m=3	k=2, m=1
10	2, 0.0	3, 0.0	3, 0.0
20	2, 0.0	3, 0.0	3, 0.0
30	2, 0.0	3, 0.0	3, 0.0
40	2, 0.0	3, 0.0	3, 0.0
50	2, 0.0	3, 0.0	3, 0.0

As tables indicate, our algorithm showed good performance in all cases, which means, identified the optimal solution in a reasonable amount of time.

6. Extension to maximum lifetime of the networks problem

In this section, we explain the application of the optimal algorithm for the maximum network lifetime problem and similar applications can be found in [2, 6, 14, 22].

Assume that for a given graph G=(V, E), let b_v represent the amount of time resources for vertex v. Now we formally define the maximum network lifetime problem. Assume that schedule S is a set of pairs (d_1,t_1), (d_2,t_2), …, (d_m,t_m), where d_i is a k-connected m-dominating set and t_i is a time duration of the vertices in d_i. Then the lifetime of the schedule S is defined as $L(S) := \sum_{i=1}^{m} t_i$, and the maximum network lifetime problem asks *maximize* L(S) while satisfying the available time limit for each vertex ($\sum_{i:v\in d_i} t_i \le b_v, v \in V$).

To enhance the understanding of the maximum network lifetime problem, the example is illustrated as shown in Figure 5.

.

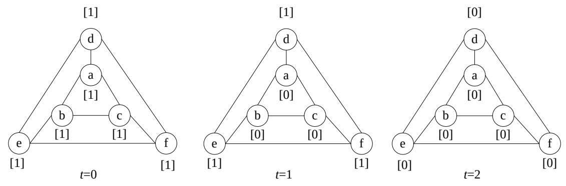

Figure 5. Optimal schedule S with $L(S)$=2.

Assume that the requirement for the d_i is 2-connected 1-dominating set, then the length of the optimal schedule is two and schedule is ((a,b,c), 1) and ((d,e,f), 1). Note that the 2-connected 1-dominating set cannot be established after t=2.

In this section, we want to design an optimal algorithm for the maximum network lifetime problem, and actually, it can be formulated as a linear programming. Assume that we enumerated all *km*CDSs for a given graph, and we numbered it from 1 to p. Let t_j indicate the active time for the j^{th} dominating set.

Then the linear programming which maximizes the network lifetime can be formulated as follows.

$$MNLP = \begin{cases} \text{maximize} \sum_{j=1}^{p} t_j \\ \text{subject to} \sum_{j=1}^{p} a_{ij} t_j \le b_i, \forall i = 1, \ldots, |V| \\ t_j \ge 0, \forall j = 1, \ldots, p \end{cases}$$

, where a_{ij} takes one if vertex i is included in j^{th} dominating set, otherwise it takes zero.

However, constructing the coefficient matrix A in full seems impractical since the number of dominating sets can be huge. Therefore, in this research, we used the standard column generation approach to handle the dominating set implicitly rather than explicitly, and each column (=dominating set) is generated by the optimal algorithm for the minimum *km*CDS problem which is suggested in the previous section. More interested readers can find the detailed process of the column generation procedure in [3].

Experimental environment is exactly the same as in Section 5. In this paper, we assume that the requirement for the dominating set is k=2 and m=1 and b_v is randomly assigned between 10 to 20. Preliminary computational results are presented as shown in Table 5.

Table 5. Maximum network lifetime with random amount of resources

|V|	Sparse graph		
	Columns	Optimal	Runtime
10	3	12	0.0
20	4	21	0.03
	Medium graph		
	Columns	Optimal	Runtime
10	15	39	0.063
20	47	88.9	0.11
	Dense graph		
10	16	49	0.063
20	27	101.33	0.047

7. Conclusion and future works

In this research, we proposed an optimal algorithm for the minimum *km*CDS problem which can be used to construct the virtual backbone in WSN. As far as we concerned, this is the first exact approach to address the problem. Our optimal algorithm uses several sub-algorithms, such as finding a K^{th} best minimum cut, identification of vertex connectivity and edge connectivity of a given graph. We also proposed a simple algorithm for the K^{th} best minimum cut problem using a combinatorial Benders' cut. Computational results show that our algorithm found the optimal solutions in a relative short amount of time.

Other applications of the minimum *km*CDS can be found in a maximum network lifetime problem. To address the lifetime problem, we proposed a column generation procedure to find a maximum network lifetime and preliminary computational results are included. However, our algorithms still have a room for the performance improvement and we are working on this issue. Also, even at this point, we are unsure about the possibility of expressing all the requirements of the *km*CDS in one formulation. If this is a possible task, then studying the basic polyhedral structure arising from this formulation can be a future research topic.

References

[1] P. Basu and J. Redi, Movement control algorithms for realization of fault tolerant ad hoc robot networks, *IEEE Networks*, 18(4):36-44, 2004.

[2] P. Berman, G. Calinescu, C. Shah and A. Zelikovsky, Power efficient monitoring management in sensor networks, *Wireless communication and networking conference*, 4:2329-2334, 2004.

[3] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, Massachusetts, 1997.

[4] J. Blum, M. Ding, A. Thaeler and X. Cheng, *Connected dominating set in sensor networks and MANETs*, Handbook of combinatorial optimization(Vol. 5), D. Z. Du and P. Pardalos(Editors), 329-369, Kluwer Academic Publishers, 2004.

[5] J. L. Bredin, E. D. Demaine, M. Hajiaghayi and D. Rus, Deploying sensor networks with guaranteed

capacity and fault tolerance, *6th ACM international symposium on mobile ad hoc networking and computing*, 309-319, 2005.

[6] M. Cardei, M. T. Thai, Y. Li and W. Wu, Energy-efficient target coverage in wireless sensor networks, *IEEE INFOCOM*, 3:1976-1984, 2005.

[7] G. Codato and M. Fischetti, Combinatorial Benders' cuts for mixed-integer linear programming, *Operations Research*, 54(4):756-766, 2006.

[8] F. Dai and J. Wu, On Constructing k-Connected k-Dominating Set in Wireless Network, Journal of parallel and distributed computing, 66:947-958, 2006.

[9] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, 1962.

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

[11] M. Grotschel, C. L. Monma, and M. Stoer, *Design of survivable networks*, Handbook in Operations Research and Management Science 7, M. O. Ball, C. L. Monma and G. Nemhauser(Editors), 617-671, North-Holland, 1995.

[12] H. Koskinen and J. Karvo, On improving connectivity of static ad-hoc networks by adding nodes, *4th annual Mediterranean workshop on ad hoc networks*, 169-178, 2005.

[13] F. Kuhn, T. Moscibroda and R. Wattenhofer, Fault-tolerant clustering in ad hoc and sensor networks, *26th international conference on distributed computing systems*, 2006.

[14] T. Moscibroda and R. Wattenhofer, Maximize the Lifetime of Dominating Sets, *IEEE International Parallel and Distributed Processing Symposium*, 13:242.2, 2005.

[15] S. Y. Ni, Y. C. Tseng, Y. S. Chen and J. P. Sheu, The broadcast storm problem in a mobile ad hoc network, *Wireless networks*, 8(2-3):153-167, 2002.

[16] W. Shang, F. Yao, P. Wan and X. Hu, Algorithms for Minimum m-Connected k-Dominating Set Problem, *LNCS*, 4616:182-190, 2007.

[17] W. Shang, F. Yao, P. Wan and X. Hu, On minimum m-connected k-dominating set problem in unit disc graphs, *Journal of combinatorial optimization*, 16(2):99-106, 2008.

[18] P. Sinha, R. Sivakumar and V. Bharghavan, Enhancing ad hoc routing with dynamic virtual infrastructures, *20th annual joint conference of the IEEE computer and communications societies*, 3:1763-1772, 2001.

[19] I. Stojmenovic, J. Wu, *Broadcasting and Activity Scheduling in AD HOC Networks*, Mobile ad hoc networking, S. Basagni, M. Conti, S. Giordano, I. Stojmenovic(Editors), 205-229, IEEE, Inc., 2004.

[20] D. S. Ryl, I. Stojmenovic, J. Wu, *Energy-Efficient Backbone Construction, Broadcasting, and Area Coverage in Sensor Networks*, Handbook of sensor networks, I. Stojmenovic(Editor), 343-380, John Wiley & Sons, Inc., 2005.

[21] M. T. Thai, N. Zhang, R. Tiwari and X. Xu, On approximation algorithms of k-connected m-dominating sets in disk graphs, *Theoretical Computer Science*, 385(1-3):49-59, 2007.

[22] M. T. Thai, F. Wang, D. H. Du, X. Jia, Coverage problems in wireless sensor networks: designs and analysis, *International Journal of Sensor Networks*, 3(3):191-200, 2008.

[23] K. Thulasiraman and M.N.S. Swamy, *Graphs: Theory And Algorithms*, Wiley, New York, 1992.

[24] F. Wang, M. T. Thai, D. Z. Du, On the construction of 2-connected Virtual Backbone in Wireless Network, *IEEE Transactions on Wireless Communications*, 8(3):1230-1237, 2007.

[25] Y. Wu and Y. Li, Construction algorithms for k-connected m-dominating sets in wireless sensor networks, *9th ACM international symposium on mobile ad hoc networking and computing*, May:83-90, 2008.