International Journal of Management Science
Vol 17, No 2, November 2011

An Optimization Algorithm for the Maximum Lifetime Coverage Problems in Wireless Sensor Network

Namsu Ahn
Daejeon Center, Defense Agency for Technology and Quality

Sungsoo Park*
Department of Industrial and Systems Engineering, KAIST

(Received: March 11, 2011 / Revised: August 25, 2011 / Accepted: September 14, 2011)

ABSTRACT

In wireless sensor network, since each sensor is equipped with a limited power, efficient use of the energy is important. One possible network management scheme is to cluster the sensors into several sets, so that the sensors in each of the sets can completely perform the monitoring task. Then the sensors in one set become active to perform the monitoring task and the rest of the sensors switch to a sleep state to save energy. Therefore, we rotate the roles of the active set among the sensors to maximize the network lifetime. In this paper, we suggest an optimal algorithm for the maximum lifetime coverage problem which maximizes the network lifetime. For comparison, we implemented both the heuristic proposed earlier and our algorithm, and executed computational experiments. Our algorithm outperformed the heuristic concerning the obtained network lifetimes, and it found the solutions in a reasonable amount of time.

Keywords: Wireless Sensor Network, Maximum Lifetime Coverage Problem, Optimal Algorithm, Integer Programming, Column Generation

1. Introduction

Due to the recent advances in digital technologies, developments of low-cost and multi-functional sensors become possible. Sensor is a small digital device equipped with sensing unit, transceiver, processing unit, storage unit and battery, and it can be used to monitor their environment such as temperature, light, sound and humidity. A

* Corresponding author, E- mail: sspark@kaist.ac.kr

large number of sensors collaborate using wireless communication, and the sensors gather/process/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). WSN has many applications, such as biological detection (*e.g.* habitat monitoring), environmental monitoring (*e.g.* forest-fire detection), healthcare (*e.g.* patient status monitoring), home appliance, inventory tracking and military (*e.g.* battlefield monitoring). Some are used in real life and others are in development stages.

As discussed in [15], energy density of the battery has been doubled in every 5 to 20 years. Compared to the rapid advances in digital technologies, advances in battery technologies seem to be quite slow. Also, in some applications of WSN, targets may be located in a dangerous or remote area. Thus replacing batteries is impractical in many WSN applications, and energy efficiency is one of the important topics in WSN. Therefore prolonging the lifetime of the network is one of the critical objectives in the network design.

Possible major states of a sensor in WSN can be either active or sleep. The active state consists of the following sub-states: transmit signal state, receive signal state and state which is not engaged in transmitting or receiving signal. As discussed in [27], the energy consumption ratio between the active state and the sleep state is almost as high as 100. Therefore scheduling the sensors' activities according to the following simple scheme can save great amount of battery resources. If the sensor is necessary to perform the monitoring task, the sensor goes into an active state. On the other hand, if the sensor is not necessary to perform the monitoring task, it goes into a low-battery sleep state to save the energy for future use. This scheme assumes that we disperse large population of sensors than the minimum number of sensors required to perform the monitoring task. Since the cost of a sensor is low due to the advances in digital technologies and mass production, this scheme makes sense.

Now, a scheduling mechanism based on this characteristic of a sensor can be described as follows. Cluster the sensors into several sets, such that sensors in each of the sets can completely perform the monitoring task. Then, these sets are activated successively. We define the active set as a set of sensors which perform the monitoring task completely. Therefore, at any moment, only the sensors in one active set

go into an active state and perform the monitoring task. On the contrary, all the other sensors, which belong to the non-active sets, are in the low-energy sleep state. Therefore, to maximize the lifetime of the network, it is critical to rotate the roles of the active set among the sensors in the network. Clustering the sensors has proven to be one of the successful strategies when handling the complexity of WSN [23], and it helps efficient usage of another scarce resource such as bandwidth.

When we model the network as a graph $G = (V, E)$, each sensor of the network can be represented as a vertex and there exist an edge between two vertices u and v if and only if u and v are located within the communication range. Thus, aforementioned active set may be modeled as the *dominatingset* considered in graph theory. In graph theory, a dominating set D of G is a subset of vertices such that each $v \in V$ is either in D or has a neighbor in D. Since the neighboring sensors are close enough to take over each other's sensing task, at any time, only the sensors in a dominating set are asked to be active. Clearly, the lifetime of a dominating set to perform the monitoring task is equal to the minimum of the lifetimes of the sensors (vertices) in the dominating set.

Let S be a set of pairs $(D_1, t_1), \cdots, (D_k, t_k)$, where D_i denotes i^{th} dominating set and t_i represents the lifetime of the dominating set D_i. Clearly, the maximum lifetime coverage (MLC) problem asks for the maximum length of the schedule, $\sum_{i=1}^{k} t_i$, while satisfying the available resource limit for each vertex.

In this research, we assumed that there exist enough number of sensors than the minimum required to perform the monitoring task and each sensor can have a different amount of battery capacity. The purpose of this paper is finding a schedule that maximizes the lifetime of the network by solving the MLC problem.

Note that, depending on the limit of the channel bandwidth and the requirement of the sensors' connectivity, various network lifetime coverage models have been suggested and the comprehensive review can be found in [28].

To improve the lifetime of the network, many dominating set based models and algorithms have been suggested. Recent research efforts can be found in [4, 10, 22, 24]. In [26], the authors implicitly introduced the MLC problem and later, the authors in [2] formally defined the MLC problem and developed a performance-guaranteed approximation algorithm with an approximation guarantee of $O(logn)$, where n is the number of sensors. The authors in [23] performed extensive research on the MLC

problem and introduced some variants of the problem. Also, they suggested a randomized algorithm which guarantees $O(log(b_{\max}n))$ approximation ratio, where $b_{\max}$ is the largest amount of the battery capacities among the sensors in the network and n is the number of sensors. In [5], the authors proposed an integer programming formulation, which may be used to solve the MLC problem exactly, and developed two heuristics. However, we will discuss later that the proposed formulation is an incomplete one and we suggest some corrections. Recently, the authors in [8] proposed an exact algorithm which can be used to solve the MLC problem defined using sensors and targets instead of graph. However, their algori-thm consists of so many complicated steps. Also, in the first step, the algorithm need to generated all dominating set which seems impractical. Although maximizing the lifetime of a network is an important issue, to the best of our knowledge, no other exact algorithm can be found.

In this paper, we designed an optimal algorithm for the MLC problem, and our algorithm uses the output of the two heuristics, one is developed in [5] and the other is a newly suggested one in this research. Extensive computational experiments show that our algorithm generates an optimal solution in a reasonable amount of time.

The rest of the paper is organized as follows. In Section 2, we give a detailed explanation of the MLC problem defined on a graph. In Section 3, we propose an optimal algorithm based on a mathematical formulation, and several techniques will be introduced to accelerate the convergence speed to an optimal solution. The following Section 4 contains computational experiments, and finally, Section 5 concludes this paper.

2. Problem Description

In this section, we describe the MLC problem and introduce notation which will be used throughout this paper. For convenience, we first explain the problem using the graph, then the problem will be extended to other sensor network applications, such as sensor network for target coverage or sensor network for area coverage.

We model the network as a graph $G = (V, E)$, and each sensor of the network is represented as a vertex $v \in V$ and there exists an edge $e \in E$ between two vertices u

and v if the two sensors are located within a communication range. In this graph, we only consider the undirected edges, which means, if a vertex u can send a message to a vertex v using the edge e, then v can send a corresponding message to u over the same edge. Let the set of neighbors of a vertex i as $N(i)$ and $N[i] := N(i) \cup \{i\}$. Here b_v represents the maximum lifetime of vertex v, which means, vertex v can stay in a network for at most b_v time units (b_v can be regarded as the initial battery supply of sensor v).

Now, to improve the network lifetime, we need to schedule the sensors' activities which alternate between the active state and the sleep state. Assume that schedule S is a set of pairs $(D_1, t_1), (D_2, t_2), \cdots, (D_k, t_k)$, where D_i is a dominating set and t_i is an active time duration of the vertices in D_i. Clearly, the lifetime of the schedule S is defined as $L(S) := \sum_{i=1}^{k} t_i$, and the MLC problem requires *maximize* $L(S)$ while satisfying the available resource limit for each vertex, $\sum_{i:v \in D_i} t_i \le b_v, \forall v \in V$.

To facilitate the understanding of the MLC problem, suppose that the network is given as shown in Figure 1. The numbers next to each vertex indicate b_v, and optimal schedule of the network is given in Table 1.

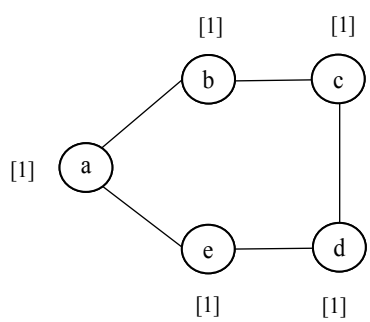

Figure 1. Example of Maximum Lifetime Coverage Problem

Table 1. Optimal Schedule of Figure 1

Time slot	0~0.5	0.5~1.0	1.0~1.5	1.5~2.0	2.0~2.5
Dominating set	(a, c)	(b, d)	(c, e)	(d, a)	(e, b)

Since we cannot construct the dominating set after 2.5^{th} time slot due to the depletion of $b_v(\forall v \in V)$, the maximum length of the schedule is 2.5. One of the main

characteristics of the MLC problem is, a sensor can be included in several dominating sets. Suppose that the sensor can participate in only one dominating set, then optimal schedule will be ((a, c), 1) and ((b, d), 1), and the maximum network lifetime will be 2 for the example of Figure 1.

The MLC problem can be extended to the following sensor network applications. Assume that we need to monitor a set of targets with known locations and we spread a large number of sensors in the proximity of the targets. This application arises when the targets are located in a hostile or dangerous area. In this application, we modify the definition of the dominating set as a set of sensors, which can cover the targets. In another application, we need to cover an area instead of the targets. To represent the area, we first divide the area into smaller square fields, and then regard each field as a target. Similarly, the dominating set will be a set of sensors, which can cover the collection of the fields. In both types of the applications, the roles of the dominating sets are asked to be rotated among the sensors in the network to improve the lifetime of the network.

Lastly, the following proposition explains that the set of minimal dominating sets (= removing any vertex in the set fails to form a dominating set) are enough to construct the schedule S.

Proposition 1: *Set of minimal dominating set are enough to construct the schedule S.*
***Proof*:** Assume that the schedule S consists of some non-minimal dominating sets and the network lifetime is given as $L(S)$. For each non-minimal dominating set, we can remove some vertices, which are not necessary to form the minimal dominating set. Let the schedule S' consists of the minimal dominating sets from S. Since the minimal dominating set is still form the dominating set, the modified schedule S' has the same network lifetime as $L(S)$.

Proposition 1 is used to find a diminating set in a newly developed heuristic in this research which will be explained in subsection 3.2.

3. Optimal Algorithm

Although constructing a schedule S which maximizes the network lifetime is an

important research issue, few research efforts for the exact algorithm can be found. To the best of our knowledge, an integer programming formulation suggested in [5] is the only method to address the MLC problem exactly. However, the formulation is an incomplete one and the corrections will be discussed in this section. In this section, we introduce a new mathematical formulation and an optimal algorithm which is based on the standard column generation technique. To accelerate the convergence speed to an optimal solution, several techniques are applied and it will be introduced also in this section.

3.1 Mathematical Formulation

To address the MLC problem using a mathematical formulation, the authors in [5] defined the decision variables as the following.

$$t_j = \textit{time allocated for dominating set } j.$$

$$y_{ij} = \begin{cases} t_j, \textit{ amount of time for vertex } i \textit{ spent in dominating set } j, \\ 0, \textit{ vertex } i \textit{ is not included in dominating set } j. \end{cases}$$

Then they set an upper bound p for the number of dominating sets. Note that, the original formulation was given for, so called, maximum set covers problem. However, we changed the notation a little bit to explain it in the context of the MLC problem. Now the formulation was given as the following.

$$\textit{Maximize} \sum_{j=1}^{p} t_j \tag{1}$$

$$\textit{Subject to}$$

$$\sum_{j=1}^{p} y_{ij} \le b_i,\ i = 1, \cdots, |V|, \tag{2}$$

$$\sum_{i' \in N[i]} y_{i'j} \ge t_j,\ i = 1, \cdots, |V|;\ j = 1, \cdots, p, \tag{3}$$

$$y_{ij} = 0 \textit{ or } t_j,\ i = 1, \cdots, |V|;\ j = 1, \cdots, p. \tag{4}$$

However, this formulation has two major drawbacks. First, a bound p probably too large to handle it explicitly. Although no research can be found on the upper bound of the number of dominating sets for a given graph, since a maximal indepen-

dent set is a dominating set, we can estimate the size of p by using the bound on the number of maximal independent sets. In [17], it is shown that the bound for the number of maximal independent sets is $3^{|V|/3}$, where $|V|$ is the number of vertices. Therefore, the size of p can be exponential and the corresponding mathematical formulation probably requires huge numbers of variables and constraints. Second drawback is, the formulation contains the non-linear expressions (4). Generally, it is known that the non-linear integer programming is much harder than the linear integer programming. Therefore, one of the widely used techniques to handle the non-linear programming is, if possible, to transform the non-linear constraints into the linear constraints.

Since the purpose of the research in [5] is the development of a heuristic which uses the output of the linear programming relaxation, the authors replaced constraints (4) with linear expressions $0 \le y_{ij} \le t_i$, $i = 1, \cdots, |V|$; $j = 1, \cdots, p$.

Actually, if we introduce new binary variables x_{ij}, non-linear constraints (4) can be linearized by replacing them with the following additional constraints. Suppose that M denotes a large positive number.

$$x_{ij} = \begin{cases} 1, & \textit{If vertex } i \textit{ is included in dominating set } j, \\ 0, & \textit{otherwise}. \end{cases}$$

$$y_{ij} \le M x_{ij},\ i = 1, \cdots, |V|;\ j = 1, \cdots, p, \tag{5}$$

$$y_{ij} \le t_j,\ i = 1, \cdots, |V|;\ j = 1, \cdots, p, \tag{6}$$

$$t_j - y_{ij} + M x_{ij} \le M,\ i = 1, \cdots, |V|;\ j = 1, \cdots, p. \tag{7}$$

If x_{ij} is one, y_{ij} equals to t_j from the constraints (6) and (7). Otherwise, if x_{ij} is zero, y_{ij} takes zero from the constraints (5). However, even this revised mixed integer linear programming formulation still has the same drawback. It has huge numbers of constraints and variables.

If we can enumerate all dominating sets for a given graph, the MLC problem can be formulated as a linear program with $|V|$ number of constraints and a huge number of variables. Suppose that we identified all minimal dominating sets, and numbered them from 1 to p. Then, we have to maximize the network lifetime by assigning

the active time for each dominating set while satisfying the maximum time limit of each vertex. Let t_j represent the active time for j^{th} dominating set. Then the linear programming formulation for the MLC problem *P* can be given as follows.

$$Maximize \sum_{j=1}^{p} t_j \tag{8a}$$

$$Subject\ \ to$$

$$\sum_{j=1}^{p} a_{ij} t_j \le b_i, i = 1, \cdots, |V|, \tag{8b}$$

$$t_j \ge 0,\ j = 1, \cdots,\ p. \tag{8c}$$

, where a_{ij} is one if vertex *i* is included in the dominating set *j*, zero otherwise.

Note that, although our formulation uses an exponential number of variables, the number of constraints is $|V|$ and there is no integer restriction on the variables. As noted before, since the number of dominating sets (= columns) can be huge, constructing the coefficient matrix *A* in full is impractical. Therefore, the dominating sets need to be dealt implicitly rather than explicitly, and this can be achieved by using the standard column generation procedure.

Generally, when we need to solve the linear programming problem which has an exponential number of variables, instead of including all the variables all at once, we can generate the variable on demand. In [11], the authors suggested handling the variables of a multicommodity flow problem implicitly. After then, many researches are performed to use the column generation as a method to handle a huge number of variables. These approaches can be found in [9], [13, 14]. Nowadays, column generation is widely used to handle many difficult problems arising in crew scheduling, vehicle routing, and *etc* [18].

When we use the column generation procedure to solve *P*, we generate the columns of *A* as needed rather than in advance. Suppose that, we start with a manageable part of the columns of *A*, which may be the dominating sets obtained using the heuristic in [5]. Since *P* consists of a small part of the columns, we call it the restricted master problem. We first solve the restricted master problem optimally using the simplex method, then we can have a basis matrix and a basic feasible solution. Now we need to perform the next iteration of the revised simplex method.

At this moment, instead of computing the reduced cost of every dominating set, we solve another optimization problem to identify the most profitable dominating set for the restricted master problem. We call this optimization problem a subproblem. When we solve the subproblem, the following two cases are possible. First case is, the maximum of the subproblem is less than or equal to zero, which means, all the reduced costs of the dominating sets are nonpositive. Then we stop the procedure and the basic feasible solution in the restricted master problem is an optimal solution. Second case is, the maximum of the subproblem is greater than zero, which implies we obtained a dominating set that can improve the current objective function of the restricted master problem. Then, this dominating set is added to the restricted master problem as a new column. Now we solve the enlarged restricted master problem again. This procedure is continued until the maximum of the subproblem is less than or equal to zero. More interested readers can find the detailed process of the column generation procedure in [3].

Now we discuss the subproblem in more detail. Let π_i be the dual variable associated with the i^{th} constraint of (8b). Also, suppose that x_i is the binary decision variable denoting whether vertex i is included in the dominating set or not. Then the subproblem which identifies the minimum weight dominating set can be formulated as the following. Note that, although the original objective function for the subproblem is $maximize\ 1-\sum_{j=1}^{|V|}\pi_i x_i$, we replace it with $minimize \sum_{j=1}^{|V|}\pi_i x_i$.

$$Minmize \sum_{j=1}^{|V|}\pi_i x_i \tag{9}$$

$$Subject\ \ to$$

$$\sum_{j\in N[i]} x_j \geq 1,\ i=1,\cdots,|V|. \tag{10}$$

$$x_i \in \{0,\ 1\},\ i=1,\cdots,|V|. \tag{11}$$

Constraints (10) indicate that, according to the definition of the dominating set, every vertex needs to be covered at least once by itself or adjacent vertices.

However, the minimum dominating set problem is a well-known NP-hard problem [12]. Thus, assuming that $P \neq NP$, we cannot expect to find a polynomial time algorithm for this problem. Therefore, in this research, we first try to find the domi-

nating set which improves the objective function of the restricted master problem using a heuristic method instead of solving the subproblem optimally. If the heuristic succeeds to identify a profitable dominating set, the set is added to the restricted master problem as a new column. Otherwise, if the heuristic fails to identify a profitable dominating set, we solve the minimum weight dominating set problem exactly using the above formulation. Simialrly, if we can obtain the profitable dominating set using the formulation, this set is added to the restricted master problem. Otherwise, lastly, we stop the procedure since we obtained an optimal solution.

3.2 Heuristic to Compute a Dominating Set in a Subproblem

In this section, we propose a simple heuristic algorithm which can be used to identify a profitable dominating set in a subproblem. Generally, constructing a *mini mal dominating set* can be done easily. We first prepare an empty set, then choose a vertex and add it to the set. If the set is a dominating set, we stop the procedure and output the set. Otherwise, we choose another vertex, which is not included in the set and add it to the set. This procedure is continued until the set satisfies the dominating set requirement. Clearly, overall performance of this procedure is significantly affected by the sequence we select the vertices. Therefore when we select a vertex, we first calculate the scores for the vertices which are not assigned to the set, and then the highest score vertex is selected.

Now we discuss the score calculation rule for the vertex which is not included in the set. When a new vertex i is added to the set, some vertices will be dominated more than once due to the existing vertices in the set. We define the number of vertices dominated again by the addition of the vertex i to the set as redundancy of the set. Similarly, when a new vertex is added to the set, some vertices may be dominated exactly once since no existing vertices in the set dominate the vertices. We call the number of vertices dominated exactly once due to the addition of the vertex as new covering of the set. Intuitively, when we assign a new vertex to the set, we expect that the vertex is used to form the dominating set in an efficient manner. Therefore when we add the vertex to the set, we want to minimize the number of redundancies and maximize the number of new coverings. Also, since the objective of the subproblem is finding a dominating set whose $\sum_{i=1}^{|V|} \pi_i x_i$ is a minimum, we want to choose the vertex whose corresponding value of π_i is as small as possible. There-

fore the following score function for vertex i is suggested (To avoid division by zero, one is added to the divisor).

$$f(i) = \frac{new\ covering\ by\ adding\ vertex\ i - redundancy\ by\ adding\ vertex\ i}{\pi_i + 1} \tag{12}$$

However, note that this procedure does not guarantee the optimal solution. Therefore even if the dominating set which improves the objective function of the restricted master problem exists, our heuristic may fail to generate the dominating set. To address this case, if the heuristic fails to obtain the profitable dominating set, we solve the minimum weight dominating set problem optimally using the mathematical formulation.

3.3 Stabilized Column Generation

When the column generation is applied to solve the problem which has an exponential number of variables, one of the difficulties is a bad convergence behavior of the dual variable values in some problems. It has been observed that, during the column generation procedure, the values of the dual variables do not smoothly converge to their optimal values and sometimes dual values oscillate and follow no pattern. Therefore, techniques restricting the range of dual variables to obtain smooth convergence to their optimal values were first suggested in [1], and similar approaches can be found in [20, 21]. Overview of the stabilizing techniques in column generation can be found in [19].

In this research, we adopted the stabilization technique suggested in [25]. To apply the stabilization technique, we solve the following stabilized problem $P(\bar{\pi}, \varepsilon)$ instead of P using the column generation. Here $\bar{\pi}$ is an educated guess of π and ε is a parameter which has a small positive value.

$$Maximize \sum_{j=1}^{p} t_i - \sum_{i=1}^{|V|} \bar{\pi}_i w_i + \sum_{i=1}^{|V|} \bar{\pi}_i z_i \tag{13a}$$

Subject to

$$\sum_{j=1}^{p} a_{ij} t_j - w_i + z_i \le b_i, \ i = 1, \cdots, |V|, \tag{13b}$$

$$w_i \le \varepsilon, \ i = 1, \cdots, |V|, \tag{13c}$$

$$z_i \le \varepsilon, \ i = 1, \cdots, |V|, \tag{13d}$$

$$w_i,\ z_i \geq 0,\ i = 1, \cdots,\ |V|, \tag{13e}$$

$$t_j \geq 0,\ i = 1, \cdots,\ p. \tag{13f}$$

Note that, in $P(\bar{\pi},\ \varepsilon)$, we have introduced two sets of artificial variables, w and z. Let ω and ς represent the dual variables corresponding to the constraints (13c) and (13d), respectively. Considering the dual problem of $P(\bar{\pi},\ \varepsilon)$, we can see that the range of π_i (dual variables associated to the i^{th} constraint of (13b)), are now restricted by the following constraints.

$$\bar{\pi}_i - \varsigma_i \leq \pi_i \leq \bar{\pi}_i + \omega_i,\ i = 1, \cdots, |V|. \tag{14}$$

Therefore, the values of π is not likely to oscillate much during the column generation procedure. Note that, $P(\bar{\pi},\ \varepsilon)$ is equal to P when ε is equal to zero.

However, this stabilization technique requires an educated guess of π. To obtain an educated guess of π, we used the heuristic procedure suggested in [5], with some modifications, to obtain initial dominating sets which were used to construct the initial restricted master problem. Then the identified dominating sets form the initial restricted master problem, and the dual variable values obtained from the initial restricted master problem were used as the initial guess $\bar{\pi}$.

In this paper, initial value of ε was set to 0.1 and the following three steps are used.

START:

Step 1: Obtain $\bar{\pi}$ from the optimal dual solution in the initial restricted master problem.

Step 2: Solve $P(\bar{\pi},\ 0.1)$ using column generation.

Step 3: Take the optimal values of $\bar{\pi}$ from Step 2 as a new guess $\bar{\pi}$, and then solve $P(\bar{\pi},\ 0.0)$.

END:

3.4 Flow Chart of an Optimal Algorithm

In the previous subsections, we discussed the serveral sub-procedures to design the optimal algorithm for the MLC problem. Now the overall procedure of the optimal algorithm can be illustrated as shown in Figure 2.

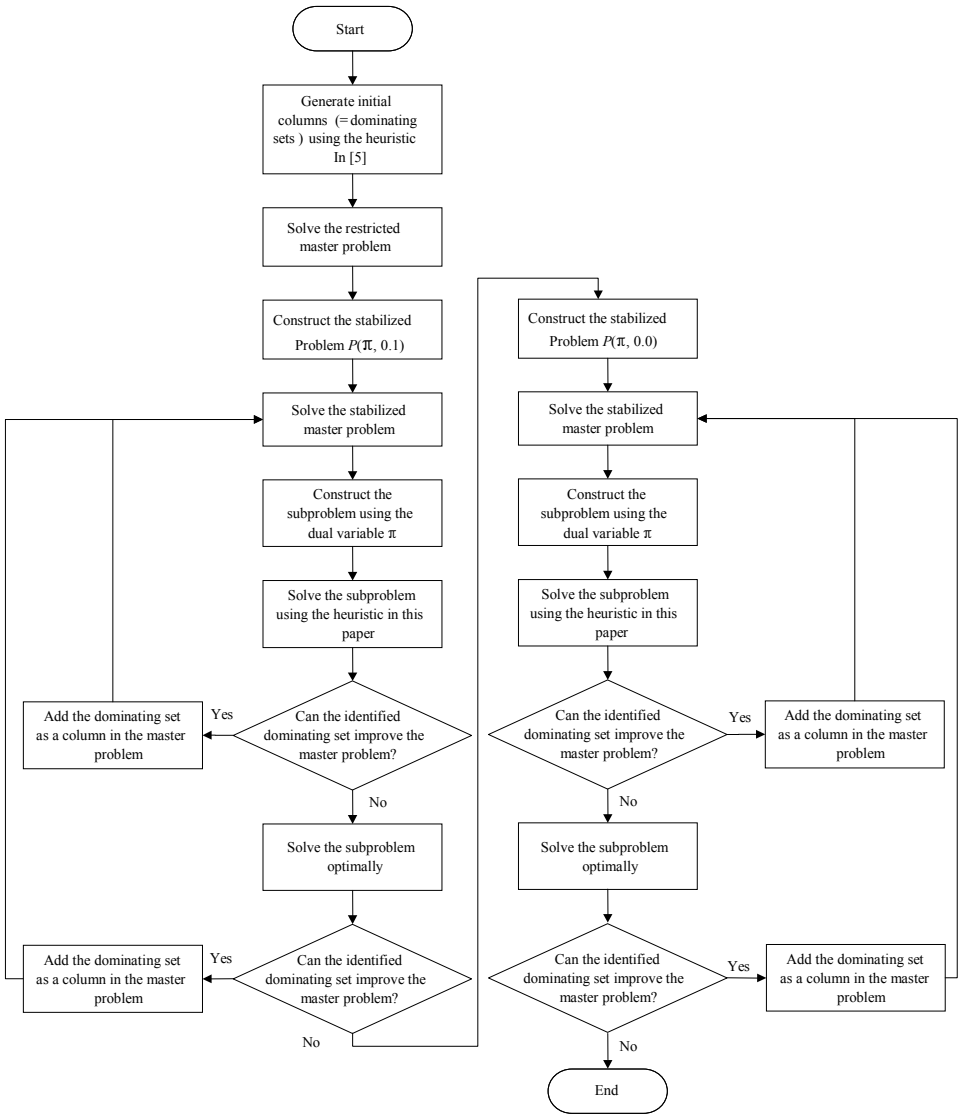

Figure 2. Flowchart of an Optimal Algorithm

4. Computational Results

In this section, we report the performances of the heuristic given in [5] and our optimal algorithm on some test problems. We implemented both algorithms and

tested them on the same problems. For each of the algorithms, the lifetimes and the algorithm run times are reported. In the case of the optimal algorithm, the numbers of generated columns to obtain the optimal solutions are also reported. In these experiments, we intended to observe the differences in the quality of the solutions obtained by the heuristic and the optimal algorithm on various problem instances, and the computation times needed to solve them.

Computational results included in this section consist of the following three subsections. In the first subsection, we tested the performances of the algorithms on randomly generated graphs. In the second subsection, we designed a 500 meters×500 meters square area and spread sensors and targets randomly within the area. In this experiment, there are three parameters: the number of sensors, the number of targets, and the sensing radius of a sensor. We performed the simulations for some combinations of the fixed two parameters and the adjustable one parameter to identify the effect of the adjustable parameter on the network lifetime. Note that, the values of the three parameters are the same as the values used in [6, 16]. In the last subsection, we assumed that the monitoring target is an area itself. To represent the area, we divided the area into smaller square fields and then regard each center point of the field as a monitoring target. Therefore, more detailed representation of the area becomes possible as the number of the center points becomes large.

In summary, our optimal algorithm outperformed the heuristic for all the problems in terms of the network lifetimes, and our algorithm found the solutions in a reasonable amount of time. The heuristic was implemented in C++, and the optimal algorithm was implemented in C++ and ILOG CPLEX 11.0 was used as an optimization software. All experiments were run on an AMD *Athlon™* 64 X2 Dual Core (2.70GHz) with 2GB Ram, and running time is given in seconds.

4.1 Tests on Random Graphs

In this subsection, we report computational results on randomly generated graphs. To generate the random graphs, we used the code obtained from http://www.brandonparker.net/graph_gen.php. We generated three types of connected graphs: sparse, medium dense graphs. According to the writer of the code, graph classification criterion is the number of edges. For a given number of vertices $|V|$, the number of edges of the sparse graph is strictly 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 strictly greater than $|V|\times|V-1|/4$.

The number of vertices ranges from 25 to 75 with an increment of 5, and the amount of power is randomly assigned between 10 to 20 to each vertex. For every value of the number of vertices, we generated 100 random graphs. Thus, only the average information is given in computational results.

Computational results are given in Table 2, Table 3 and Table 4 for each type of the graph.

Table 2. Comparison of the Two Algorithms for Sparse Graph Problems

$\|V\|$	Heuristic in [5]		Optimal algorithm		
	Run time	Lifetime	Columns	Run time	Lifetime
25	0.002	26	10	0.01	27
30	0.003	26.99	9.78	0.0075	27.34
35	0.006	25.73	11.63	0.009	26.21
40	0.007	24.41	10.08	0.008	24.61
45	0.004	24.8	11.15	0.007	24.82
50	0.005	23.76	10.91	0.009	23.88
55	0.005	24.25	13.24	0.011	24.41
60	0.005	23.42	15.06	0.012	23.71
65	0.009	23.84	14.59	0.014	24.12
70	0.009	23.59	14.56	0.016	23.71
75	0.011	22.94	14.39	0.016	23.09

Table 3. Comparison of the Two Algorithms for Medium Graph Problems

$\|V\|$	Heuristic in [5]		Optimal algorithm		
	Run time	Lifetime	Columns	Run time	Lifetime
25	0.004	109.72	53.58	0.076	128.98
30	0.006	124.89	75.59	0.18	151.04
35	0.0075	139.54	93.97	0.33	169.34
40	0.01	154.52	107.18	0.78	184.64
45	0.014	171.47	144.34	1.99	212.98
50	0.02	189.88	177.67	5.19	235.98
55	0.02	204.42	196.6	6.95	252.67
60	0.026	219.56	232.9	13.13	275.32
65	0.031	234.64	248.33	19.14	292.72
70	0.037	249.2	264.71	25.33	309.78
75	0.045	263.01	271.98	28.86	325.92

Table 4. Comparison of the Two Algorithms for Dense Graph Problems

$\|V\|$	Heuristic in [5]		Optimal algorithm		
	Run time	Lifetime	Columns	Run time	Lifetime
25	0.0064	150.31	57.06	0.065	176.2
30	0.0086	169.5	69.2	0.11	199.86
35	0.0094	199.45	87.1	0.21	235.09
40	0.012	218.03	93.38	0.27	252.43
45	0.016	243.71	101.44	0.36	278.46
50	0.02	262.66	110.2	0.48	298.08
55	0.023	278.29	121.53	0.64	316.32
60	0.03	301.28	133.55	0.83	341.87
65	0.037	322.77	139.96	1.088	363.92
70	0.044	336.46	149.76	1.34	379.06
75	0.05	358.68	156.77	1.66	402.33

In summary, compared with the heuristic shown in [5], our optimal algorithm obtained 1.15%, 22.71% and 14.66% increased lifetime in sparse, medium and dense graphs, respectively. Also, the optimal algorithm obtained the optimal solution in a reasonable amount of time.

Generally, the larger the number of edges, the smaller the size of the dominating set in a graph. Therefore, the dense graphs probably generate a large number of dominating sets, which may results in the longer network lifetimes. As expected, for a given number of vertices, the computational results show that the dense graphs generate the longest network lifetime, medium graph follows the next, and lastly, sparse graph shows the shortest lifetime. However, when we compare the run times of the optimal algorithm for a given number of vertices, medium graphs require the longest run time, dense graph follows the next, and sparse graphs indicates the fastest run time. This can be explained by the number of generated columns (= dominating sets).

Generally, it takes very small time for solving the restricted master problem and running the two heuristics. However, when the heuristic failed to obtain a profitable dominating set in the subproblem, we need to solve the subproblem optimally using the mathematical formulation. Since the medium graph required the longest run time to solve the subproblem optimally, the overall algorithm run time is longer than other

types of graphs.This probably due to the symmetry and the value of the gap between the linear programming relaxation and the integer optimal solution. Breaking symmetry issue and how strengthening the value of the linear programming relaxation can be another future direction research issue.

Cleary, our algorithm outperformed the heuristic for all the problems in terms of the network lifetimes. Although the heuristic showed good performance on sparse graphs, the differences in solution quality on medium and dense graphs are significant.

4.2 Tests on Target Monitoring Applications

In this section, we report computational results on target monitoring sensor network application. For experimental purpose, we designed a 500 meters×500 meters square area and spread sensors and targets randomly within the area. In some applications of the WSN, the monitoring targets are located in a dangerous area, thus positioning the sensors in precise locations can be a difficult task. Therefore, we spread the sensors in a random manner instead of precise positioning of the sensors. This random deployments of the sensors and the targets simulate the hostile situation.

Table 5. Comparison of the Two Algorithms on Target Coverage for 90 Sensors and 10 Targets Problems

Sensing range	Heuristic in [5]		Optimal algorithm		
	Run time	Lifetime	Columns	Run time	Lifetime
100	0.0054	61.55	23.55	0.016	62.48
120	0.0095	97.3	30.54	0.017	97.66
140	0.0087	129.91	43.56	0.033	131.02
160	0.009	154.76	43.98	0.026	156.26
180	0.012	192.51	56.04	0.051	194.74
200	0.013	246.88	71.68	0.073	251.26
220	0.015	282.49	82.69	0.11	287.32
240	0.018	340.33	89.49	0.11	344.78
260	0.018	375.84	98.85	0.12	383.13
280	0.021	441.72	85.68	0.066	448
300	0.023	481.52	130.86	0.2	495.93

Since there are three parameters, the number of sensors, the number of targets, and the sensing radius of a sensor, we designed the experiments as the following. First, we fixed the two parameters, the number of sensors and the number of targets, as 90 and 10, respectively, and then changed the sensing range between 100 and 300 meters with an increment of 20 meters. Second, we fixed the number of sensors and the sensing range as 90 and 250 meters, respectively, then changed the number of targets between 10 and 50 with an increment of 5. Lastly, we fixed the number of targets and the sensing range as 10 and 250 meters, respectively. Then changed the number of sensors between 90 and 140 with an increment of 5. The amount of battery capacity is randomly assigned between 10 to 20 to each sensor, and we generated 100 random problems for every value of the fixed two parameters. Therefore, only the average information is reported in computational results.

Computational results are given in Table 5, Table 6 and Table 7.

Table 6. Comparison of the Two Algorithms on Target Coverage for 90 Sensors with Sensing Range of 250 Meters Problems

Targets	Heuristic in [5]		Optimal algorithm		
	Run time	Lifetime	Columns	Run time	Lifetime
10	0.018	362.09	94.81	0.125	368.07
15	0.021	340.05	110.51	0.19	348.58
20	0.023	307.97	109.32	0.19	314.17
25	0.027	300.11	122.38	0.23	308.66
30	0.031	312.1	144.4	0.34	322.72
35	0.032	282.71	110.85	0.24	289.25
40	0.034	275.62	132.28	0.33	283.27
45	0.039	279.12	148.81	0.4	289.33
50	0.041	273.71	126.9	0.31	280.8

When we compare the optimal algorithm run times with the run times on graph problems, our algorithm requires very small amount of run time to obtain the maximum network lifetimes. Also, the differences in solution quality between the heuristic and our algorithm are relatively small. Thus, the maximum network lifetime coverage problems for the target monitoring may be easier than the graphs.

Table 7. Comparison of the Two Algorithms on Target Coverage for 10 Targets with Sensing Range of 250 Meters Problems

Sensors	Heuristic in [5]		Optimal algorithm		
	Run time	Lifetime	Columns	Run time	Lifetime
90	0.02	365.24	100.38	0.12	371.5
95	0.021	372.5	377.26	0.15	377.26
100	0.022	419.27	108.58	0.17	425.4
105	0.023	416.1	112.96	0.2	422.35
110	0.025	421.1	118.91	0.25	426.07
115	0.026	462.13	116.31	0.22	467.94
120	0.030	487.23	144.3	0.34	493.25
125	0.032	494.84	152.25	0.45	499.49
130	0.033	500.88	121.93	0.2	507.61
135	0.037	575.61	171.56	0.55	585.21
140	0.036	531.55	174.52	0.67	540.46

4.3 Tests on Area Monitoring Applications

In this section, we report the computational results on area monitoring sensor network applications. We assumed that the monitoring area is a 500 meters×500 meters square area. To discretize the monitoring area, we first divided the area into smaller square fields and then considered each center point of the field as the monitoring target. When we discretize the monitoring area, if the number of fields is too small, the monitoring area will not be well defined. On the other hand, the size of the corresponding formulation will be small. On the contrary, if the number of fields is too large, the monitoring area will be well defined, but the size of the corresponding mathematical formulation will be huge. In this experiment, we set the size of the field as a 10 meters×10 meters square area. Therefore 2,500 fields are generated to represent the monitoring area. We assumed that the sensing range is the same for all the sensors as 250 meters and the amount of power is randomly assigned between 10 to 20 for each sensor. Then we changed the number of sensors between 90 and 140 with an increment of 5. We generated 100 random problems for every value of the number of sensors, thus only the average information will be included in the computational results.

Computational results are given in Table 8.

Table 8. Comparison of the Two Algorithms on Area Monitoring for Sensing Range is Fixed as 250 Meters Problems

Sensors	Heuristic in [5]		Optimal algorithm		
	Run time	Lifetime	Columns	Run time	Lifetime
90	0.59	207.64	132.24	2.85	214.04
95	0.69	218.88	141.07	2.31	225.5
100	0.81	226.7	173.37	5.08	234.6
105	0.92	247.3	182.09	5.46	255.01
110	1.068	259.55	234.68	8.48	269.81
115	1.19	266.93	238.08	9.35	276.87
120	1.34	288.56	250.3	9.92	298.2
125	1.51	298.2	268.59	11.67	308.07
130	1.69	311.47	330.59	16.80	324.2
135	1.84	320.18	364.96	20.21	333.35
140	2.1	339.31	360.5	20.38	352.24

The results show that, compared with the algorithm run times of the target monitoring applications, our algorithm required more run times to obtain the solutions. Since the number of targets (2,500) used in the area monitoring application is much greater than the number of targets (10~50) used in the target monitoring applications, the subproblems ask for a longer run time. Nevertheless, our algorithm obtained solutions in a reasonable amount of time for all the problems.

5. Conclusions

One of the main features of the wireless sensor network is the scarcity of the battery resource. Since each sensor is powered by a limited amount of battery capacity, the lifetime of the network is limited. To prolong the lifetime of the network, we can schedule the sensors' activities so that they alternate between the active state and the low-battery sleep state. One possible solution for this scheduling scheme is to partition the sensors into several dominating sets, so that the sensors in each of the dominating sets can individually perform the monitoring task. After the dominating sets are identified, each dominating set is activated successively. Therefore, at a

specific time, only the sensors in one dominating set are required to be active. This mechanism can prolong the network lifetime, and the performance of the mechanism depends on the identification of the dominating sets and the efficient allocation of the resources for the dominating sets.

In this research, we developed a column generation algorithm which can be used to find an optimal solution of the network lifetime. To accelerate the convergence speed to an optimal solution, two techniques are applied. One is the stabilized column generation for smooth convergence to the optimal values of the dual variables, the other one is the utilization of the two heuristics. We performed extensive computational experiments, and our algorithm outperformed the heuristic suggested earlier in terms of the network lifetimes, and it found the optimal solutions for all the problems in a reasonable amount of time.

Although the MLC problem probably made several assumptions, the MLC model can be extended to incorporate other requirements such as bandwidth limit and connectivity, as shown in [7, 29]. Therefore, we expect that our mathematical model and the optimal approach for the MLC problem probably be used to address the upcoming new models and the problems.

References

[1] Agarwal, Y., K. Mathur, and H. M. Salkin, "A set-partitioning-based exact algorithm for the vehicle routing problem," *Networks* 19 (1989), 731-749.

[2] Berman, P., G. Calinescu, C. Shah, and A. Zelikovsky, "Power efficient monitoring management in sensor networks," *Proceedings of IEEE Wireless communication and networking conference*, Atlanta, GA, (2004), 2329-2334.

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

[4] Calinescu, G., "A fast localized algorithm for scheduling sensors," *Journal of Parallel and Distributed Computing* 68 (2006), 507-514.

[5] Cardei, M., M. T. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," *IEEE INFOCOM* 24th *Annual Joint Conference of the IEEE Computer and Communications Societies*, IEEE Communications Society, Miami, Florida, 3 (2005), 1976-1984.

[6] Cardei, M. and D. Z. Du, "Wireless Sensor Network Lifetime through Power Aware Organization," *Wireless Networks* 11, 3 (2005), 333-340.

[7] Cheng, M. X., L. Ruan, and W. Wu, "Coverage Breach Problems in Bandwidth-Constrained Sensor Networks," *ACM Transactions on Sensor Networks* 3, 12 (2007).

[8] Cheng, M. X. and X. Gong, "Maximum Life Time Coverage Preserving Scheduling Algorithms in Sensor Networks," *Journal of Global Optimization*, 2010.

[9] Dantzig, G. B. and P. Wolfe, Decomposition principle for linear programs, *Operations Research* 8, 1 (1960), 101-111.

[10] Floreen, P. P. Kaski, T. Musto, and J. Suomela, "Local Approximation Algorithms for Scheduling Problems in Sensor Networks," *Lecture Notes in Computer Science* 4837 (2008), 99-113.

[11] Ford, L. R. and D. R. Fulkerson, "A suggested computation for maximal multi-commodigy network flows," *Management Science* 5, 1 (1958), 97-101.

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

[13] Gilmore, P. C. and R. E. Gomory, "A linear programing approach to the cutting-stock problem," *Operations Research* 9, 6 (1961), 849-859.

[14] Gilmore, P. C. and R. E. Gomory, "A linear programing approach to the cutting-stock problem-Part II," *Operations Research* 11, 6 (1963), 863-888.

[15] Hahn, R. and H. Reichl, "Batteries and power supplies for wearable and ubiquitous computing," 3rd *International Symposium on Wearable Computers*, IEEE Computer Society, San Francisco, California, (July/August 1999), 168-169.

[16] Lai, C., C. Ting, and R. Ko, An effective genetic algorithm to improve wireless sensor network lifetime for large-scale surveillance applications, *IEEE Congress on Evolutionary Computation*, Singapore, (2007), 3531-3538.

[17] Johnson, D. S., M. Yannakakis, and C. H. Papadimitriou, On Generating all Maximal Independent Sets, *Information Processing Letters* 27 (1988), 119-123.

[18] Lübbecke, M. E. and J. Desrosiers, "Selected topics in column generation," *Operations Research* 53, 6 (2005), 1007-1023.

[19] du Merle, O., D. Villeneuve, J. Desrosiers, and P. Hansen, "Stabilized column generation," *Discrete Mathematics* 194 (1999), 229-237.

[20] Marsten, R. E., "The use of the box step method in discrete optimization," *Mathematical Programming Study* 3 (1975), 127-144.

[21] Marsten, R. E., W. W. Hogan, and J. W. Blankenship, "The BOXSTEP method for large-scale optimization," *Operations Research* 23, 3 (1975), 389-405.

[22] Mito, M. and S. Fujita, "Maximum Connected Domatic Partition of Directed Path Graphs with Single Junction," *Lecture Notes in Computer Science* 5092 (2008), 425-433.

[23] Moscibroda, T. and R. Wattenhofer, "Maximize the Lifetime of Dominating Sets," *IEEE International Parallel and Distributed Processing Symposium*, Denver, Colorado, April 2005.

[24] Pemmaraju, S. V. and I. A. Pirwani, "Energy conservation via domatic partitions," 7th *ACM international symposium on Mobile ad hoc networking and computing*, ACM SIGMOBILE, Florence, Italy, (2006), 143-154.

[25] Pigatti, A., M. P. de Aragao, and E. Uchoa, "Stabilized branch and cut and price for the generalized assignment problem," *Electronic Notes in Discrete Mathematics* 19 (2005), 389-395.

[26] Slijepcevic, S. and M. Potkonjak, Power efficient organization of wireless sensor networks, *IEEE International Conference on Communications*, Helsinki, Finland, 2 (2001), 472-476.

[27] Stine, J. and G. de Veciana, Improving energy efficiency of centrally controlled wireless data networks, *Wireless Networks* 8, 6 (2002), 681-700.

[28] Wang, B., Coverage Lifetime Maximization in *Coverage Control in Sensor Networks*, Springer, London, 2010.

[29] Zorbas, D. and C. Douligeris, Satisfying coverage and connectivity in bandwidth constrained sensor networks, 9th *International Conference on Communications and information technologies*, Incheon, Korea, 2C (2009), 390-395.