An Optimal Algorithm for the Sensor Location Problem to Cover Sensor Networks

Kim, Hee seon*, Sungsoo Park**
* Department of Industrial Engineering, KAIST (kimhs3004@kaist.ac.kr)
** Department of Industrial Engineering, KAIST (sspark@kaist.ac.kr)

Abstract

We consider the sensor location problem (SLP) on a given sensor field. We present the sensor field as grid of points. There are several types of sensors which have different detection ranges and costs. If a sensor is placed in some point, the points inside of its detection range can be covered. The coverage ratio decreases with distance.

The problem we consider in this thesis is called multiple-type differential coverage sensor location problem (MDSLP). MDSLP is more realistic than SLP. The coverage quantities of points are different with their distance form sensor location in MDSLP. The objective of MDSLP is to minimize total sensor costs while covering every sensor field. This problem is known as NP-hard.

We propose a new integer programming formulation of the problem. In comparison with the previous models, the new model has a smaller number of constraints and variables. This problem has symmetric structure in its solutions. This group is used for pruning in the branch-and-bound tree. We solved this problem by branch-and-cut(B&C) approach. We tested our algorithm on about 60 instances with varying sizes.

1. Introduction

The research issues in distributed networks are effective surveillance and environment monitoring. The monitoring and surveillance can be done with sensors, and the covered networks are called sensor networks. The covered areas are simply called sensor fields. If the sensor fields are predetermined, we can make a decision to choose the method that locates sensors to meet a certain quality of service requirement, such as surveillance [6], [7], target location [4], [5], [10], and target tracking. If the location of sensors is decided randomly, the deterioration of service quality is sure to result. This is the reason why we have to research the method of deployment of sensors.

The sensor fields are covered by several different types of sensors, which are appropriately located in sensor networks. These sensors have different detection range and cost. The sensor can monitor the region which is the inside of its detection range. The cost of sensors increases with their detection range. Clearly, sensors which have longer detection range have higher cost. If we use only long-range sensors, the sum of total sensor costs may be too much expensive. On the contrary, if we use only short-range sensors, the effectiveness of the sensor network may not be achieved. Therefore, the efficient sensor location strategy is necessary to minimize total cost.

The sensor networks can be deployed in two ways. The first one is a random placement. When the environment of sensor fields is unknown, this is the only choice. It is impossible to decide the optimal sensor locations, because we have no information about the sensor fields. Sensors are thrown to any place by aircrafts randomly. The second one is a grid-based placement. If the sensor fields are predetermined, we can use this way. The sensor fields are divided into grids. Sensors can be deployed at any grid points. If the number of grid points is infinitely large, we can regard the sensor fields as continuous fields. It is more realistic. This thesis focuses on this method.

The problem in the sensor networks can be applied to many practical problems ([5], [6], [20]). Nevertheless, prior researches have ignored the sensor location issues. Most previous study has focused on efficient sensor communication ([8], [15]) and sensor fusion ([3], [14]) for a given sensor field. As the size of sensor networks is expanded and the number of sensors is increased, efficient deployment strategies is required.

The sensor location problem (SLP) determines the number of sensors and the location of sensors to be selected to satisfy the service quality in a given area. K. Chakrabarty et al.[4] formulates SLP with grid-based placement in terms of cost minimization under coverage constraints. Frank Y.S. Lin et al.[10] presents a heuristic algorithm to SLP for target location. We consider the more realistic problem, which has differential coverage quantity. We call this problem as the multiple-type differential coverage sensor location problem (MDSLP). In SLP, the coverage quantities of points inside of detect range are one. In contrast, the coverage quantities of points are different with their distances from sensor location in MDSLP. In this thesis, we focus on MDSLP with grid-based placement.

The thesis is organized as follows. Chapter 2 defines the problem. In chapter 3, we present mathematical formulations for two models of MDSLP. Chapter 4 provides an algorithm for the problem. In chapter 5, we present computational results for the algorithm. Finally, we give concluding remarks in Chapter 6.

2. Problem Description

This chapter contains a detailed description of MDSLP (multiple-type differential coverage sensor location problem): input and output of MDSLP, assumptions, and objective function will be described.

Inputs are as follows.

- The size of sensor networks
- The minimal sum of coverage quantity by sensors
- The number of types of sensors

The sensor field is defined as a square. The size of sensor networks is the width (or length) of the sensor field. We represent the sensor field as a grid of points. So, the total number of grid points in the sensor field is the square of the size of sensor networks.

Coverage can be considered as the measure of quality of service of a sensor network. Every grid point must be covered at least the minimal sum of coverage quantity by sensors. The available sensor types which can be appropriately placed in the sensor field are restricted. These sensors differ form each other in their detection ranges and costs. Figure 2-1 shows the types of sensors.

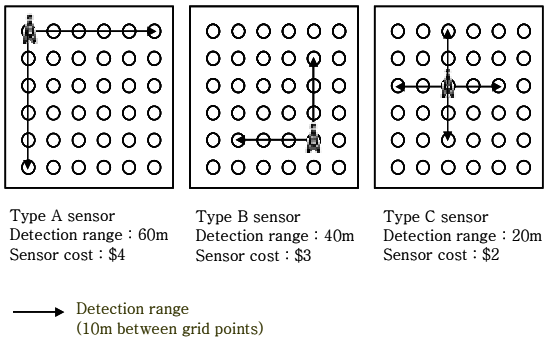

Figure 2-1. The types of sensors

Output is the assignment of sensors to grid points. When we solve MDSLP, we assume that the sensor networks consist of two-dimensional square sensor field which is presented as a grid of points. The size of sensor networks means the number of grid points in width (length) of the sensor field. The size of sensor networks in figure 2-1 is 6. It means that the width and length of the sensor field is 60m, and the sensor networks have an area of 3600 square meters. If a sensor is placed in some grid point, the points inside of its detection range can be covered. The coverage ratio decreases with distance between sensor and the target point. The sensor cost increases with its detection range.

Our objective is to assign sensors for complete coverage of the sensor field with minimum total cost of sensors.

3. Mathematical Formulation

In this chapter, we introduce two formulations of the MDSLP. The first one is modified from the model proposed by [4] and [10]. The second one is the formulation we propose.

3.1. Formulation 1

The original formulation of the MDSLP was proposed by [4]. [10] presented another formulation with different objective function. Formulation 1 is modified from above two models. The following notation and decision variables are used in the formulation:

[Notation]

N_H the set of grid points in width of the sensor field, $N_H = \{0,1,...,u-1\}$

N_V the set of grid points in length of the sensor field, $N_V = \{0,1,...,u-1\}$

T the set of sensor types, $T = \{1,...,t\}$

u the number of grid points in width (length) of the sensor field

t the number of sensor types

α the minimal sum of coverage quantity by sensors

r^s the range of type-s sensor, $s \in T$

c^s the cost of type-s sensor

$d_{(i,j)(k,l)}$ distance between grid points (i,j) and (k,l), $i,k \in N_H,\ j,l \in N_V$,

$$d_{(i,j)(k,l)} = \left\lceil \sqrt{(i-k)^2 + (j-l)^2} \right\rceil - 0.5$$

$\gamma^s_{(i,j)(k,l)}$ coverage ratio of type-s sensor at grid point (k,l) in (i,j),

$$\gamma^s_{(i,j)(k,l)} = \frac{r^s - (d_{(i,j)(k,l)} + 0.5)}{r^s}$$

[decision variables]

$$x^s_{(i,j)(k,l)} = \begin{cases} 1, & \text{if the grid point}(i,j) \text{ can be covered by type-}s \text{ sensor at grid point}(k,l) \\ 0, & \text{otherwise} \end{cases}$$

$$x^s_{(k,l)(k,l)} = \begin{cases} 1, & \text{if type-}s \text{ sensor is assigned at grid point } (k,l) \\ 0, & \text{otherwise} \end{cases}$$

Formulation can be constructed as follows.

[Formulation 1]

$$\text{Min} \quad \sum_{s\in T}\sum_{k\in N_H}\sum_{l\in N_V} c^s x^s_{(k,l)(k,l)} \tag{1}$$

$$\text{s.t.} \quad d_{(i,j)(k,l)} x^s_{(i,j)(k,l)} - x^s_{(k,l)(k,l)} r^s \le 0 \tag{2}$$
$$\forall i,k \in N_H,\ \forall j,l \in N_V,\ (i,j) \ne (k,l),\ \forall s \in T$$

$$x^s_{(k,l)(k,l)} - x^s_{(i,j)(k,l)} \le \frac{d_{(i,j)(k,l)}}{r^s} \tag{3}$$
$$\forall i,k \in N_H,\ \forall j,l \in N_V,\ (i,j) \ne (k,l), \forall s \in T$$

$$\sum_{s\in T}\sum_{k\in N_H}\sum_{l\in N_V} \gamma^s_{(i,j)(k,l)} x^s_{(i,j)(k,l)} \ge \alpha \tag{4}$$
$$\forall i \in N_H,\ \forall j \in N_V$$

$$\sum_{s\in T} x^s_{(k,l)(k,l)} \le 1 \tag{5}$$
$$\forall k \in N_H,\ \forall l \in N_V$$

$$x^s_{(i,j)(k,l)} \in \{0,1\} \tag{6}$$
$$\forall i,k \in N_H,\ \forall j,l \in N_V,\ \forall s \in T$$

The objective (1) is to minimize the sum of sensor costs with complete coverage of the sensor field.

Constraints (2) and (3) mean that sensor with range r^s placed on grid point (k,l) can detect a target at grid point (i,j) if the distance between these two grid points is less than r^s. Constraints (4) mean that every grid point must be covered at least α. Constraints (5) represent that at most one sensor can be placed on a grid point. Constraints (6) are the binary conditions on the variables.

3.2. Formulation 2

In this section, we propose another integer programming formulation of MDSLP. Formulation 1 has a large number of constraints and variables. To reduce the size of the model, we make new formulation. It has a smaller number of constraints and variables than the previous one. We remove constraints (2) and (3) in formulation 1. We use the new coefficient

$\delta^s_{(i,j)(k,l)}$ instead of $\gamma^s_{(i,j)(k,l)}$. It contains the meaning of constraints (2) and (3). The decision variables in formulation 2 are only one kind of variable set that means the assignment of sensor at grid point.

Notation and decision variables for new formulation are as follows:

[Notation]

N_H the set of grid points in width of the sensor field, $N_H = \{0,1,...,u-1\}$

N_V the set of grid points in length of the sensor field, $N_V = \{0,1,...,u-1\}$

T the set of sensor types, $T = \{1,...,t\}$

u the number of grid points in width (length) of the sensor field

t the number of sensor types

α the minimal sum of coverage quantity by sensors

r^s the range of type-s sensor, $s \in T$

c^s the cost of type-s sensor

$d_{(i,j)(k,l)}$ distance between grid points (i,j) and (k,l), $i,k \in N_H,\ j,l \in N_V$, $d_{(i,j)(k,l)} = \sqrt{(i-k)^2 + (j-l)^2}$

$\delta^s_{(i,j)(k,l)}$ coverage ratio of type-s sensor at grid point (k,l) in (i,j),

$$= \begin{cases} \dfrac{r^s - d_{(i,j)(k,l)}}{r^s}, & \text{if } r^s - d_{(i,j)(k,l)} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

[decision variables]

$$x^s_{(i,j)} = \begin{cases} 1, & \text{if type-}s\text{ sensor is assigned at grid point } (i,j) \\ 0, & \text{otherwise} \end{cases}$$

With above notation and decision variables, new formulation can be constructed as follows.

[Formulation 2 (MDSLP)]

$$\text{Min} \quad \sum_{s\in T}\sum_{i\in N_H}\sum_{j\in N_V} c^s x^s_{(i,j)} \tag{7}$$

s.t.

$$\sum_{s\in T}\sum_{i\in N_H}\sum_{j\in N_V} \delta^s_{(i,j)(k,l)} x^s_{(i,j)} \geq \alpha \quad \forall k \in N_H,\ \forall l \in N_V \tag{8}$$

$$\sum_{s\in T} x^s_{(i,j)} \leq 1 \quad \forall i \in N_H,\ \forall j \in N_V \tag{9}$$

$$x^s_{(i,j)} \in \{0,1\} \quad \forall i \in N_H,\ \forall j \in N_V,\ \forall s \in T \tag{10}$$

The objective function (7) means the sum of sensor costs for complete coverage of the sensor field is minimized. By constraints (8), every grid points can be covered by sensor at least minimal sum of coverage quantity by sensors α. Coefficient $\delta^s_{(i,j)(k,l)}$ describes that a sensor always detects a target that lies within its range. That is, sensor with range r^s placed on grid point (k,l) can detect a target at grid point (i,j) if the distance between these two grid points is less than or equal to r^s. Constraints (9) mean that at most one sensor should be placed on a grid point. Constraints (10) are the binary conditions on the variables.

The sensor location problem (SLP) is NP-hard ([4]). This problem is restricted to the minimum-cost satisfiability problem, which is known as NP-hard ([11],[23]). SLP is a special case of MDSLP. Therefore, MDSLP is NP-hard. (Frank Y.S. Lin et al.[05ieee]].

4. Algorithm

4.1. Overview

The basic approach of algorithm of this thesis is branch-and-cut. Branch-and-cut is a branch-and-bound algorithm in which cutting planes are generated throughout the branch-and-bound tree. This algorithm has been widely used for large-scale mixed integer problem [1], [2], [9], [13], [21].

At first, we get an upper bound of MDSLP by heuristic algorithm. Then, we construct the initial formulation of LP_0. LP_0 is the LP relaxation of the initial IP formulation. After solving the LP_0, we continue to find the valid inequalities (Gomory fractional cuts, cover inequalities or generalized upper bound (GUB) cover inequalities) which are violated by the current solution. If one violated inequality of these three kinds of inequalities is found, the valid inequality found in its separation algorithm formulation is added as cut to LP_0 and no other separation algorithm is called. So we construct the next formulation, which is LP_1.

If we get LP_1, we go through the same procedure as we do after the LP_0 is obtained.

After iterating this procedure, when no more valid inequalities can be found, we check if the solution obtained by solving the last LP, which is LP_k, is integral. If we have obtained an integral solution, we are done with an optimal solution of MDSLP. Otherwise, we have to start the branch-and-cut procedure to find an optimal solution to the final formulation, LP_k, which is also an optimal solution of MDSLP.

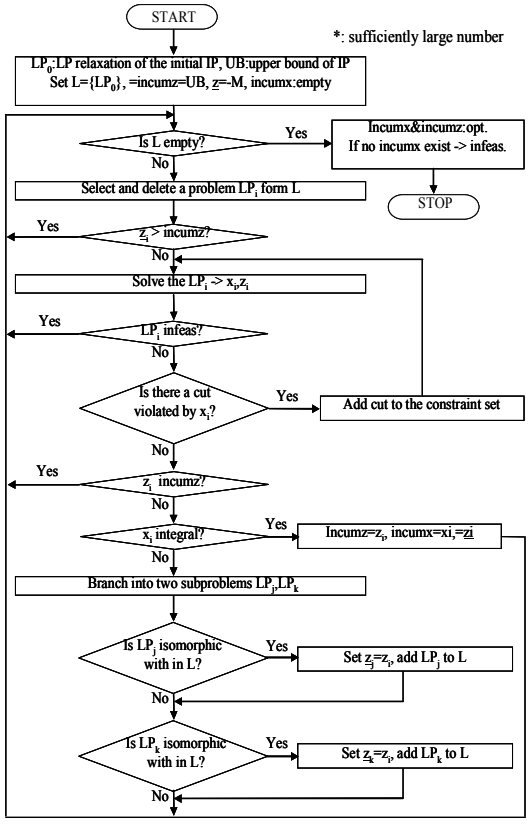

Figure 4-1. The overall procedure of the branch-and-cut algorithm

MDSLP has a large symmetry group which has 8 elements. These elements return the same solutions, so we consider one of them when we execute the branch-and-cut procedure.

The overall procedure of the algorithm to solve MDSLP is presented in Figure 4-1.

4.2. Valid inequalities

A *valid inequality* for an integer programming is an inequality that is satisfied by all feasible solutions. We are interested in valid inequalities, which are called *cuts* that are not included in the current formulation and are not satisfied by all feasible points to the LP relaxation. A *violated cut* is a cut that is not satisfied by the given optimal solution to the LP relaxation. If we find a violated cut, we can add it to the LP relaxation. Then we can tighten current formulation. The LP feasible region becomes smaller but the IP feasible region does not changed. Then we can resolve the LP and repeat the above procedure, if necessary, so long as we can continue to find violated cuts.

4.2.1. Gomory fractional cuts

Gomory fractional cuts are generated by applying integer rounding on a pivot row in the optimal LP tableau for a basic integer variable with a fractional solution value ([19]).

Now, we denote the constraints $S = \{x \in Z_+^n \mid Ax \le b\}$ in equality form as $S^l = \{x \in Z_+^{n+m} \mid (A,I)x = b\}$. The original variables are $(x_1, x_2, ..., x_n)$ and the slack variables are $(x_{n+1}, x_{n+2}, ..., x_{n+m})$. A and b are an integral matrices. Suppose $N = \{1,..,n\}$, $M = \{1,..,m\}$, $A = \{a_1,..,a_n\}$, and $I = \{e_1,..,e_m\}$. We consider the linear combination of equations given by $\lambda(A,I)x = \lambda b$, where $\lambda = (\lambda_1, \lambda_2, ..., \lambda_m)$ is a weight vector. We define $\overline{a_j} = \lambda a_j$ for $j \in N$ and $\overline{b} = \lambda b$. Then $\lambda(A,I)x = \lambda b$ can be written as $\sum_{j \in N} \overline{a_j} x_j + \sum_{i \in M} \lambda_i x_{n+i} = \overline{b}$.

Proposition 1. The inequality

$$\sum_{j \in N} f_j x_j + \sum_{i \in M} g_i x_{n+i} \ge f_0 \qquad (11)$$

is valid, where $f_j = \overline{a_j} - \lfloor \overline{a_j} \rfloor$ for $j \in N$, $g_i = \lambda_i - \lfloor \overline{\lambda_i} \rfloor$ for $i \in M$, and $f_0 = \overline{b} - \lfloor \overline{b} \rfloor$.

4.2.2. Cover cuts

If a constraint takes the form of a knapsack constraint, then there is a minimal cover associated with the constraint. A minimal cover is a subset of the variables of the inequality such that if all the subset variables were set to one, the knapsack constraint would be violated, but if any one subset variable were excluded, the constraint would be satisfied.

IP often has a row i of the form $\sum_{j \in N} a_{ij} x_j \le b_i$. We assume without loss of generality that $a_{ij} \ge 0$ for all $j \in N$. If not, we can complement the variables for which $a_{ij} < 0$. When we consider only row i, IP can be relaxed to a *0-1 knapsack problem,* with the feasible region $P^{cover} = \{x \in \{0,1\}^N \mid \sum_{j \in N} a_j x_j \le b\}$. A set $C \subset N$ is called a *cover* if $\sum_{j \in C} a_j > b$.

Proposition 2. If $C \subset N$ is a cover,

$$\sum_{j \in C} x_j \le |C| - 1 \qquad (12)$$

is a valid inequality.

In general, these inequalities are not facet-defining for P^{cover}, but they can be strengthened by a procedure called *lifting* ([12], [16], [19]).

4.2.3. GUB cover cuts

A GUB constraint for a set of binary variables is a sum of variables less than or equal to one. If the variables in a GUB constraint are included in a knapsack constraint, the minimal cover can be selected with the additional consideration that at most one of the members of the GUB constraint can be one in a solution.

A GUB inequality is an inequality of the form $\sum_{j \in Q} x_j \le 1$, where $Q \subset N$. When IP contains a knapsack row i and a set of GUB inequalities defined by disjoint sets $Q_k \subset N$ for $k \in K$, we obtain a relaxation of IP with the feasible region $P^{GUB} = \{x \in \{0,1\}^N \mid \sum_{j \in N} a_{ij} x_j \le b_i, \sum_{j \in Q_k} x_j \le 1 \ \forall k \in K\}$. A *GUB cover* C_G is a cover that obeys the GUB constraints, that is, no two elements of the cover belong to the same Q_i.

Proposition 3. For any *GUB cover* C_G, the inequality

$$\sum_{j \in C_G} x_j \le |C_G| - 1 \qquad (13)$$

is valid for P^{GUB}.

The lifting procedure of the GUB cover also leads to significant strengthening of this inequality ([12], [19]).

4.3. Isomorphism pruning

When we solve huge size of IP in a branch-and-cut framework, the number of branch nodes is large. However, these problems often have a large number of symmetries, and if we are not interested in equivalent solutions, we can simply ignore symmetric problems. This has the possibility of dramatically reducing the number of nodes that we need to consider in a branch-and-bound tree ([17], [18], [22]).

MDSLP has a large symmetry group. We consider the sensor field as a square. We have 8 isomorphic problems as the (aspect) direction of sight: the identity I, three rotations R_{90}, R_{180}, R_{270}, the vertical symmetry V, the horizontal symmetry H, the symmetry along the main diagonal D, and the symmetry along the other diagonal D'. If we consider in an isomorphism-free branch-and-cut tree, we can reduce the effect of solving the problem by 1/8. Figure 4-2 shows the symmetry group with size 6.

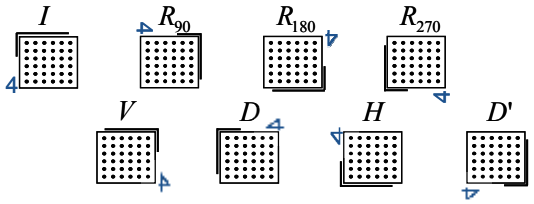

Figure 4-2. The symmetry group with size 6

We use permutations to present the symmetry group. Here, we give some definitions of the permutation. Let Π^n be the set of all permutations over the ground set $I^n = \{0,1,...,n-1\}$. A permutation $\pi \in \Pi^n$ is represented by an n-vector. $\pi[i]$ is the image of i under π. If v is an n-vector, $w = \pi(v)$ is the n-vector obtained by permuting the coordinates of v according to π. Let's consider the IP problem whose coefficient of the objective is c^T, coefficient of the constraints is A, and RHS of the constraints is b. A is an $m \times n$ matrix. For a permutation π of the n variables such that $\pi(c) = c$ and a permutation μ of the m constraints of A such that $\mu(b) = b$, $A(\pi,\mu)$ is the matrix obtained form A by permuting its columns according to π and its rows according to μ.

Let $G = \{\pi \mid \text{there exists } \mu \text{ s.t. } A(\pi,\mu) = A\}$, which is a permutation group of I^n. For an n-vector x and any permutation $\pi \in G$, we note that :

x feasible $\Leftrightarrow \pi(x)$ feasible
x optimal $\Leftrightarrow \pi(x)$ optimal

Hence, we call G the *symmetry group of the feasible (and optimal) set of the IP*.

Example 1. Consider the group G of symmetries of the 3×3 square.

⓪	①	②
③	④	⑤
⑥	⑦	⑧

G comprises 8 permutations :

$G \in \{I, R_{90}, R_{180}, R_{270}, V, H, D, D'\}$,

$I = \{0,1,2,3,4,5,6,7,8\}^T$, $R_{90} = \{2,5,8,1,4,7,0,3,6\}^T$,

$R_{180} = \{8,7,6,5,4,3,2,1,0\}^T$, $R_{270} = \{6,3,0,7,4,1,8,5,2\}^T$,

$V = \{2,1,0,5,4,3,8,7,6\}^T$, $H = \{6,7,8,3,4,5,0,1,2\}^T$,

$D = \{0,3,6,1,4,7,2,5,8\}^T$, $D' = \{8,5,2,7,4,1,6,3,0\}^T$.

When we execute the branch-and-cut procedure, we have to treat the symmetry group with arrays of the indices of the fixed variables. If we have t types of sensor, we should store $t+1$ arrays for each branch node. Given a node a of our branch-and-cut tree over $n \cdot t$ variables, we define the following $t+1$ sets :

$F_a^0 = \{j \cdot k \in Z_{n \cdot t} \mid x_j^k \text{ is fixed to 0 at } a\}$

$F_a^1 = \{j \cdot 1 \in Z_{n \cdot t} \mid x_j^1 \text{ is fixed to 1 at } a\}$

$F_a^2 = \{j \cdot 2 \in Z_{n \cdot t} \mid x_j^2 \text{ is fixed to 1 at } a\}$

M

$F_a^t = \{j \cdot t \in Z_{n \cdot t} \mid x_j^t \text{ is fixed to 1 at } a\}$

Every branch node has these $t+1$ sets. The set for descendant is inherited form the ancestor's one, and adds one index to an array. When we make two branch nodes, we should check these new nodes are isomorphic to the previously defined nodes. If the isomorphic node is found, we can prune the node. It gives the same solution value as the previous node. Searching the isomorphic node does not give the improvement of the solution. It is waste of time. So, we consider one problem for each isomorphism class, which is called the *representative* of the class.

4.4. Getting an upper bound

Before the branch-and-cut procedure, we run a heuristic algorithm that describes an upper bound to the branch-and-bound tree. In branch-and-bound, an upper bound is obtained by discovering feasible solutions to the original MDSLP. When we search an upper bound, there exists a trade-off between the quality and the search time. We would like to get a solution as close to optimal as possible, but it takes quite a time to search. We can get a loose solution in a short time. In our heuristic algorithm, we use a short-time algorithm. We propose a pattern of the sensor placement. We assume the minimal sum of coverage quantity by sensors α is 1. At first, choose an arbitrary point and locate sensor at this point. With this point as the central figure, locate sensors at the points on the upper, lower, left, and right sides of the point. Intervals between arbitrary point and new points are the range of sensor. Then, choose the other point which is located on the diagonal of the first chosen point, and locate sensor. The vertical and horizontal distances between two points are half of the range of sensor. With this new point, locate sensors with the previous pattern. Figure 4-3 shows the pattern that we suggest.

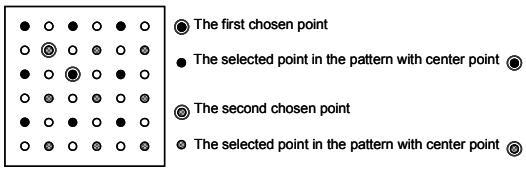

Figure 4-3. An example of getting an upper bound with the range of sensor is 2.

Here is the procedure for getting an upper bound solution value to the branch-and-bound ([24]).

Given $u(\geq 2)$, t, r^s, and c^s for $s \in T$

Step 1. Sort the efficiency of sensor type. $\frac{r^t}{c^t} \geq \frac{r^{t-1}}{c^{t-1}} \geq \Lambda \geq \frac{r^1}{c^1}$.

Step 2. If $u-1 \leq \left\lfloor \frac{r^2-1}{2} \right\rfloor$, $h=1$. Else if $u-1 \leq \left\lfloor \frac{r^3-1}{2} \right\rfloor$, $h=2$. Λ. Else if $u-1 \leq \left\lfloor \frac{r^t-1}{2} \right\rfloor$, $h=t-1$. Else, $h=t$.

Step 3. Set $a = \left\lfloor \frac{u}{2} \right\rfloor$, and locate type-$h$ sensor at (a,a) grid point. Set $b = a - \frac{r^h}{2}$ (or $a + \frac{r^h}{2}$), and locate type-h sensor at (b,b) grid point. According to the pattern, locate type-h sensors.

Step 4. Check the feasibility for all grid points. If we find a point that is not covered yet, we should locate sensor at this point.
Step 5. If every grid points is covered, stop. Else, go to Step 4.

If the minimal sum of coverage quantity by sensors α is not 1, we can simply modify the algorithm. The pattern is composed with selected points whose interval between the center point is $\left\lfloor \frac{r^h}{\alpha} \right\rfloor$ instead of r^h. This heuristic algorithm provides an upper bound of the branch-and-bound tree in a short time.

4.5. Branch-and-cut procedure

4.5.1. Cutting plane algorithm

The cutting plane algorithm generates useful valid inequalities. We avoid adding a huge number of valid inequalities at the root node of a branch-and-bound tree. It may be a disadvantage to add all valid inequalities we have found. Suppose that P is the polyhedron of MDSLP, and X is the set of points in P ($X = P \cap Z^{u^2 t}$). We assume that we know a family F of valid inequalities for X, $\pi x \le \tau_0$ with $(\tau, \tau_0) \in F$. The cutting plane procedure is described as follows

Step 1. Set $i = 0$ and $P_0 = P$.
Step 2. Solve the LP_i with $x \in P_i$. Let x_i denote the optimal solution.
Step 3. If $x_i \in Z_{u^2 t}$, stop. x_i is the optimal solution of the MDSLP.
Step 4. If $x_i \notin Z_{u^2 t}$, try to find a valid inequality $(\tau_i, \tau_{i,0}) \in F$ such that $\tau_i x_i > \tau_{i,0}$.
Step 5. If $(\tau_i, \tau_{i,0}) \in F$ can be found, augment $P_{i+1} = P_i \cap \{x \mid \pi_i x \le \pi_{i,0}\}$. Otherwise, stop.
Step 6. Increment $i \leftarrow i+1$.
Step 7. Go to Step 2.

4.5.2. Branch –and-cut procedure.

The branch-and-cut procedure is a combination of cutting plane algorithm and branch-and-bound. After solving the LP relaxation in branch node, we try to find a violated cut by using the cutting planes. Each branch node has a formulation of LP_i which fixes a certain variable to 0 or 1. The descendent nodes inherit the bounds of the variables and violated cuts from their ancestor node.

Suppose a certain problem LP_i provides the optimal solution x whose objective value z is less than incumbent value. If x is integral, the incumbent solution is updated to x. Else, we branch into two nodes.

When we select another node to solve after making branch, the branching strategy that is used in this thesis is depth-first search. If the current node is not pruned, the next node considered is one of its descendents. Otherwise, we go back on the path from this node to the root until we find a node that has a child that has not been considered yet. The main motivation for this is to find feasible solutions quickly. When we make branching, the variable selection rule that is used is maximum integer infeasibility. We select one fractional variable with value closet to 0.5. If the tie occurs, we select one variable that has a higher coefficient of the objective. The two new branch nodes are generated by setting the value of the variable to 0 and 1, respectively.

5. Computational Results

In this chapter, we state the results we have obtained by applying the algorithm presented in the previous sections to the test problems.

5.1. Test problems

We assume that we have 3 types of sensors in test problems. Each type of sensor owns its cost and range. Table 5-1 shows the information about sensor type. Test problems are divided into three classes. The first one is problem A that includes a type of sensor, which is type1. The second one is problem B that includes two types of sensors, which are type 1 and type 2. The last one is problem C that includes three types of sensors, which are type 1, type 2, and type 3. Table 5-2 describes them. We test two cases of the minimal sum of coverage quantity by sensors α. At first, we test problems with $\alpha = 1$, and then, test problems with $\alpha = 2$. So, we test 6 problems. The size of sensor field varies from 2 to 13. Each test problem is considered with these various sizes.

Table 5-1. Information about sensor type

Sensor type	Cost ($10)	Range (10m)
Type 1	2	2
Type 2	3	4
Type 3	4	6

Table 5-2. The ID of test problem

ID	t	Sensor type
A	1	Type 1
B	2	Type 1 ,2
C	3	Type 1, 2, 3

* t : the number of sensor types

To compare our algorithm, we test two control groups. The first one runs with branch-and-bound procedure which does not provide a cut generation. The second one runs with branch-and-cut procedure without isomorphism pruning.

5.2. LP solver

For the computational study, we coded the procedure in C language. We used CPLEX 9.0 concert technology. All problems were tested on a AMD Athlon™ 64 X2 Dual Core(2.01GHz).

5.3. Computational results

The results for test problems are summarized at Table 5-3. In this table, the heading α refers to the minimal sum of coverage quantity by sensors. The heading u refers to the number of grid points in width (length) of the sensor field. The headings Z_{IP}, Z_{LP} refer to the optimal solution of IP, and the objective value of LP relaxation of MDSLP at the root node. GAP is defined as follows:

$$GAP(\%) = \frac{Z_{IP} - Z_{LP}}{Z_{IP}} \times 100$$

The heading UB is the upper bound which is obtained by heuristic at the root node. The heading #cuts and #B&C means the number of cuts generated, and the number of nodes generated in the branch-and-cut procedures. Finally, Time refers to the accumulated

CPU time needed to solve the problem until the optimal solution is obtained.

We could confirm our algorithm provides an optimal solution in a short time.

Table 5-3. Results for test problems

ID	α	u	Z_{IP}	Z_{LP}	GAP(%)	UB	#cuts	#B&C	Time(sec)
A	1	2	4	4	0	4	0	0	0.063
A	1	3	8	7.656	4.3	10	0	0	0.031
A	1	4	14	12.05	13.9	16	4	0	0.031
A	1	5	18	17.04	5.31	26	13	0	0.031
A	1	6	26	23.56	9.38	36	20	12	0.078
A	1	7	32	30.21	5.61	56	24	10	0.109
A	1	8	42	39	7.15	64	195	630	1.078
A	1	9	50	47.37	5.26	82	215	575	1.265
A	1	10	64	57.88	9.57	100	303	6751	18.391
A	1	11	72	68.57	4.76	122	286	1052	3.968
A	2	2	8	8	0	8	0	0	0.078
A	2	3	16	16	0	18	0	0	0.031
A	2	4	26	26	0	32	0	0	0.031
A	2	5	38	36.16	4.85	50	0	0	0.047
A	2	6	52	48.51	6.71	72	19	0	0.062
A	2	7	68	63.86	6.1	98	86	73	0.219
A	2	8	88	79.23	9.97	128	180	5272	3.453
A	2	9	106	98.07	7.48	162	231	7261	8.141
A	2	10	130	118.8	8.61	200	288	804439	14015.9
A	2	11	152	139.8	8.06	242	351	987646	22597.8
B	1	2	4	3.489	12.8	4	0	0	0.031
B	1	3	6	4.85	19.2	6	9	0	0.031
B	1	4	9	6.59	26.8	12	12	0	0.062
B	1	5	12	9.549	20.4	18	9	4	0.046
B	1	6	15	13.35	11	24	63	68	0.171
B	1	7	20	15.99	20	24	297	1567	2.406
B	1	8	24	19.03	20.7	33	387	15215	27.469
B	1	9	27	23.55	12.8	42	489	165317	1181.45
B	1	10	33	29.34	11.1	54	603	931182	25026.2
B	2	2	8	6.978	12.8	8	0	0	0.078
B	2	3	12	9.821	18.2	12	49	37	0.02
B	2	4	15	13.38	10.8	21	10	0	0.047
B	2	5	22	19.33	12.1	33	57	89	0.05
B	2	6	29	26.71	7.91	36	219	2348	1.703
B	2	7	36	32.56	9.55	48	297	74580	185.781
B	2	8	42	38.45	8.44	60	387	17338	34.109
C	1	2	4	3.489	12.8	4	0	0	0.031
C	1	3	6	4.85	19.2	6	9	0	0.047
C	1	4	7	6.282	10.3	8	0	0	0.031
C	1	5	8	7.673	4.09	12	0	0	0.031
C	1	6	12	9.341	22.2	20	99	93	0.187
C	1	7	16	11.65	27.2	24	297	1780	2.078
C	1	8	19	15.31	19.4	24	387	1243	4.25
C	1	9	22	18.75	14.8	32	489	26074	92.187
C	1	10	26	20.96	19.4	36	603	664740	10569.8
C	1	11	28	23.4	16.4	44	729	92833	808.563
C	1	12	31	26.41	14.8	48	867	47286	589.204
C	2	2	8	6.978	12.8	8	0	0	0.047
C	2	3	11	9.821	10.7	12	4	20	0.046
C	2	4	14	12.56	10.3	20	9	7	0.062
C	2	5	16	15.47	3.31	20	0	0	0.046
C	2	6	22	18.68	15.1	32	219	582	0.562
C	2	7	27	23.42	13.3	40	163	76	0.39
C	2	8	32	30.96	3.24	48	12	0	0.078

6. Concluding Remarks

In this thesis, we have introduced an optimization algorithm to solve MDSLP. We provided the new formulation. MDSLP has a large number of constraints, which are knapsack constraints and generalized upper bound (GUB) constraints. We used branch-and-cut algorithm with Gomory fractional cuts, cover cuts, and GUB cover cuts. MDSLP has a large number of symmetries, so we can consider one of them, and ignore the others. When we execute branch-and-cut procedure, we check that new branch node is

isomorphic to the previously defined nodes or not. If new branch node is isomorphic, then we can prune this node. The proposed algorithm solves the problem in a reasonable time.

The sensor location problem can be applied to various coverage problems. This algorithm is one of the most basic approaches to solve coverage problem. It is expected to apply to various extensions. It is possible to solve problem with arbitrary sensor field. In the telecommunication networks, we can consider the communication between sensors. These kinds of research are more realistic.

References

[1] Araque, J.R., Kudva, G., Morin, T.L., and Pekny, J.F. (1994). A branch-and-cut algorithm for vehicle routing problems. *Annals of Operations Research*, 50, 37–59.

[2] Balas, E., Ceria, S., and Cornue'jols, G.. (1996). Mixed 0–1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42(9), 1229–1246.

[3] Brooks, R.R. and Iyengar, S.S. (1998). Multi-sensor fusion: fundamentals and applications with software. Upper Saddle River, N.J.: Prentice Hall.

[4] Chakrabarty, K., Iyengar, S.S., Qi, H., and Cho, E. (2002). Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transactions on Computers*, 51, 1448-1453.

[5] Chiu, P.L. and F.Y.S.Lin (2004). A simulated annealing algorithm to support the sensor placement for target location. *Proc. IEEE CCECCE*, 867-870.

[6] Dhillon, S.S. and Chakrabarty, K. (2003). Sensor placement for effective coverage and surveillance in distributed sensor networks. *Proc. IEEE WCNC*, 3, 1609-1614.

[7] Dhillon, S.S., Chakrabarty, K., and Iyengar, S.S. (2002). Sensor placement for grid coverage under imprecise detections. *Proc. 15th International Conference on Information Fusion*, 2, 1581-1587.

[8] Estrin, D., Govindan, R., Heidemann, J., and Kumar, S. (1999). Next century challenges: scalable coordination in sensor networks. *Proc. ACM/IEEE Int'l Conf. Mobile Computing and Networks*.

[9] Fischetti, M., Toth, P., and Vigo, D. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3), 378-394.

[10] Frank, Y. S. Lin, and Chiu, P.L. (2005). A near-optimal sensor placement algorithm to achieve complete coverage/discrimination in sensor networks, *IEEE Communications Letters*, 9(1), 43-45.

[11] Garey, M., and Johnson, D. (1979). Computers and intractability: a guide to the theory of NP-completeness. San Francisco: W. H. Freeman.

[12] Gu, Z., Nemhauser, G.L., Savelsbergh, M.W.P. (1998). Lifted cover inequalities for 0-1 integer programs: computation. IJOC 10, 427–437.

[13] Hoffman, K., Padberg, M.W. (1993). Solving airline crew scheduling problems by branch and cut. *Management Science*, 39, 657–682.

[14] Iyengar, S.S., Prasad, L., and Min, H. (1995). Advances in distributed sensor technology. Englewood Cliffs, N.J.: Prentice Hall.

[15] Kahn, J.M., Katz, R.H., and Pister, K.S.J. (1999). Mobile networking for smart dust. *Proc. ACM/IEEE Int'l Conf. Mobile Computing and Networks*.

[16] Kaparis, K. and Letchford, A.N. (2005). A cut-and-branch algorithm for the multidimensional knapsack problem. . Lancaster University.

[17] Margot, F. (2002). Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94, 71–90.

[18] Margot, F. (2003). Exploiting orbits in symmetric ILP. *Mathematical Programming* Ser.B, 98, 3–21.

[19] Nemhauser, G.L. and Wolsey, L.A. (1998). Integer and combinatorial optimization. New York: John Wiley and Sons, Ltd..

[20] O'Rourke, J. (1987). Art gallery theorems and algorithms. New York: Oxford Univ. Press.

[21] Padberg, M. and Rinald, G.. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33, 60-100.

[22] Raaphorst, S. (2004). Branch-and-cut for symmetrical ILPs and combinatorial designs. A master's thesis. University of Ottawa.

[23] Sebastiani, R., Giorgini, P., and Mylopoulos, J. (2004). Simple and minimum-cost satisfiability for goal models. In 16th International Conference on Advanced Information Systems Engineering, Riga, Latvia.

[24] Wolsey, L.A. (1998). Integer programming, New York: Wiley.