An Efficient Genetic Algorithm for Maximum Coverage Deployment in Wireless Sensor Networks

Yourim Yoon and Yong-Hyuk Kim, *Member, IEEE*

***Abstract*—Sensor networks have a lot of applications such as battlefield surveillance, environmental monitoring, and industrial diagnostics. Coverage is one of the most important performance metrics for sensor networks since it reflects how well a sensor field is monitored. In this paper, we introduce the maximum coverage deployment problem in wireless sensor networks and analyze the properties of the problem and its solution space. Random deployment is the simplest way to deploy sensor nodes but may cause unbalanced deployment and therefore, we need a more intelligent way for sensor deployment. We found that the phenotype space of the problem is a quotient space of the genotype space in a mathematical view. Based on this property, we propose an efficient genetic algorithm using a novel normalization method. A Monte Carlo method is adopted to design an efficient evaluation function, and its computation time is decreased without loss of solution quality using a method that starts from a small number of random samples and gradually increases the number for subsequent generations. The proposed genetic algorithms could be further improved by combining with a well-designed local search. The performance of the proposed genetic algorithm is shown by a comparative experimental study. When compared with random deployment and existing methods, our genetic algorithm was not only about twice faster, but also showed significant performance improvement in quality.**

***Index Terms*—Genetic algorithm, maximum coverage, sensor deployment, solution space.**

I. Introduction

THE scope of wireless sensor network applications has been expanding beyond its original military purposes such as surveillance, target identification, and location tracking, to social purposes, such as disaster intervention, ecological environment and habitat monitoring, status identification for production equipment, emergency search-and-rescue, and medical care [1], [36]. To build sensor networks, economic factors as well as technical factors must be considered. Although the price of sensors gradually decreases as the sensor technology evolves, cost considerations still make it necessary to cover a wide range of a target area with a minimum number of sensors, which can be accomplished by efficient deployment of the sensors.

For a small number of sensors, manually placing them after considering their sensing range can be effective. But as most sensor network applications deal with a large number of sensors, many studies for efficient placement under various situations and environments have been conducted [3], [6], [8], [10], [15], [18], [22], [24], [35], [37], [38], [41], [46], [47], [49], [50], [54]–[57]. Wang [45] presents a deep survey of coverage problems in sensor networks.

There have been studies to expand the coverage of sensors with static sensors, which have limited movement [14], [15], and studies with mobile sensors, which can move within a certain area [23], [31], [39], [56]. There has also been a study to reduce costs and expand the coverage with a proper mix of relatively cheap static sensors, and expensive mobile sensors [46].

An important objective of surveillance sensor networks is to effectively monitor and detect targets of interest [51]. The optimal sensor emplacement enables us to minimize manpower and time, and to acquire accurate information on target situation and movement. For the optimal sensor emplacement, we confront the problem of maximizing coverage and minimizing the number of sensors. It is a type of the minimum set cover problem, which is known to be NP-hard [20]. Most of the earlier studies for this problem have focused on minimizing the number of sensors to cover the whole sensor field [9], [45]. Instead, we consider here the dual problem of maximally covering the sensor field for a given type and number of sensors. It has many applications in general sensor deployment problems, as the number of sensors that can be used is finite due to cost considerations, but covering as broad an area as possible is still necessary and achievable using sensor networks. In real world, it frequently appears that the sensor field is too large to fully cover with a limited number of sensors [37]. For instance, this problem actually appeared when Republic Of Korea Army (ROKA) deployed sensors to surveil military demarcation line (MDL) and five strategic important islands in the Yellow Sea for national defense.

However, to the authors' best knowledge, this problem has not been studied in the literature. Moreover this problem cannot be solved by deterministic methods such as the circle packing algorithm [30] because the sensing range may not be equal. We define the problem formally and propose an efficient

Manuscript received May 21, 2012; revised October 21, 2012 and January 8, 2013; accepted February 2, 2013. Date of publication April 12, 2013; date of current version September 11, 2013. The present research was conducted by the Research Grant of Kwangwoon University in 2012. This work was supported in part by LG Electronics, Inc, and by the Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education, Science, and Technology under Grant 2012-0001855. This paper was recommended by Associate Editor S. X. Yang. *(Corresponding author: Y.-H. Kim.)*

Y. Yoon is with the Future IT Research and Development Laboratory, LG Electronics, Umyeon Research and Development Campus, Seoul 137-724, Korea (e-mail: yryoon@soar.snu.ac.kr).

Y.-H. Kim is with the Department of Computer Science and Engineering, Kwangwoon University, Seoul 139-701, Korea (e-mail: yhdfly@kw.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2013.2250955

genetic algorithm for solving the problem. In addition, the characteristics of the problem space is discussed.

There have been some studies by applying genetic algorithms (GAs) to sensor deployment problems [11], [25], [37], [48], [50], [55]. In addition to the previous work, we analyze the problem, its representation, and its properties especially in relation to genetic algorithms, and propose a new methodology that is more tailored to the properties of genetic algorithms. An efficient evaluation method and a novel normalization technique are also presented. Xu and Yao [50] used a decoding method involving normalization that decoded chromosomes into order-based representation. We devise a more problem-specific normalization method and study suitable representation schemes in relation to the structure of the solution space.

The remainder of this paper is organized as follows. In Section II, we present the maximum coverage sensor deployment problem. Some assumptions and the definition of the problem are presented in Section II-A and Section II-B, respectively. Test data set generated for this paper is described in Section II-C. Analysis of representation schemes and the problem space is given in Section II-D. In Section III, GAs designed based on the analyzed properties are presented. The genetic framework used is described in Section III-A, and an evaluation method and normalization technique are provided in Section III-B and Section III-C, respectively. Section IV presents the effectiveness of the proposed genetic algorithm through experimental results, and conclusions are given in Section V.

II. Maximum Coverage Sensor Deployment

A. Assumptions

Sensor coverage models measure the sensing capability and quality by capturing the geometric relation between a point and sensors. Among them, the Boolean disk coverage model might be the most widely used sensor coverage model in the literature [45]. The coverage function of the model is given by

$$f(d(s,x)) = \begin{cases} 1, & \text{if } d(s,x) \le r_s \\ 0, & \text{otherwise} \end{cases}$$

where $d(s, x)$ is the Euclidean distance between a sensor s and a point x, and the constant r_s is called *sensing range*. Indeed, this function defines a disk centered at the sensor with the radius of the sensing range. This model is an omnidirectional coverage model. All points within such a disk have a coverage measure of 1, and are said to be covered by this sensor. All points outside such a disk have a coverage measure of 0, and are said not covered by this sensor. The sensing range r_s is used to characterize the sensing capability of a sensor. In general, different sensor types are assumed to have different sensing ranges.

According to the subject to be covered, coverage in sensor networks can be classified into three types: area coverage, barrier coverage, and target coverage [7]. In this paper, we adopt the Boolean disk coverage model and deal with area coverage problem, which equally treats every point in the sensor field. In this paper, as a new trial, we focus on the problem of maximizing the covered area of the sensor field with a given number of sensors. This problem can not only be directly used to solve previous area coverage problem that minimizes the number of sensors to cover the whole sensor field by adjusting the number of sensors in a binary-search manner, but can also be transformed into other types of coverage problems, e.g., barrier coverage problem in [37], which constructs a barrier that maximizes the coverage of vehicle paths across the sensor field with a given number of sensors. Although we assume a specific coverage problem, the proposed methodology is easily applicable to other types of coverage problems by just changing the evaluation function.

B. Problem Formulation

We define the maximum coverage sensor deployment problem (MCSDP) in the following way. There are k types of static sensors, and each type of sensors can cover a given area with an arbitrary fixed radius $r_1, r_2, \ldots, r_k$. The total number of sensors is n , and assuming that there is at least one sensor for each type of sensors, sensors for type i are numbered from n_i to $n_{i+1} - 1$ with $n_1 = 1$ and $n_{k+1} = n + 1$. That is, for given $k + 1$ numbers $n_1(= 1) < n_2 < \cdots < n_{k+1}(= n + 1)$, there are $n_{i+1} - n_i$ sensors for sensor type i, where $i = 1, 2, \ldots, k$. The objective of the problem is to find locations $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ for all n sensors that produce the maximum coverage for a given domain A.

Consider a circle $C_r(x, y)$ for which the center is (x, y) and the radius is r, i.e., $C_r(x, y) := \{(v, w) \in \mathbb{R}^2 : (v - x)^2 + (w - y)^2 \le r^2\}$, the problem can be formalized as follows:

$$\begin{aligned} \text{maximize} \quad & \text{area}\left(\bigcup_{i=1}^{k}\bigcup_{j=n_i}^{n_{i+1}-1} C_{r_i}(x_j, y_j) \cap A\right) \\ \text{subject to} \quad & (x_i, y_i) \in A, \quad i = 1, 2, \ldots, n, \end{aligned}$$

where r_is are positive real numbers for $1 \le i \le k$, n_is are nondecreasing positive integers for $1 \le i \le k + 1$ with $n_1 = 1$ and $n_{k+1} = n + 1$, and A is a domain in 2-D Euclidean space.

The basic idea of our notation is from [33], which is well known for introducing the multidimensional multiple-choice knapsack problem. To help the readers' understanding of our notation, Fig. 1 illustrates an example case, in which there are 3 types of sensors to which 2, 3, and 4 sensors belong, respectively.

C. Test Data Set

We assume that the domain A to be covered with sensors is a square region in 2-D Euclidean space, $[0, 100] \times [0, 100]$. We used 3 types of sensors for which the detection radius for each sensor type is r_1, r_2, and r_3, respectively. The values of radii are determined by the criteria that $r_1 = 0.8 \times r_2$ and $r_2 = 0.8 \times r_3$. The number of sensors for each type is determined to match the given tightness ratio α, i.e., to make the total of each sensor's detection area be a product of α and the area of given domain A. In this manner, 15 test instances given in Table I were generated. The total number n of sensors varies from 17 of instance S1-0.7 to 130 of instance S5-0.9.

Main parameters in our test data generation are modeled from literature including real-world sensor information. Lamm

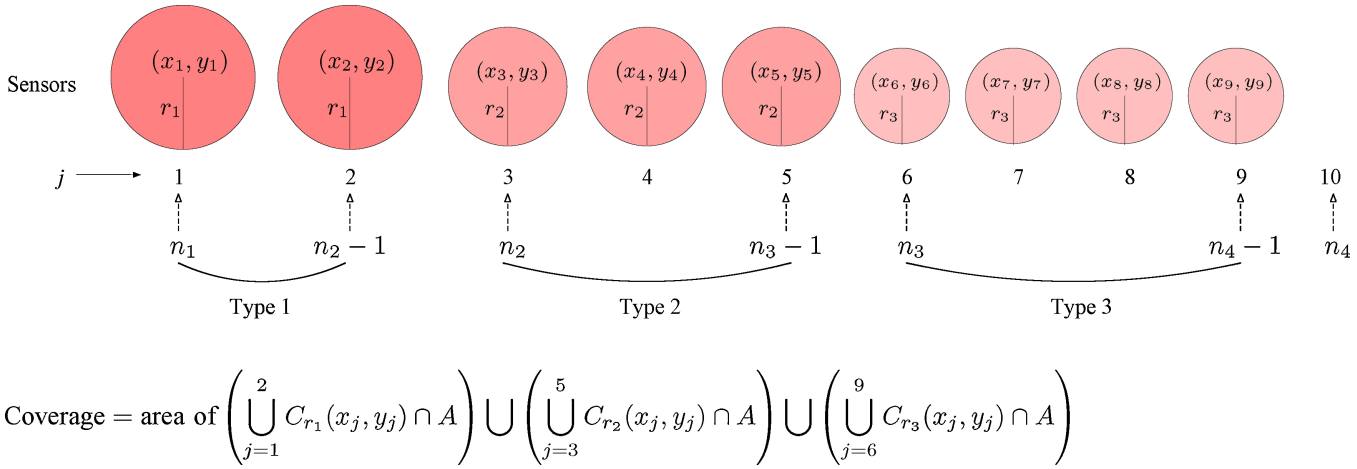


Fig. 1. Illustration of the used notation.

TABLE I
TEST INSTANCES

Instance	r_1	n_1	r_2	n_2	r_3	n_3	n	α
S1-0.7	14.00	5	11.20	5	8.96	7	17	0.68
S2-0.7	12.00	6	9.60	8	7.68	10	24	0.69
S3-0.7	10.00	8	8.00	12	6.40	16	36	0.70
S4-0.7	8.00	12	6.40	18	5.12	27	57	0.70
S5-0.7	6.00	22	4.80	32	3.84	47	101	0.70
S1-0.8	14.00	5	11.20	6	8.96	10	21	0.80
S2-0.8	12.00	6	9.60	9	7.68	14	29	0.79
S3-0.8	10.00	9	8.00	13	6.40	19	41	0.79
S4-0.8	8.00	14	6.40	20	5.12	29	63	0.78
S5-0.8	6.00	25	4.80	36	3.84	55	116	0.80
S1-0.9	14.00	6	11.20	7	8.96	10	23	0.90
S2-0.9	12.00	7	9.60	11	7.68	14	32	0.89
S3-0.9	10.00	11	8.00	14	6.40	21	46	0.90
S4-0.9	8.00	16	6.40	23	5.12	34	73	0.90
S5-0.9	6.00	28	4.80	41	3.84	61	130	0.90

r_1, r_2, and r_3 are radii of sensors according to sensor type.
n_1, n_2, and n_3 are the numbers of sensors according to sensor type.
n is the total number of sensors.
α is the tightness ratio.

[28], Srour [40], and Seo *et al.* [37] dealt with 3 types of sensors named acoustic sensor, seismic sensor, and forward-looking infrared radar (FLIR), which are used in battlefield surveillance of the U.S. Army. The detection radius of seismic sensor is about 0.8 times that of acoustic sensor. Similarly, the detection radius of FLIR is about 0.8 times that of seismic sensor. For the tightness ratio α, we referred to the work of Zou and Chakrabarty [56], [57]. They used just one type of sensors and determined the number of sensors with the tightness ratio 0.7. We extended their test data with more sensor types (3 types) and more various tightness ratios (0.8 and 0.9) including 0.7 because the price of sensors is expected to be lowered as the technology for manufacturing sensors advances. Given a budget, we will be able to use more sensors to cover a given domain. So additionally we considered larger tightness ratios (0.8 and 0.9) than 0.7 for an extended study.

D. Representation and Solution Space

The most natural representation for the MCSDP is an array of 2-D coordinates of all sensors. In this case, the coverage

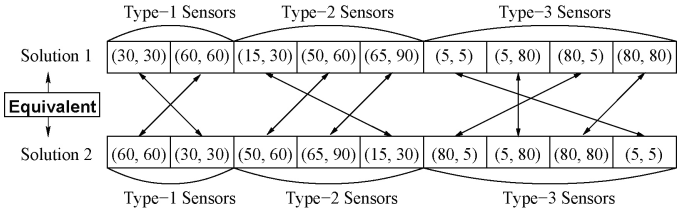


Fig. 2. Two equivalent solutions.

does not change when the index order among the same type of sensors is changed. That is, there may be more than one different encoding for the same solution. Thus the space of encodings (genotypes) and the space of solutions (phenotypes) are different.

Let G be the space of encodings, and let g and h in G be $(g_1, g_2, \ldots, g_n)$ and $(h_1, h_2, \ldots, h_n)$, respectively. Here, g_i and h_i are 2-D locations or coordinates of sensor i though they are denoted as single symbols. Then, the following relation $\sim$ can be defined on G.

Definition 1: $x \sim y$ if and only if there is a permutation $\sigma_i \in \Sigma_{n_{i+1}-n_i}$ such that $\sigma_i(g_{n_i}, g_{n_i+1}, \ldots, g_{n_{i+1}-1}) = (h_{n_i}, h_{n_i+1}, \ldots, h_{n_{i+1}-1})$ for each $i \leq k$, where $\Sigma_{n_{i+1}-n_i}$ is the set of permutations with size $n_{i+1} - n_i$ and $\sigma(x)$ represents the permuted x by a permutation σ.

Proposition 1: The relation $\sim$ is an equivalence relation.

Proof: For each i, $\langle \Sigma_{n_{i+1}-n_i}, \circ \rangle$ forms a symmetric group $S_{n_{i+1}-n_i}$, where $\circ$ is the function composite operator [19]. Then their direct product $P := \Pi_{i=1}^{k} S_{n_{i+1}-n_i}$ is also a group [19]. So the identity element $e \in P$, which means that the relation $\sim$ is reflexive. If $\sigma \in P$, then there exists its inverse $\sigma^{-1} \in P$, which means that the relation $\sim$ is symmetric. Group P is closed under the operator $\circ$. That is, if $\sigma_1, \sigma_2 \in P$, then $\sigma_1 \circ \sigma_2 \in P$. So the relation $\sim$ is transitive. Therefore, the relation $\sim$ is an equivalence relation. □

$n_{i+1} - n_i$ is the number of type-i sensors. $(g_{n_i}, g_{n_i+1}, \ldots, g_{n_{i+1}-1})$ is the array of locations of type-i sensors. For example, Solution 1 and Solution 2 in Fig. 2 are equivalent. The number of elements equivalent to an element $g \in G$ are $(n_2 - n_1)!(n_3 - n_2)! \cdots (n_{k+1} - n_k)!$.

The equivalence relation defined by Definition 1 is meaningful in that the solution space can be considered as the quotient space $G/\sim$, which is the set of equivalent classes of elements in G.

Meta-heuristics such as genetic algorithms search the space of encodings G. However, the intrinsic solution space of the problem is $G/\sim$, which has to be searched by heuristics. Search in encoding space may behave differently from that in solution space, and the space we want to search is solution space $G/\sim$. Hence, new methods need to be devised for efficient search of the solution space because using given encodings themselves only searches the genotype space, which is essentially different from the intrinsic solution space. A novel normalization method to overcome this problem in GAs will be discussed in Section III-C.

III. Proposed Genetic Algorithms

A. Genetic Framework

We describe the genetic framework we use for our study. Let N be the population size. A collection of $N/2$ pairs is randomly composed, and crossover and mutation are applied to each pair, generating $N/2$ offspring. Parents and newly generated offsprings are ranked, and the best N individuals among them are selected for the population in the next generation. In all our experiments, a population size of 50 was used ($N = 50$). Our GAs terminate after 1,000 generations. As crossover and mutation operators, BLX-α [4], [17] and Gaussian mutation [21] (described below) are used, respectively.

- BLX-α: Offspring $z = (z_1, z_2, \ldots, z_{2n})$ is generated from parents $x = (x_1, x_2, \ldots, x_{2n})$ and $y = (y_1, y_2, \ldots, y_{2n})$, where z_i is uniformly randomly chosen from the interval $[\min\{x_i, y_i\} - \alpha I, \max\{x_i, y_i\} + \alpha I]$, where $I = |x_i - y_i|$. Since the value 0.5 for α is widely used [42], [53], we also used the same value in our GAs.
- Gaussian mutation: The i-th gene x_i of an individual is mutated by $N(0, \sigma_i)$ with mutation rate p_m, where $N(0, \sigma_i)$ is an independent random Gaussian number with the mean of zero and the standard deviation of σ_i. In our GAs, σ_i is fixed to 50 - a half of width of a given domain $A = [0, 100] \times [0, 100]$ -and p_m is fixed to $0.1/n$, where n is the total number of sensors.

B. Evaluation

The evaluation function of the MCSDP corresponds to calculating the total area covered with emplaced sensors. That is, the evaluation value is the area of union of circles centered around each sensor's location, and the radius is each sensor's detection range. It is not easy to obtain the closed formula for the area since circles with different radii can overlap each other. Thus, we practically evaluated a given deployment by a Monte Carlo simulation method. Monte Carlo methods perform repeated random sampling to compute results. Let X be the covered area, i.e., $\bigcup_{i=1}^{k} \bigcup_{j=n_i}^{n_{i+1}-1} C_{r_i}(x_j, y_j) \cap A$. Then,

$$\begin{aligned} \text{area}(X) &= \int_A I_X(x, y) dx dy \\ &= \lim_{L\to\infty} \frac{\text{area}(A)}{L} \sum_{l=1}^{L} I_X(\tilde{x}_l, \tilde{y}_l), \end{aligned}$$

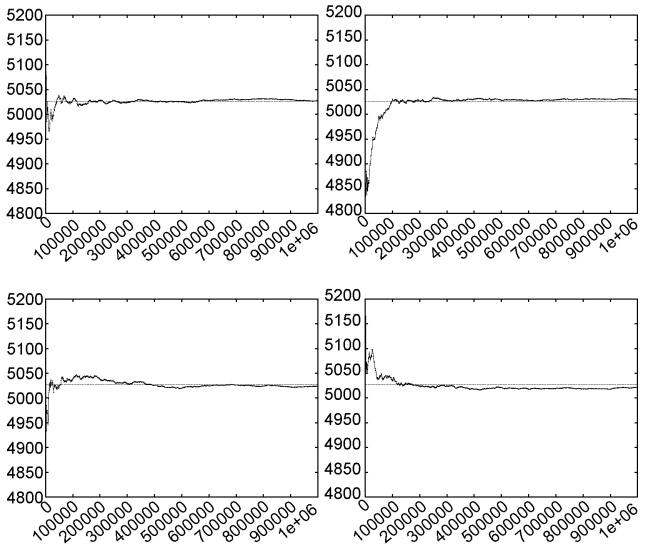


Fig. 3. Evaluation values according to the number of samples (four independent runs).

where $I_X(\cdot)$ on A is the indicator function defined as

$$I_X(x, y) = \begin{cases} 1 & \text{if } (x, y) \in X \\ 0 & \text{otherwise} \end{cases}$$

and points $(\tilde{x}_l, \tilde{y}_l)$s are independently and uniformly chosen on A. To compute $I_X(\tilde{x}_l, \tilde{y}_l)$ for each $l = 1, 2, \ldots, L$, it is enough to check whether or not a sampled point $(\tilde{x}_l, \tilde{y}_l)$ is in the sensing range of a sensor $i = 1, 2, \ldots, n$. So the worst-case time complexity of Monte Carlo-based evaluation becomes $O(nL)$. Here, the number of random samples (L) needs to be determined with prudence.

We plotted the values calculated using a Monte Carlo method according to the increasing number of samples for a simple case - the case of only one sensor with range 40 placed in the center $(50, 50)$ of the domain $A = [0, 100] \times [0, 100]$. Four independent trials for the case are shown in Fig. 3. The accurate evaluation value is $40^2\pi$ ($\simeq$ 5026.55). The evaluation result approaches the value as the number of samples increases. As the sampling number increases, the results by Monte Carlo method become more accurate, but excessively large sampling numbers lead to a large computation cost. So we determined the number of random samples 100,000 considering practicality. In Fig. 3, evaluation values tend to be stable after about 100,000 random samples. Although more random samples yield more accurate results, exceedingly precise evaluation is not essential during the runs of genetic algorithms; rather some mutation effect by a little imprecise evaluation may lead to a positive result. However, final solutions found by our genetic algorithms were re-evaluated with 1,000,000 random samples after the algorithm terminated for precise verification of the algorithm.

Final solutions should be fairly assessed, but intermediate solutions do not need to be accurately calculated using large computation costs during the run of genetic algorithms. Decreasing costs to evaluate solutions might be more efficient than getting accurate values by using a large number of random samples. Inspired by this idea, we also implemented a method

that starts with a small number of random samples and then increases the number for subsequent generations, instead of using a fixed number of random samples. The experimental performance of this method is reported in Section IV.

C. Normalization

As described in Section II-D, the encoding of the MCSDP is redundant, and hence phenotype space and genotype space are different. Phenotype space P corresponds to a quotient space $G/\sim$, where G is the genotype space. Redundant representations may lead to severe loss of performance in genetic algorithms, in particular, with respect to traditional crossovers (crossovers defined on binary and nonbinary strings using crossover masks) [12]. To alleviate the problems caused by redundant representations, a number of methods such as adaptive crossover have been proposed [16], [29], [34], [43]. Among them, a technique called normalization[1] is representative. Normalization transforms the genotype of a parent into another genotype (with the same phenotype) to be as similar as possible to the genotype of the other parent before performing traditional crossover. There have been a number of successful studies using normalization (e.g., see [32]). Extensive surveys about normalization appeared in [13], [52].

If we apply GAs to the MCSDP without normalization technique, GAs only search the genotype space and may not reflect the characteristics of the original solution space, i.e., the phenotype space. In the MCSDP, the phenotype is not changed when we change the index order among the same type of sensors. Then, the appropriate rearrangement of genes does not change the original solution and can help in efficient searching of the problem space at the same time. So we propose a novel normalization method tailored to the MCSDP to associate the search by GAs with the original solution space.

To verify that the normalization method actually works on the MCSDP, we performed experiments with three rearrangement strategies. Assume that one parent x is $((p_1, q_1), (p_2, q_2), \ldots, (p_n, q_n))$ and the other parent y is $((r_1, s_1), (r_2, s_2), \ldots, (r_n, s_n))$.

- RAND: Without rearrangement, the original arrangement of the second parent is chosen before recombination.
- MINDIST: We rearrange the genes of the second parent to minimize the distance sum, i.e., the value of $\sum_{i=1}^{n} d((p_i, q_i), (r_i, s_i))$. If we adopt the Euclidean distance as a distance for 2-D coordinates, the value becomes $\sum_{i=1}^{n} \sqrt{(p_i - r_i)^2 + (q_i - s_i)^2}$.
- MAXDIST: We rearrange the genes of the second parent to maximize the distance sum, i.e., the value of $\sum_{i=1}^{n} d((p_i, q_i), (r_i, s_i))$.

Here, MINDIST method is the closest to the meaning of normalization since both parents can be said to be similar to each other if the distance between them is small. Moreover, the minimum distance sum in the MINDIST method becomes a metric on the phenotype space $G/\sim$ by the following proposition.

[1]The term of *normalization* first appears in [26]. However, it is based on the adaptive crossovers proposed in [29], [34].

Proposition 2: Assume that $g = (g_1, g_2, \ldots, g_n)$ and $h = (h_1, h_2, \ldots, h_n)$ are in G, where g_i and h_i are 2-D coordinates for each $i = 1, 2, \ldots, n$. Let

$$D(g, h) := \sum_{i=1}^{n} d(g_i, h_i)$$

be a metric on G, where d is a metric on $\mathbb{R}^2$. Then,

$$\tilde{D}(g, h) := \min_{\sigma \in \prod_{i=1}^{k} S_{n_{i+1}-n_i}} D(g, \sigma(h))$$

becomes a metric on $G/\sim$.

Proof: Let σ be in the group $\prod_{i=1}^{k} S_{n_{i+1}-n_i}$. When we compute $D(g, h) := \sum_{i=1}^{n} d(g_i, h_i)$, since the number of summands is finite, summation order is irrelevant to the result. So $D(g, h) = D(\sigma(g), \sigma(h))$. Hence, σ is an isometry on G, and then $\prod_{i=1}^{k} S_{n_{i+1}-n_i}$ is an isometry subgroup. The relation $\sim$ becomes an equivalent relation derived from an isometry subgroup $\prod_{i=1}^{k} S_{n_{i+1}-n_i}$. Hence by [5], [52], $\tilde{D}(g, h)$ is a metric on $G/\sim$. □

So if MINDIST shows better performance than RAND, we can say that normalization is effective for solving the MCSDP using GAs.

MINDIST and MAXDIST normalization methods can be practically implemented using the Hungarian method [27] in $O(n^3)$ time.[2] To investigate what method is most effective, we performed two experiments.

- Experiment I
  1. N solutions are randomly generated. (N is set to be 100.)
  2. $N/2$ pairs of solutions are produced.
  3. Rearrangement is applied to the second parents by the three aforementioned methods.
  4. Crossover operator (BLX-0.5) is applied to each pair - the first parent and the rearranged second parent.
  5. The average and the standard deviation of the coverage of $N/2$ offspring are calculated.
- Experiment II
  1. N solutions with good quality are generated by generating $500N$ solutions randomly and choosing N solutions with the largest coverage. (N is set to be 100.)
  2–5. See steps 2–5 for Experiment I.

Tables II and III-C show the results of Experiment I and Experiment II, respectively. Both of the experiments show that MINDIST method is superior to the others as we expected, and we can expect good performance of GAs when using the MINDIST method as normalization. This result is reasonable because MINDIST best preserves parents' properties among the three methods by placing similar values at the same position of both parents. It is most consistent with the normalization concept. Preserving the properties of parents is one of the most essential elements in GAs.

In the next section, we will show that our MINDIST normalization has a great impact on the performance of GAs and it performs significantly better than existing heuristic

[2]We can also consider some fast heuristic algorithms for normalizing solutions approximately [2]. They can be implemented in $O(n^2)$ and experimental studies indicated that these methods are, in practice, very close to the optimum and much faster than the Hungarian method.

TABLE II
EFFECT OF NORMALIZATION - EXPERIMENT I

Instance	RAND		MINDIST		MAXDIST	
	Ave	SD	Ave	SD	Ave	SD
S1-0.7	4555.14	389.99	**4774.72**	313.63	4565.84	368.04
S2-0.7	4733.89	271.22	**4849.67**	284.77	4477.07	341.45
S3-0.7	4784.83	244.27	**4920.96**	203.68	4580.31	292.26
S4-0.7	4775.74	212.11	**4884.31**	188.31	4609.59	217.70
S5-0.7	4839.26	152.57	**4960.80**	142.63	4648.70	204.09
S1-0.8	5131.97	407.17	**5434.89**	352.74	5007.54	352.02
S2-0.8	5177.91	353.35	**5330.42**	302.47	5011.76	335.57
S3-0.8	5195.10	281.12	**5308.03**	223.24	4998.79	278.70
S4-0.8	5190.97	229.40	**5335.41**	196.79	4897.37	226.61
S5-0.8	5278.32	157.54	**5452.38**	125.74	4996.68	195.87
S1-0.9	5515.27	434.86	**5681.95**	444.56	5349.39	402.91
S2-0.9	5626.31	274.04	**5711.57**	283.13	5422.74	362.66
S3-0.9	5697.05	319.11	**5822.35**	305.86	5386.62	284.58
S4-0.9	5618.71	250.61	**5828.81**	264.26	5420.76	233.92
S5-0.9	5649.90	160.80	**5911.49**	177.28	5427.47	211.73

Results from 50 offspring.
RAND is the result of offspring produced without normalization.
MINDIST is the result of offspring produced by MINDIST normalization.
MAXDIST is the result of offspring produced by MAXDIST normalization.

TABLE III
EFFECT OF NORMALIZATION - EXPERIMENT II

Instance	RAND		MINDIST		MAXDIST	
	Ave	SD	Ave	SD	Ave	SD
S1-0.7	4788.38	340.58	**5074.33**	286.44	4482.43	365.67
S2-0.7	4831.57	271.83	**5072.68**	289.55	4595.23	332.75
S3-0.7	4867.32	242.17	**5124.40**	216.60	4622.70	304.62
S4-0.7	4796.20	208.09	**5093.34**	191.67	4678.24	215.59
S5-0.7	4858.20	151.56	**5089.86**	120.96	4634.58	190.59
S1-0.8	5318.32	322.41	**5621.51**	331.67	5176.42	397.68
S2-0.8	5251.36	289.78	**5529.64**	229.37	5037.10	297.24
S3-0.8	5269.63	286.44	**5503.98**	211.87	5070.90	272.64
S4-0.8	5222.64	208.00	**5507.94**	173.25	4975.61	251.98
S5-0.8	5303.85	143.14	**5578.47**	141.02	5055.60	209.74
S1-0.9	5667.49	369.57	**6051.07**	280.46	5306.03	362.13
S2-0.9	5638.65	290.27	**6038.94**	294.22	5346.22	386.38
S3-0.9	5565.41	243.49	**6032.84**	219.02	5466.10	285.22
S4-0.9	5643.53	231.37	**6038.18**	179.47	5429.12	238.20
S5-0.9	5723.24	147.24	**6024.84**	142.79	5459.76	162.14

Results from 50 offspring.
RAND is the result of offspring produced without normalization.
MINDIST is the result of offspring produced by MINDIST normalization.
MAXDIST is the result of offspring produced by MAXDIST normalization.

normalization, which was used for other problems, on large-scale instances.

IV. EXPERIMENTAL RESULTS

We conducted experiments on 15 instances presented in Table I (see Section II-C) to verify the performance of the proposed GAs. Tables IV and V show the results. Thirty trials were performed for each method and the results were averaged over the trials. "Time" is the average CPU seconds that were taken to perform 30 trials on an Intel Xeon CPU 2.4 GHz. In the table, PGA means a pure genetic algorithm without the proposed normalization and speed-up techniques. It followed the genetic framework presented in Section III-A and a Monte Carlo method was used with a fixed number of

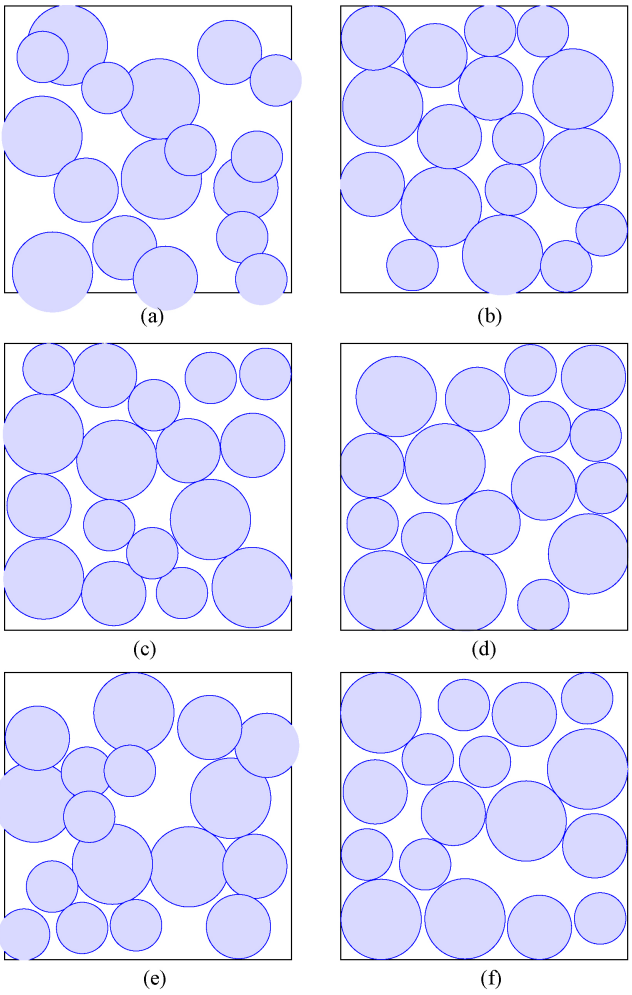


Fig. 4. Sensor deployment of instance S1-0.7 found by various methods. (a) RANDOM (Coverage: 5980.55). (b) PGA (Coverage: 6802.99). (c) MGA (Coverage: 6806.75). (d) OPTGA (Coverage: 6810.67). (e) Multi-start VFA (Coverage: 6512.01). (f) OPTHGA (Coverage: 6814.65)

random samples (100,000) to evaluate the solutions. To verify the effectiveness of the GA, we compared the results with a multistart method using randomly-deployed solutions [44]. As a single run, we repeatedly sampled a number of randomly-deployed solutions in the given domain $A = [0, 100] \times [0, 100]$ for an amount of time similar to the GA, and chose the best one. RANDOM in the table is the result. When we compare the results of RANDOM and PGA, PGA showed significantly better performance than RANDOM (see the results of one-tailed t-tests in PGA). It can be seen that GAs performed well on the MCSDP.

GA using Monte Carlo evaluation with an increasing number of random samples (called MGA) mentioned in Section III-B was also tested. In MGA, the initial number of random samples is 10,000 and increased by 10,000 per 100 generations. MGA was about two times faster than PGA, so the computational cost could be largely reduced using such a method. MGA uses the same framework as PGA except its evaluation time was reduced with the proposed speed-up technique. So the solution quality of MGA inherently cannot be better than that of PGA. Nevertheless, the results of

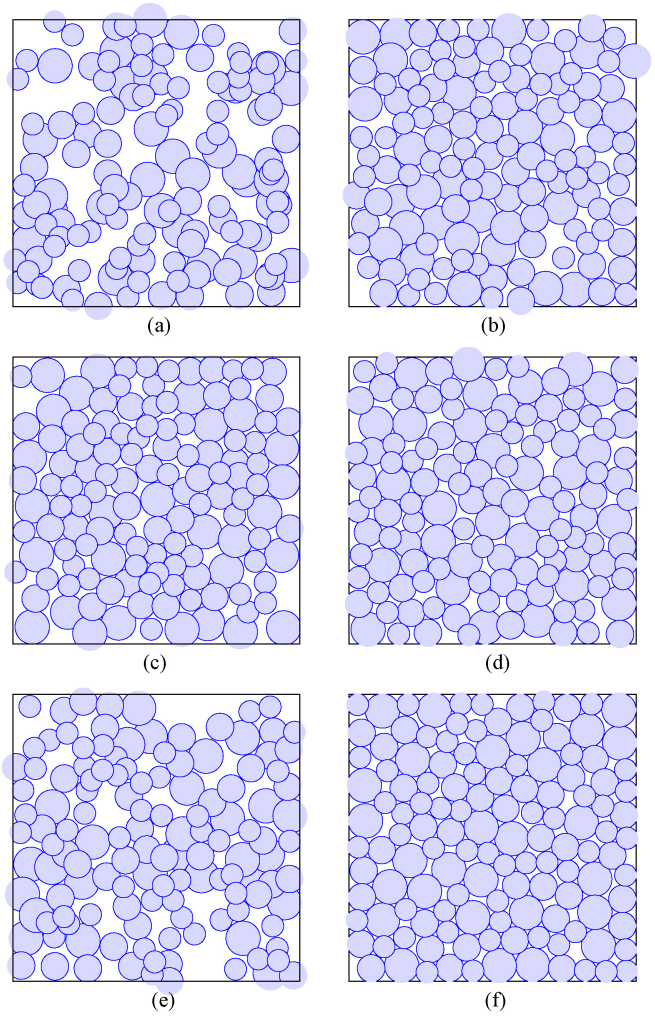


Fig. 5. Sensor deployment of instance S5-0.9 found by various methods. (a) RANDOM (Coverage: 6495.37). (b) PGA (Coverage: 8221.22). (c) MGA (Coverage: 8243.13). (d) OPTGA (Coverage: 8351.68). (e) Multi-start VFA (Coverage: 7344.68). (f) OPTHGA (Coverage: 8755.50).

Table IV show that MGA performs faster than PGA without degenerating solution quality. The performance of MGA was better in some cases (e.g., S4-0.9), and in other cases that of PGA was better (e.g., S5-0.9). However according to the two-tailed t-tests, PGA and MGA performances were very similar in quality (with p values more than 0.01). The only exception with p value less than 0.01 was the instance S4-0.8, but since MGA result is better than that of PGA on the instance, the benefits outweigh this exceptional case. So, we can say that MGA is superior to PGA in practical terms.

Next we compare these results with the GA that employs normalization. OPTGA in Table IV represents the GA with the proposed normalization. It follows the same genetic framework as MGA, which uses a Monte Carlo method with an increasing number of random samples to evaluate solutions, and rearranges the genes of the second parent to minimize the distance sum before applying recombination. Such normalization helps preserve the parents' properties by placing similar values in the same position of both parents and hence we can expect better results using the method. OPTGA showed the best performance when compared with the other methods (RANDOM, PGA, and MGA). The results of one-tailed t-tests for OPTGA show that it performs significantly better than MGA. Moreover, as can be seen in the relatively small values of standard deviation, OPTGA is more stable than GAs without normalization. Overall, OPTGA computation times were not long compared to MGA, with exceptions for instances S5-0.7, S5-0.8, and S5-0.9, which have many sensors. For instances S5-0.7, S5-0.8, and S5-0.9 with more than 100 sensors, the computational times of OPTGA were higher than those of MGA. It is because the normalization time becomes considerably large as the number of sensors increases.

As can be seen in the results of the t-tests, we can say that the discussed methods in ascending order of performance are RANDOM $\ll$ PGA $\approx$ MGA $\ll$ OPTGA with respect to quality; and RANDOM $\approx$ PGA $\ll$ MGA $\approx$ OPTGA for time efficiency, where "A $\ll$ B" means that B performs better than A and "A $\approx$ B" means that A and B perform similarly.

Finally we compare OPTGA with a well-known sensor deployment method and an existing heuristic normalization technique. Table V shows the results. A multistart method using the virtual force algorithm (VFA) [56], [57], whose details are given in Appendix, is adopted for comparison. As a single run, we repeatedly got the VFA's solutions from randomly-deployed ones for an amount of time similar to RANDOM and PGA, and chose the best one. Multistart VFA in the table is the result. OPTGA performed significantly better than Multistart VFA (see the results of one-tailed t-tests in OPTGA), i.e., Multistart VFA $\ll$ OPTGA.

We also compare our normalization with existing heuristic normalization. Seo *et al.* [37] used a normalization technique for barrier coverage problem. Stable marriage crossover in [37] used a heuristic for normalization between parents. Its basic idea is from [26], [29] for multiway graph partitioning. The difference from our normalization lies in that their method does normalization heuristically without considering the phenotype distance between solutions, hence it cannot get the optimal sensor matching. SMCGA in Table V represents the GA with normalization used in the stable marriage crossover. It follows the same framework as MGA except normalization. OPTGA performed better than SMCGA for all the instances. In particular, OPTGA showed a significantly better performance than SMCGA on large instances (S4-* and S5-*) (see the results of one-tailed t-tests in OPTGA).

The VFA itself can be considered as a local search for the proposed GAs. To further improve the performance of OPTGA, we combined OPTGA with the VFA. In OPTGA, we apply the VFA to each offspring after mutation step. "OPTHGA" in Table V represents the OPTGA hybridized with the VFA. OPTHGA significantly outperformed OPTGA for all the instances, without any loss of computing time, i.e., OPTGA $\ll$ OPTHGA. It is surprising that the results of OPTHGA were very close to the upper bounds (the last column of Table V). Because the upper bounds are computed under the assumption that there is no overlapping area between sensors, it means that there was little overlapping area between sensors in the sensor deployment obtained from OPTHGA. As

TABLE IV
EFFECT OF THE PROPOSED IDEA

Instance	RANDOM		PGA		t-test[1]	MGA		t-test[2]	OPTGA		t-test[3]	Upper
	Ave (σ)	Time*	Ave (σ)	Time*	p-value	Ave (σ)	Time*	p-value	Ave (σ)	Time*	p-value	bound
S1-0.7	5866.12 (65.74)	923	6747.01 (42.94)	868	2.05e-33	6754.37 (42.40)	473	5.09e-01	**6779.40** (24.93)	473	4.57e-03	6814.65
S2-0.7	5790.22 (71.52)	1234	6776.50 (37.52)	1107	1.63e-34	6787.21 (61.94)	616	4.24e-01	**6833.01** (39.33)	613	9.15e-04	6883.56
S3-0.7	5728.31 (77.67)	1733	6819.23 (65.58)	1664	7.66e-33	6855.70 (46.75)	868	1.90e-02	**6906.43** (37.05)	880	3.05e-05	6984.89
S4-0.7	5562.04 (37.68)	2300	6780.21 (48.01)	2084	6.89e-41	6774.84 (57.47)	1151	6.97e-01	**6852.32** (35.06)	1198	2.99e-07	6952.56
S5-0.7	5465.63 (44.18)	3646	6710.88 (59.25)	3300	1.10e-38	6727.61 (38.38)	1845	2.04e-01	**6799.29** (35.18)	2157	1.04e-08	6981.63
S1-0.8	6496.15 (79.08)	1037	7714.42 (62.42)	929	2.19e-34	7716.22 (58.09)	520	9.09e-01	**7758.31** (51.01)	515	2.82e-03	7965.37
S2-0.8	6316.86 (59.82)	1346	7633.34 (95.96)	1196	6.78e-34	7658.79 (60.05)	648	2.28e-01	**7706.98** (55.65)	648	1.52e-03	7914.28
S3-0.8	6175.59 (68.74)	1744	7582.49 (63.17)	1537	3.07e-37	7595.79 (67.06)	854	4.35e-01	**7662.09** (44.82)	861	4.73e-05	7886.15
S4-0.8	6006.44 (51.14)	2399	7415.22 (55.55)	2176	5.18e-40	7465.19 (61.48)	1176	2.48e-03	**7532.39** (40.48)	1216	1.16e-05	7776.75
S5-0.8	5950.89 (42.38)	3924	7460.19 (63.92)	3597	1.05e-40	7455.19 (78.29)	1921	7.88e-01	**7587.27** (59.61)	2427	1.72e-08	7981.05
S1-0.9	6979.18 (111.95)	1084	8407.76 (91.73)	952	9.21e-32	8440.50 (76.51)	518	1.44e-01	**8474.15** (77.78)	513	5.08e-02	8975.20
S2-0.9	6800.70 (80.72)	1428	8365.19 (77.96)	1257	3.14e-36	8371.13 (66.09)	692	7.52e-01	**8430.32** (60.28)	693	5.30e-04	8945.73
S3-0.9	6675.35 (68.62)	1808	8306.69 (90.73)	1577	1.35e-36	8350.62 (79.32)	860	5.50e-02	**8444.49** (56.90)	871	5.48e-06	8972.89
S4-0.9	6538.40 (60.28)	2557	8236.69 (97.75)	2269	5.40e-37	8254.04 (72.61)	1238	4.41e-01	**8388.12** (52.10)	1309	1.79e-09	8976.69
S5-0.9	6376.67 (40.65)	4132	8113.71 (55.30)	3683	5.63e-44	8073.18 (66.73)	2000	1.57e-02	**8258.97** (46.31)	2772	9.44e-14	8960.20

Results from 30 runs. (Ave: the average quality, σ: the standard deviation)
RANDOM is the method that generates a number of random solutions and chooses the best one.
PGA is a genetic algorithm with the fixed number of evaluation samples.
MGA is a genetic algorithm with the increasing number of evaluation samples.
OPTGA is a genetic algorithm with normalization and increasing number of evaluation samples.
* Average CPU seconds on Intel Xeon CPU 2.4 GHz.
1 One-tailed t-test of the null hypothesis that RANDOM is equal to PGA.
2 Two-tailed t-test of the null hypothesis that PGA is equal to MGA.
3 One-tailed t-test of the null hypothesis that MGA is equal to OPTGA.

a reason for the superiority of OPTHGA, we guess that the proposed normalization is more essential in the local-optimum space of sensor deployments than in the whole solution space and so OPTHGA produced near-optimal sensor deployments on the test instances.

Fig. 4 visualizes the best solutions found by RANDOM, PGA, MGA, OPTGA, VFA, and OPTHGA for the smallest instance S1-0.7, and Fig. 5 visualizes the best solutions for the largest instance S5-0.9. In both figures, we can see that the solution found by OPTHGA is the best and the solution found by RANDOM is the worst among the six methods. Based on these results, we can conclude that MCSDP is efficiently solved by GAs, and the performance can be improved by the proposed normalization method, speed-up technique, and hybridization with local search.

A. Computational Complexity of Each Tested GA

In this subsection, we give the computational complexity of each tested GA. Since every tested GA makes the same number of offspring, for convenience we consider the time complexity needed to generate an offspring. Remind the following notations. Let k be the number of sensor types. Let n_i be the number of sensors for each type $i = 1, 2, \ldots, k$. Let n be the total number of sensors, i.e., $n = \sum_{i=1}^{k} n_i$. Let L be the number of random samples for evaluation.

In PGA and MGA, the time complexity is $O(nL)$, i.e., fitness evaluation dominates the computing time. OPTGA takes $O(nL + \sum_{i=1}^{k} n_i^3)$ time, because it performs the optimal normalization before crossover. In SMCGA, its normalization takes $O(\sum_{i=1}^{k} n_i^2)$ time, so the time complexity of SMCGA is $O(nL + \sum_{i=1}^{k} n_i^2)$. Since VFA takes $O(n^2)$ time as shown in Appendix, the time complexity of OPTHGA becomes $O(nL + n^2 + \sum_{i=1}^{k} n_i^3)$. In view of computational complexity, OPTHGA is slower than other GAs, but for sufficiently large L ($n \ll L$), the difference is negligible, i.e., the time complexity of all the tested GAs becomes $O(nL)$.

V. CONCLUSION

In this paper, we formally defined the maximum coverage sensor deployment problem (MCSDP), which frequently arised in real-world applications. We analyzed the properties of

TABLE V
COMPARISON WITH EXISTING METHODS

	Multi-start VFA		SMCGA		OPTGA		t-test[1]	t-test[2]	OPTHGA		t-test[3]	Upper
Instance	Ave (σ)	Time*	Ave (σ)	Time*	Ave (σ)	Time*	p-value	p-value	Ave (σ)	Time*	p-value	bound
S1-0.7	6438.13 (42.80)	886	6778.86 (20.69)	473	6779.40 (24.93)	473	3.81e-27	4.64e-01	**6813.29** (4.76)	478	1.91e-08	6814.65
S2-0.7	6417.61 (33.94)	1180	6818.03 (35.01)	612	6833.01 (39.33)	613	4.72e-29	6.48e-02	**6881.97** (3.83)	612	7.93e-08	6883.56
S3-0.7	6415.36 (38.85)	1666	6893.49 (38.01)	863	6906.43 (37.05)	880	8.81e-31	9.59e-02	**6982.42** (4.49)	863	1.70e-12	6984.89
S4-0.7	6300.25 (37.91)	2193	6789.50 (47.54)	1156	6852.32 (35.06)	1198	8.57e-33	1.14e-06	**6949.92** (4.64)	1156	7.15e-16	6952.56
S5-0.7	6245.00 (30.43)	3509	6648.37 (61.95)	1869	6799.29 (35.18)	2157	3.39e-34	6.44e-13	**6977.32** (4.15)	2091	3.78e-23	6981.63
S1-0.8	7179.51 (58.80)	990	7757.41 (56.42)	514	7758.31 (51.01)	515	4.04e-28	4.74e-01	**7878.44** (31.70)	510	2.62e-12	7965.37
S2-0.8	7063.08 (44.13)	1257	7706.50 (50.21)	650	7706.98 (55.65)	648	1.15e-30	4.86e-01	**7858.79** (18.95)	644	4.14e-15	7914.28
S3-0.8	6947.92 (44.45)	1656	7623.34 (53.26)	847	7662.09 (44.82)	861	1.59e-33	2.38e-03	**7832.63** (19.10)	850	1.10e-18	7886.15
S4-0.8	6811.24 (40.39)	2275	7461.12 (61.91)	1175	7532.39 (40.48)	1216	6.26e-35	5.32e-06	**7745.07** (10.10)	1184	2.51e-23	7776.75
S5-0.8	6813.00 (34.64)	3719	7340.29 (98.50)	1977	7587.27 (59.61)	2427	1.98e-33	4.71e-13	**7935.62** (11.28)	2291	7.91e-25	7981.05
S1-0.9	7719.05 (57.32)	1031	8459.29 (79.14)	513	8474.15 (77.78)	513	9.28e-29	2.34e-01	**8634.27** (41.91)	515	2.72e-11	8975.20
S2-0.9	7599.49 (48.65)	1359	8413.28 (87.04)	683	8430.32 (60.28)	693	7.79e-33	1.92e-01	**8617.57** (41.65)	678	5.43e-15	8945.73
S3-0.9	7537.25 (45.87)	1706	8379.40 (67.72)	850	8444.49 (56.90)	871	1.00e-34	1.75e-04	**8663.14** (37.48)	852	1.21e-17	8972.89
S4-0.9	7437.67 (37.61)	2409	8270.38 (75.59)	1288	8388.12 (52.10)	1309	5.36e-37	4.15e-08	**8689.45** (22.86)	1239	8.29e-24	8976.69
S5-0.9	7295.69 (24.99)	3981	7860.14 (150.70)	2114	8258.97 (46.31)	2772	9.17e-40	7.08e-15	**8705.76** (20.41)	2538	2.52e-30	8960.20

Results from 30 runs. (Ave: the average quality, σ: the standard deviation)
Multi-start VFA is the method that generates many solutions of VFA ($\alpha_r = 1, \alpha_a = 0.01$) and chooses the best one.
SMCGA is a genetic algorithm with normalization by stable marriage crossover [37] and increasing number of evaluation samples.
OPTHGA is OPTGA hybridized with VFA ($\alpha_r = 1, \alpha_a = 0$) as a local search.
* Average CPU seconds on Intel Xeon CPU 2.4 GHz.
1 One-tailed t-test of the null hypothesis that multi-start VFA is equal to OPTGA.
2 One-tailed t-test of the null hypothesis that SMCGA is equal to OPTGA.
3 One-tailed t-test of the null hypothesis that OPTGA is equal to OPTHGA.

the problem space and tried to find good sensor deployments using novel genetic algorithms.

According to our analysis, the relation between the genotype space and the phenotype space of the MCSDP can be viewed in terms of quotient space. Considering this property, we devised a novel normalization method for the problem that could improve the performance of genetic algorithms. The effectiveness of the proposed normalization method and genetic algorithms were shown by extensive experiments and it can be concluded from these results that the MCSDP is efficiently solved by genetic algorithms with performance improved by our normalization method. The evaluation of sensor deployment was implemented using a Monte Carlo method that reduced time cost by a speed-up technique that started with a small number of random samples that increased with subsequent generations. Moreover the performance could be improved even further with memetic algorithms combined with a well-designed local search.

We applied our genetic algorithms to a sensor deployment problem with static sensors, but it is expected that the proposed methodology will also show good performance with mobile sensors [23], [56], [57]. Further exploration with mobile sensors is left for future study.

APPENDIX
VIRTUAL FORCE ALGORITHM

Zou and Chakrabarty [56], [57] proposed a virtual force algorithm (VFA) as a sensor deployment strategy. VFA enhances the coverage after an initial random placement of sensors. It employs the repulsive and attractive forces so as to enlarge the coverage for the target region. In VFA, every sensor has a force on it influenced by other sensors, obstacles, and preferential areas. However since the original VFA is designed to solve the problem focusing on mobile sensors that is quite different from the MCSDP, it is not easy to apply the original VFA as it is to the MCSDP. So we modified the algorithm to be suitable for the objective of the MCSDP with static sensors. Fig. 6 shows the pseudo-code of our variant of VFA. It takes $O(n^2)$ time, where n is the number of sensors. In our VFA, every sensor has a force on it influenced by other sensors and boundaries. Our VFA has two weight parameters α_r and α_a for the repulsive

VFA($s_1, s_2, \ldots, s_n$)
{
// $\vec{F}_r$: repulsive force, $\vec{F}_a$: attractive force
$\vec{F}_r \leftarrow \vec{0}$, $\vec{F}_a \leftarrow \vec{0}$;
$n_r \leftarrow 0$, $n_a \leftarrow 0$;
for each sensor $s_i = (x_i, y_i)$
for each sensor $s_j (\neq s_i)$
if $d(s_i, s_j) < r_{s_i} + r_{s_j}$ // s_i and s_j are overlapped
$\vec{F}_r \leftarrow \vec{F}_r + \left(1 - \frac{r_{s_i}+r_{s_j}}{d(s_i,s_j)}\right) \cdot (s_j - s_i)$;
$n_r \leftarrow n_r + 1$;
if $d(s_i, s_j) > r_{s_i} + r_{s_j}$ // s_i is separated from s_j
$\vec{F}_a \leftarrow \vec{F}_a + \left(1 - \frac{r_{s_i}+r_{s_j}}{d(s_i,s_j)}\right) \cdot (s_j - s_i)$;
$n_a \leftarrow n_a + 1$;
for each boundary point b of sensor s_i
// $b = (x_i, 0), (x_i, 100), (0, y_i), (100, y_i)$
if $d(s_i, b) < r_{s_i}$ // s_i and boundary are overlapped
$\vec{F}_r \leftarrow \vec{F}_r + \left(1 - \frac{r_{s_i}}{d(s_i,b)}\right) \cdot (s_j - s_i)$;
$n_r \leftarrow n_r + 1$;
if $d(s_i, b) > r_{s_i}$ // s_i is separated from boundary
$\vec{F}_a \leftarrow \vec{F}_a + \left(1 - \frac{r_{s_i}}{d(s_i,b)}\right) \cdot (s_j - s_i)$;
$n_a \leftarrow n_a + 1$;

$s_i \leftarrow s_i + \alpha_r \cdot \frac{\vec{F}_r}{n_r} + \alpha_a \cdot \frac{\vec{F}_a}{n_a}$;
}
// r_{s_i} is the detection radius of sensor s_i.
// $d(s_i, s_j) := \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
// α_r and α_a are constant weight parameters.

Fig. 6. Our variant of virtual force algorithm (VFA).

and attractive forces, respectively. In our preliminary test, the VFA as a stand-alone heuristic showed the best performance when $\alpha_r = 1, \alpha_a = 0.01$. When combined with the proposed genetic algorithms, the VFA of $\alpha_r = 1, \alpha_a = 0$ performed the best. So as OPTHGA, we used OPTGA combined with the VFA without any attractive force.

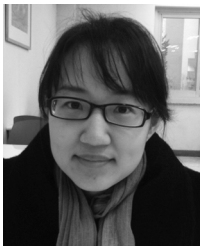
ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their constructive comments based on which the presentation of this paper has greatly improved.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.
[2] D. Avis, "A survey of heuristics for the weighted matching problem," *Networks*, vol. 13, no. 4, pp. 475–493, 1983.
[3] N. Bartolini, T. Calamoneri, E. G. Fusco, A. Massini, and S. Silvestri," "Push & pull: Autonomous deployment of mobile sensors for a complete coverage," *Wireless Netw.*, vol. 16, no. 3, pp. 607–625, 2010.
[4] H. J. Bremermann, J. Rogson, and S. Salaff, "Global properties of evolution processes," in *Natural Automata and Useful Simulations*, H. H. Pattee, 2nd ed. Spartan Books, 1966, pp. 3–42.
[5] D. Burago, Y. Burago, S. Ivanov, and Iu. D. Burago, *A Course in Metric Geometry*. American Mathematical Society, USA, 2001.
[6] M. Cardei, M. Thai, Y. Li, and W. Wu, "Energy efficient target coverage in wireless sensor networks," in *Proc. IEEE Comput. Commun. Soc.*, Mar. 2005, pp. 1976–1984.
[7] M. Cardei and J. Wu, "Coverage in wireless sensor networks," in *Handbook of Sensor Networks*, M. Ilyas and I. Mahgoub, Eds. Boca Raton, FL, USA: CRC Press, 2004.
[8] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1448–1453, Dec. 2002.
[9] C.-Y. Chang, C.-T. Chang, Y.-C. Chen, and H.-R. Chang, "Obstacle-resistant deployment algorithms for wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 58, no. 6, pp. 2925–2941, 2009.
[10] R.-S. Chang and S.-H. Wang, "Self-deployment by density control in sensor networks," *IEEE Trans. Veh. Technol.*, vol. 57, no. 3, pp. 1745–1755, 2008.
[11] S. Y. Chen and Y. F. Li, "Automatic sensor placement for model-based robot vision," *IEEE Trans. Sys., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 393–408, 2004.
[12] S.-S. Choi and B.-R. Moon, "Normalization in genetic algorithms," in *Proc. Gen. Evolut. Comput. Conf.* pp. 862–873, 2003.
[13] S.-S. Choi and B.-R. Moon, "Normalization for genetic algorithms with nonsynonymously redundant encodings," *IEEE Trans. Evol. Comput.*, vol. 12, no. 5, pp. 604–616, 2008.
[14] T. Clouqueur, V. Phipatanasuphom, P. Ramanathan, and K. K. Saluja, "Sensor deployment strategy for target detection," in *Proc. Int. Workshop Wireless Sens. Netw. Application*, Sep. 2002, pp. 42–48.
[15] S. S. Dhillon, K. Chakrabarty, and S. S. Iyengar, "Sensor placement for grid coverage under imprecise detections," in *Proc. Int. Conf. Information Fusion*, 2002, pp. 1581–1587.
[16] R. Dorne and J. K. Hao, "A new genetic local search algorithm for graph coloring," in *Proc. 5th Conf. Parallel Problem Solving Nature*, 1998, pp. 745–754.
[17] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Proc. 2nd Workshop Found. Genet. Algorithms*, 1993, pp. 187–202.
[18] S. Ferrari, G. Zhang, and T. A. Wettergren, "Probabilistic track coverage in cooperative sensor networks," *IEEE Trans. Syst., Man, Cybern. B Cybern.*, vol. 40, no. 6, pp. 1492–1504, 2010.
[19] J. B. Fraleigh, *A First Course in Abstract Algebra*, 7th ed. Reading, MA, USA: Addison-Wesley, 2002.
[20] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco, CA, USA: Freeman, 1979.
[21] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.
[22] Z. Guo, M. Zhou, and G. Jiang, "Adaptive sensor placement and boundary estimation for monitoring mass objects," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 1, pp. 222–232, 2008.
[23] A. Howard, M. J. Matarič, and G. S. Sukhate, "An incremental self-deployment algorithm for mobile sensor networks," *Autonomous Robot.*, vol. 13, no. 2, pp. 113–126, Sep. 2002.
[24] A. Howard, M. J. Matarič, and G. S. Sukhate, "Mobile sensor network deployment using potential field: A distributed, scalable solution to the area coverage problem," in *Proc. 6th Int. Symp. Distributed Autonomous Robot. Syst.*, Jun. 2002, pp. 299–308.
[25] D. B. Jourdan and O. L. de Weck, "Layout optimization for a wireless sensor network using a multi-objective genetic algorithm," in *Proc. 59th IEEE Veh. Technol. Conf.*, vol. 5. 2004, pp. 2466–2470.
[26] S.-J. Kang and B.-R. Moon, "A hybrid genetic algorithm for multiway graph partitioning," in *Proc. Genet. Evol. Comput. Conf.*, 200, pp. 159–1660.
[27] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logistic Quart.*, vol. 2, pp. 83–97, 1955.
[28] L. M. J. Lamm, M.S. thesis, Division Syst. Eng., School Eng. Appl. Sci., Univ. Virginia, May 2001.
[29] G. Laszewski, "Intelligent structural operators for the *k*-way graph partitioning problem," in *Proc. 4th Int. Conf. Genet. Algorithms*, 1991, pp. 45–52.
[30] M. Locateli and U. Raber, "Packing equal circles in a square: A deterministic global optimization approach," *Discrete Appl. Math.*, vol. 122, pp. 139–166, Oct. 2002.
[31] Y. Ma, D. Yau, N. Yip, N. Rao, and J. Chen, Stochastic steepest-descent optimization of multiple-objective mobile sensor coverage," *IEEE Trans. Veh. Technol.*, vol. 61, no. 4, pp. 1810-1822, May 2012.
[32] A. Moraglio, Y.-H. Kim, Y. Yoon, and B.-R. Moon, "Geometric crossover for multiway graph partitioning," *Evol. Comput.*, vol. 15, no. 4, pp. 445–474, 2007.

[33] M. Moser, D. P. Jokanovic, and N. Shiratori, "An algorithm for the multidimensional multiple-choice knapsack problem," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E80-A, no. 3, pp. 582–589, 1997.
[34] H. Mühlenbein, "Parallel genetic algorithms in combinatorial optimization," in *Computer Science and Operations Research: New Developments in Their Interfaces*. Oxford, U.K.: Pergamon Press, 1992, pp. 441–456.
[35] K. Mukherjee, S. Gupta, A. Ray, and T. A. Wettergren, "Statistical-mechanics-inspired optimization of sensor field configuration for detection of mobile targets," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 3, pp. 783–791, Jun. 2011.
[36] D. Puccinelli and M. Haenggi, "Wireless sensor networks: Applications and challenges of ubiquitous sensing," *IEEE Circuits Syst. Mag.*, vol. 5, no. 3, pp. 19–31, 2005.
[37] J.-H. Seo, Y.-H. Kim, H.-B. Ryou, S.-H Cha, and M. Jo, "Optimal sensor deployment for wireless surveillance sensor networks by a hybrid steady-state genetic algorithm," *IEICE Trans. Commun.*, vol. E91-B, no. 11, pp. 3534–3543, 2008.
[38] C. Sharp, S. Schaffert, A. Woo, N. Sastry, C. Karlof, S. Sastry, and D. Culler, "Design and implementation of a sensor network system for vehicle tracking and autonomous interception," in *Proc. Eur. Workshop Sensor Netw.*, 2005, pp. 93–107.
[39] Z. Shen, Y. Chang, H. Jiang, Y. Wang, and Z. Yan, "A generic framework for optimal mobile sensor redeployment," *IEEE Trans. Veh. Technol.*, vol. 59, no. 8, pp. 4043–4057, Oct. 2010.
[40] N. Srour, "Unattended ground sensors a prospective for operational needs and requirements," Tech. Rep., Sensors Electron Devices Directorate, U.S. Army Research Lab., Oct. 1999.
[41] Y.-R. Tsai, "Sensing coverage for randomly distributed wireless sensor networks in shadowed environments," *IEEE Trans. Veh. Technol.*, vol. 57, no. 1, pp. 556–564, Jan. 2008.
[42] S. Tsutsui and D. E. Goldberg, "Search space boundary extension method in real-coded genetic algorithms," *Inform. Sci.*, vol. 133, nos. 3–4, pp. 229–247, 2001.
[43] C. van Hoyweghen, B. Naudts, and D. E. Goldberg, "Spin-flip symmetry and synchronization," *Evol. Comput.*, vol. 10, no. 4, pp. 317–344, Dec., 2002.
[44] P.-J. Wan and C.-W. Yi, "Coverage by randomly deployed wireless sensor networks," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2658–2669, Jun. 2006.
[45] B. Wang, "Coverage problems in sensor networks: A survey," *ACM Comput. Surveys*, vol. 43, no. 4, pp. 32:1–32:53, 2011.
[46] G. Wang, G. Cao, and T. F. La Porta, "A bidding protocol for deploying mobile sensors," in *Proc. 11th IEEE Int. Conf. Netw. Protocols*, Nov. 2003, pp. 315–324.
[47] G. Wang, G. Cao, T.L. Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *Proc. IEEE Infocom Conf.*, Mar. 2005, pp. 2302–2312.
[48] Q. Wu, N. S. V. Rao, X. Du, S. S. Iyengar, and V. K. Vaishnavi, "On efficient deployment of sensors on planar grid," *Comput. Commun.*, vol. 30, nos. 14–15, pp. 2721–2734, 2007.
[49] X. Xu and S. Sahni, "Approximation algorithms for wireless sensor deployment," *IEEE Trans. Comput.*, vol. 56, no. 12, pp. 1681–1695, Dec. 2007.
[50] Y. Xu and X. Yao, "A GA approach to the optimal placement of sensors in wireless sensor networks with obstacles and preferences," in *Proc. 3rd IEEE Consumer Commun. Networking Conf.*, 2006, pp. 127–131.
[51] Y. Yao, C.-H. Chen, B. Abidi, D. Page, A. Koschan, and M. Abidi, "Can you see me now? Sensor positioning for automated and persistent surveillance," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 1, pp. 101–115, 2010.
[52] Y. Yoon, Y.-H. Kim, A. Moraglio, and B.-R. Moon, "Quotient geometric crossovers and redundant encodings," *Theor. Comput. Sci.*, vol. 425, pp. 4–16, 2012.
[53] Y. Yoon, Y.-H. Kim, A. Moraglio, and B.-R. Moon, "A theoretical and empirical study on unbiased boundary-extended crossover for real-valued representation," *Inform. Sci.*, vol. 183, no. 1, pp. 48–65, 2012.
[54] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc Sensor Wireless Netw.*, vol. 1, no. 1, pp. 89–124, 2005.
[55] C. Zhao, Z. Yu, and P. Chen, "Optimal deployment of nodes based on genetic algorithm in heterogeneous sensor networks," in *Proc. IEEE Int. Conf. Wireless Commun. Networking Mobile Comput.*, Sep. 2007, pp. 2743–2746.
[56] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in *Proc. IEEE Infocom Conf.*, Mar. 2003, pp. 1293–1303.
[57] Y. Zou and K. Chakrabarty, Sensor deployment and target localization in distributed sensor networks, *ACM Trans. Embedded Comput. Syst.*, vol. 3, no. 1, pp. 61–91, 2004.



Yourim Yoon received the B.E. degree in computer engineering and the Ph.D. degree in computer science and engineering from Seoul National University, Seoul, Korea, in 2003 and 2012, respectively.

Since March 2012, she has been a Senior Research Engineer at Future IT Research and Development Laboratory, LG Electronics, Seoul, Korea. Her current research interests include optimization theory, machine learning, combinatorial optimization, evolutionary computation, discrete mathematics, operations research, smart grids, and sensor networks.

Dr. Yoon served as a reviewer for BIC-TA 2007, BMIC 2011, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, and *KSII Transactions on Internet and Information Systems*.

Yong-Hyuk Kim (M'13) received the B.S. degree in computer science, and the M.S. and Ph.D. degrees in computer science and engineering from Seoul National University (SNU), Seoul, Korea, in 1999, 2001, and 2005, respectively.

From March 2005 to February 2007, he was a Post-Doctoral Scholar at SNU and also a Research Staff Member at the Inter-University Semiconductor Research Center, SNU, Seoul. Since March 2007, he has been a Professor at the Department of Computer Science and Engineering, Kwangwoon University, Seoul, Korea. His current research interests include algorithm design/analysis, discrete mathematics, optimization theory, combinatorial optimization, evolutionary computation, operations research, data/web mining, and sensor networks.

Dr. Kim has served as an editor of the TIIS journal from 2010 to 2013, a Committee Member of GECCO from 2005 to 2006, in 2013, IEEE CEC from 2009 to 2012, and has been a reviewer for journals (CIM, IS, TC, TEC, TKDE, TPDS, TSE) of the IEEE since 2003. He is a member of the IEEE SMC Society and the IEEE Communications Society, and is the corresponding author of this paper.