An Effective Genetic Algorithm to Improve Wireless Sensor Network Lifetime for Large-Scale Surveillance Applications

Chih-Chung Lai, *Student Member, IEEE*, Chuan-Kang Ting, *Member, IEEE*, and Ren-Song Ko, *Member, IEEE*

***Abstract*— Wireless sensor network lifetime for large-scale surveillance systems is defined as the time span that all targets can be covered. One approach to extend the lifetime is to divide the deployed sensors into disjoint subsets of sensors, or sensor covers, such that each sensor cover can cover all targets and work by turns. The more sensor covers can be found, the longer sensor network lifetime can be prolonged. Finding the maximum number of sensor covers can be solved via transformation to the Disjoint Set Covers (DSC) problem, which has been proved to be NP-complete. For this optimization problem, existing heuristic algorithms either get unsatisfactory solutions in some cases or take exponential time complexity. This paper proposes a genetic algorithm to solve the DSC problem. The simulation results show that the proposed algorithm can get near-optimal solutions with polynomial computation time and can improve the performance of the most constrained-minimum constraining heuristic algorithm by 16% in solution quality.**

I. Introduction

With recent advances in hardware miniaturization, communication technologies, and low-cost mass production, large-scale networks with hundreds or even thousands of small, inexpensive, battery-powered, and wirelessly connected sensors have become possible and bring up a wide range of new applications. Some applications are surveillance, biological detection, home security, smart spaces, and environmental monitoring [1], [2], [3], [4]. These sensors can collect environmental information within their sensing ranges and have capability for further data processing. They can also transmit, relay, and receive information within their communication ranges.

For many promising applications, sensors may be ad-hoc deployed in distant locations or hostile environments such as battlefields. In the military surveillance applications, for instance, we can scatter sensors by an aircraft into the critical area for detecting intrusions, traps, or land mines. However, it is uneconomical or infeasible to replace battery when sensors run out of energy. One important design consideration in sensor networks is therefore to efficiently utilize the limited energy.

Though there is a significant amount of literature addressing the issue of efficient energy management in generic wireless ad-hoc networks, sensor networks face new constraints about sensing coverage introduced by their distributed sensing applications. For example, surveillance applications may require each location in a geographic region of interest to be covered by at least one sensor, while object tracking applications may require at least three sensors and data sampling applications may require a given percentage of monitored region to be covered. Therefore, an alternative technique commonly adopted in sensor networks is to schedule some sensors to enter the *power-saving mode*, or *sleeping mode*, while the remaining sensors can still satisfy the coverage constraints [5], [6]. Several research results [5], [7] illustrate that such a technique reduces energy consumption significantly and can prolong the sensor network lifetime. Note that the sensor network lifetime is defined here as the time span for the coverage constraints being satisfied.

One approach based on the sensor activity scheduling technique is to divide all sensors into disjoint sensor subsets, or *sensor covers*, and each sensor cover needs to satisfy the coverage constraints. Only one sensor cover is active to provide the functionality and the remaining sensor covers are in the sleeping mode. Once the active sensor cover runs out of energy and consequently cannot maintain coverage constraints, another sensor cover will be selected to enter the active mode and provide the functionality continuously. The more sensor covers we can find, the longer sensor network lifetime will be prolonged.

In this paper, we consider a discrete sensor surveillance application in which a set of discrete targets have to be fully covered by sensors. Once a target can no longer be covered, the system will become undependable. For example, on the battlefield, each location of interest must be covered by at least one sensor; otherwise there may be some critical information lost, which might lead to misjudgment of the military situation and even severe casualties. Thus, the objective of the scheduling problem considered here is to find maximum number of disjoint sensor covers such that each sensor cover can fully cover all these targets.

In the literature, finding the maximum disjoint sensor covers has been solved via transformation into the Disjoint Set Covers (DSC) problem [8], or, equivalently, the SET K-COVER problem [9]. Both are proved to be NP-complete [8], [9]. Several heuristic methods have been proposed to solve these problems. The most constrained-minimum constraining heuristic (MCMCC) [9] takes only polynomial time but it gets improvable results, e.g. 80% of the optimumal number of sensor covers, in some cases. On the other hand, the Maximum Covers using Mixed Integer Programming (MC-MIP) [8] heuristic algorithm takes implicit exhaustive search scheme and guarantees finding the optimum solutions. However, MC-MIP is impractical in large-scale applications due to its exponential time complexity. Therefore, there is a need for an algorithm that can stably deliver more disjoint sensor covers and terminate in acceptable and predictable running

The authers are with Department of Computer Science and Information Engineering, National Chung Cheng University, Chia-Yi, Taiwan 621. (e-mail: {laicc, ckting, korenson}@cs.ccu.edu.tw).

1-4244-1340-0/07$25.00

time for large-scale applications.

Genetic algorithms (GAs) have efficiently solved a great variety of optimization problems. The basic idea of GAs is to encode candidate solutions of a problem into genotypes, and candidate solutions could be improved through the simulation of evolution mechanisms in natural, such as selection, crossover, and mutation, on these genotypes [10]. To the best of our knowledge, there is no research using an evolutionary algorithm to solve the DSC problem so far. In this paper, we propose *Genetic Algorithm for Maximum Disjoint Set Covers* (GAMDSC) to solve the DSC problem. The simulation results show that GAMDSC can get near-optimal solutions and improve the performance of MCMCC by 16% in terms of the obtained sensor covers with acceptable computation time.

The rest of this paper is organized as follows. Section 2 briefly reviews existing models and algorithms for prolonging sensor network lifetime. In section 3, we explain the DSC problem and the two heuristic algorithms, MCMCC and MC-MIP. The issues of the two heuristic algorithms are discussed. Section 4 describes the components of the proposed GAMDCS in detail and give a worst-case running time analysis. Our simulation results are given and discussed in section 5. Finally, we conclude this paper and give the future work.

II. Related Work

Extending sensor network life can be undertaken in many aspects such as routing protocol, clustering and data aggregation, sensor activity scheduling, and data storage and dissemination [2]. This section mainly introduces the work related to sensor activity scheduling and applications of evolutionary techniques on sensor networks.

Slijepcevic and Potkonjak [9] study the problem of organizing sensors to achieve full coverage of a monitored area as long as possible. They propose a polynomial time algorithm that can reduce area coverage problem to discrete target coverage problem and formulate the SET K-COVER problem for extending the sensor network lifetime. It is claimed that NP-completeness of SET K-COVER problem can be proved by the reduction from the minimum cover problem [11]. The most constrained-least constraining heuristic is developed for solving the SET K-COVER problem.

Abrams et al. [12] consider a variation of the SET K-COVER problem. The coverage constraint is relaxed the every cover can only cover partial targets, and the goal is maximizing the number of times the areas are covered by the partition of sensors. Three algorithms are proposed to solve this variation. The first is randomized with an expected fraction $1 - \frac{1}{e}$ of the optimum. The second is a distributed greedy algorithm with a $\frac{1}{2}$-approximation ratio. The third is a centralized greedy algorithm, which is a derandomized version of the first algorithm, and has a $(1-\frac{1}{e})$-approximation ratio.

Similarly, Cardei and Du [8] consider the problem of energy efficiency in wireless sensor applications for surveillance of a set of discrete targets with known locations and also model it as the SET K-COVER problem or the DSC problem—the name used in their paper. It is proved that the DSC problem is NP-complete and if NP$\neq$P, the DSC problem has no polynomial-time approximation algorithm with approximation factor less than 2. Moreover, a heuristic algorithm MC-MIP with a mixed integer programming formulation is proposed to solve the problem.

Berman et al. [13] assume the knowledge of initial battery power and energy consumption rate for each sensor. The disjoint constraint is relaxed so each sensor can participate in different sensor covers as long as it still has energy. They formulate the sensor network lifetime problem (SNLP), which designates finding a monitoring schedule with the maximum length of time. A $(1+ln(1-q)^{-1})$-approximation algorithm is proposed for the case when a q-portion of the monitored area is required to be covered, e.g., for the 95% area coverage, their schedule guarantees to be at most 2.99 times shorter than the optimum.

A technique to reduce the data traffic is clustering in combination with data aggregation. Active sensors are partitioned into small clusters and each cluster elects a sensor as the *cluster head*. The cluster head aggregates raw data from member nodes in the cluster, and reports a requested statistical data to the base station. Qin and Zimmermann [14] propose a genetic algorithm to find more energy-efficient clustering based on a linear programming model which tries to maximize the period that at least K sensors can work at a time.

Ferentinos and Tsiligiridis [15] propose a multi-objective optimization genetic algorithm to design cluster-based sensor networks. A linear weighted function of multiple parameters is used as fitness to balance the trade-off between the parameters. These parameters are related to application requirements, network, and energy consumption.

III. Disjoint Set Covers Problem and Existing Heuristics

A. Problem definition

Assume that n sensors $s_1, s_2, ..., s_n$ are deployed in territory to monitor m targets $t_1, t_2, ..., t_m$. A target is said to be *covered* by a sensor if it lies within the sensing region of the sensor. We can prolong the sensor network lifetime by finding the maximum number of disjoint sensor covers. The problem can be solved via transformation to the DSC problem, which is defined as follows:

Definition *(Disjoint Set Covers Problem)* [8] Given a collection S of subsets of a finite set T, find the maximum number of disjoint covers for T. Every cover C_i is a subset of S, $C_i \subseteq S$, such that every element of T belongs to at least one member of C_i, and for any two covers C_i and C_j , $C_i \cap C_j = \phi$.

That is, let T be the set of the targets, $t_1, t_2, ..., t_m$, sensor s_i can be represented by a subset, denoted as S_i, of T, where $t_j \in S_i$ if and only if t_j lies within the sensing region of sensor s_i. Thus, S is a collection of subsets representing sensors and a cover C_i represents a sensor cover.

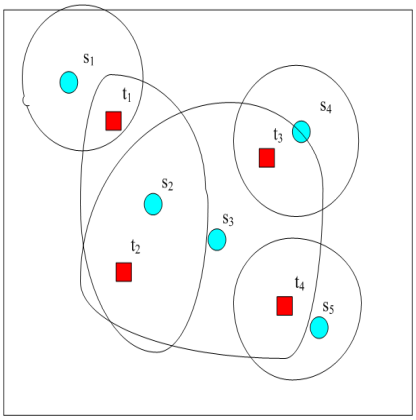

Fig. 1. A deployment of sensor networks. Note that in real-world applications, sensing region of a sensor can be an irregular shape [16].

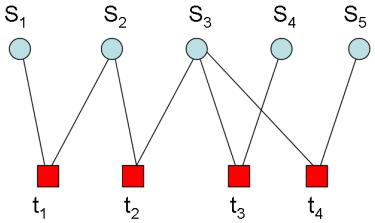

Fig. 2. An instance of the DSC problem. Link between S_i and t_i means t_i lies within the sensing region of s_i.

As depicted in Fig. 1, s_1, s_2, s_3, s_4, and s_5 are five sensors, and t_1, t_2, t_3, and t_4 are four targets. It is obvious that each sensor network may be represented by a bipartite graph $G(V, E)$, where $V = S \cup T$ and $e_{ij} \in E$ if s_i covers t_j. Figure 2 depicts the bipartie graph for Fig. 1. In this example, $S_1 = \{t_1\}$, $S_2 = \{t_1, t_2\}$, $S_3 = \{t_2, t_3, t_4\}$, $S_4 = \{t_3\}$, and $S_5 = \{t_4\}$. We can find two disjoint covers, $C_1 = \{S_1, S_3\}$ and $C_2 = \{S_2, S_4, S_5\}$. Note that, in this example, optimum number of disjoint covers is 2.

B. Existing heuristics for the DSC problem

MCMCC finds disjoint covers in an iterative way, i.e., deciding one cover per round. Initially, S' is set as S. At the beginning of each round, the cover to be determined C_i is set as ϕ, the available sensors set V is S', and the set of targets uncovered by C_i, denoted as U, is T. Then, the most sparsely covered target $t_{min} \in U$ is identified and the score of each $S_i \in V$ such that $t_{min} \in S_i$ is decided by the following scoring rules:

For each target $t_i \in S_i$ that is not t_{min},

1) $+M - K$ if $t_i \in U$. Here M is an user-determined parameter and K is the number of $S_j \in V$ such that $t_i \in S_j$.
2) $-N + L$ if $t_i \notin U$. Here N is an user-determined parameter and L is the number of $S_j \in S' - C_i$ such that $t_i \in S_j$.

For each S_i, $V = V - \{S_i\}$ after the score of S_i is determined. After calculating all scores, the S_i with the highest score will be selected and added into C_i and the all targets covered by the selected S_i will be removed from U. If two S_i's have the same score, arbitrary one can be selected. This procedure for identifying t_{min} and choosing a sensor into C_i will iteratively proceed until $U = \phi$. Hence, one cover C_i is found. $S' = S' - C_i$ and the current round terminates. If the sensors in S' can cover all targets, a new round will begin for the next cover. That is, new round will continuously launch until the sensors in S' can not cover all the targets, and then the algorithm outputs all covers found and terminates. This heuristic needs $O(N^2)$ computation time, where N is the number of sensors.

MCMCC does not guarantee to find the maximum disjoint covers. For example, consider the DSC problem instance depicted in Fig. 2. We apply MCMCC on it with parameters $M = 10$ and $N = 8$. In the first round, t_1 may be identified as the t_{min}. There are two candidate sensors, S_1 and S_2, that could be selected. According to the scoring rule 1, t_1 contributes no score since it is the ct. t_2 contributes $10 - 2 = 8$ to S_2. In total, the score of S_2 is equal to $0 + 8 = 8$, which is higher than the score of S_1, 0. So S_2 will be added to the current cover. Then, t_4 may be identified as the next t_{min}. S_3 and S_5 are both candidates. According to the scoring rule 2, t_2 contributes $-8 + 1 = -7$ to S_3. According to the scoring rule 1, t_3 contributes $10 - 2 = 8$ to S_3. t_4 contributes no score. In total, S_3 gets $-7 + 8 = 1$, which is higher than the score of S_5, 0. Thus, S_3 will be selected into the current cover and one cover $C_1 = \{S_2, S_3\}$ is found. Since S_1, S_4, and S_5 can not cover all the targets, the algorithm output the only found cover C_1 and terminates. MCMCC fails to find maximum number of covers of this problem instance of DSC.

Another heuristic algorithm MC-MIP adopts the implicit exhaustive search scheme, which guarantees the optimum solution but needs exponential running time in the worst case. MC-MIP transforms a DSC problem instance into a constrained maximum flow network. It has been proved that the DSC problem instance has the maximum number, c^*, of disjoint covers if and only if the corresponding flow network has the maximum flow $c^* m$ [8]. MC-MIP formulates the flow network as mix integer programming and solves it by branch-and-bound heuristic. Hence, the maximum disjoint covers can be derived from the results of the maximum flow network. Although MC-MIP can guarantee the optimum solution, the problem scale will be extended after transformation and thus introduces longer computation time, which is impractical in large-scale applications.

IV. Genetic Algorithm for Disjoint Set Cover Problem

This paper proposes Genetic Algorithm for Maximum Disjoint Set Covers (GAMDSC). In GAMDSC, the key of design is the chromosome representation. Besides, an additional operator, called *scattering*, is applied on the offspring

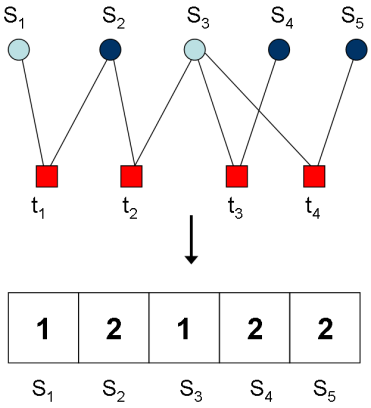

Fig. 3. Encoding a grouping combination. S_1 and S_3 join the group 1, and S_2, S_4, and S_5 join the group 2. Since either group 1 or group 2 can fully cover the targets, both groups form covers. Thus, the fitness of the chromosome is 2.

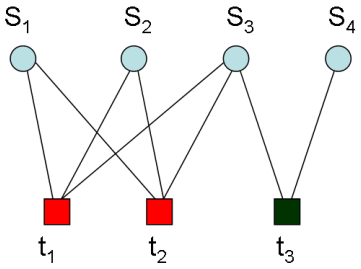

Fig. 4. Upper bound of number of covers. t_3 is the most sparsely covered target since it is only covered by 2 sensors while t_1 and t_2 are covered by 3 sensors. Thus, we can find at most 2 covers and $ub = 2$.

after mutation. Scattering does not involve any fitness evaluation. In this section, we only describe the GA components depending on the representation, namely, encoding scheme, fitness function, variation operators, and scattering. Finally, the worst-case time complexity analysis of GAMDSC is given.

A. Representation

An intuitive way to find disjoint covers is that each sensor randomly joins a group among prescribed groups. A group forms a cover if it can cover all targets. Based on this idea, we use integer representation to encode a grouping combination of sensors. The value of a gene indicates the index of the group that the sensor joins. Figure. 3 illustrates that S_1 joins group 1, S_2 joins group 2. The value of each gene is an integer ranging from 1 to the number of the prescribed groups. The number of the prescribed groups needs to be close to and greater than the optimum number of covers. Note that if a target is only covered by k sensors, it is impossible to find more than k disjoint covers. Thus, the number of the prescribed groups can be set as an upper bound of the number of covers, ub, which is decided by the number of sensors covering the most sparsely covered target. Figure 4 illustrates how to decide ub. Target t_3 is the most sparsely covered target since it is covered by 2 sensors and the other two targets are covered by 3 sensors. In this case, we will find at most 2 covers and thus $ub = 2$. Furthermore, since there is no sensor joining two groups simultaneously due to the one-to-one mapping of genes and sensors, the disjoint constraint is always satisfied, which implies that the whole genotypic space corresponds to feasible solutions.

B. Fitness

The fitness of a chromosome is defined as the number of disjoint covers that can be found by the grouping combination represented by the chromosome. Since the whole genotypic space corresponds to feasible solutions, we only need to calculate how many groups can form covers. A list of covered targets for each sensor is maintained. To check if a group of sensors can form a cover, we go through the list of each sensor of the group once and check if all targets are involved. Consider the chromosome depicted in Fig. 3. Group 1 and 2 can cover all targets. Thus, the chromosome has fitness value 2. For the input DSC problem instance, the fitness evaluation needs $O(|S|\times|T|)$ computation complexity in worst case, where $|S|$ is number of sensors and $|T|$ is the number of targets.

C. Variation Operators

The variation operators for integer-based GA are applicable. In our simulation, we adopt uniform crossover and creep mutation. Uniform crossover exchanges each gene of the two parents with probability $1/2$ to generate two offspring. Creep mutation changes each gene of the offspring to a random number varying from 1 to ub with a given mutation rate.

D. Scattering Operator

Let us refer to the sensors covering the most sparsely covered target as the *critical sensors*. The scattering operator is to keep all the critical sensors joining different groups. Consider the instance shown in Fig. 4. S_3 and S_4 are the critical sensors. If they join the same group, then at most only one cover can be found since no more sensor in other groups can cover t_3. This implies that it is better to prevent multiple critical sensors from joining the same groups. In other words, the critical sensors need to be "scattered" to different groups. Scattering is applied on the offspring soon after mutation. As execution, the operator first check the genes corresponding to the critical sensors and find out the genes with repeated values. Then it replaces the values of the repeated genes with other values from 1 to ub such that all genes corresponding to the critical sensors have different values. This can be done because there are at most ub critical sensors. With simple data structures, e.g., arrays, scattering on an individual can finish in $O(|S|)$ time, which is asymptotically smaller than that of fitness evaluation.

TABLE I

SUMMARY OF THE OPERATORS AND PARAMETERS OF GAMDSC.

Type	Generational GA
Representation	$\{1, 2, ..., ub\}^{\lvert S\rvert}$
Parent Selection	Fitness Proportionate Selection
Crossover	Uniform with crossover rate 1.00
Mutation	Creep with mutation rate $1/\lvert S\rvert$
Heuristic	Scattering
Survivor Selection	$\mu + \lambda$
Population Size	100
Stop Condition	200 generations
Runs	100

E. Worst-Case Time Complexity of GAMDSC

For each generation, computation time bottleneck lies on the fitness evaluation. Thus, in the worst case, time complexity of a generation is $O((\mu+\lambda) \times |S| \times |T|)$, where μ is the population size, λ is the number of offspring, $|S|$ is the number of sensors, and $|T|$ is the number of targets. Hence, the worst case time complexity of GAMDSC will be $O(\tau \times (\mu+\lambda) \times |S| \times |T|)$, where τ is the maximum number of generations. It will be equal to $O(\tau \times \mu \times |S| \times |T|)$ if $\mu = \lambda$. Note that the worst-case running time of MCMCC is $O(|S|^2)$, and the worst-case running time of MC-MIP is exponential. Thus, the time complexity of GAMDSC lies between MCMCC and MC-MCIP.

V. SIMULATION RESULTS AND DISCUSSION

We conduct a series of simulations to compare MC-MIP, MCMCC, and the proposed GAMDSC. The simulation platform is JDK 5.0 on top of Windows XP/Pentium 4 2.8 GHz. Each simulation case includes 100 runs of GAMDSC. Table I summarizes the operators and parameters used in our simulations. Note that a sensor network is usually deployed at random due to the huge scale and sometimes the hostile environment. Thus, it is difficult to control the exact topology formed by sensors and targets. Instead, it is possible to control the upper bound of the number of covers, ub. There are two approaches usable. The first approach is adjusting the sensing radius of a sensor. Although the sensing region of a sensor can be irregular, a statistical average radius representing the size of sensing region is available. For the same deployment of sensors and targets, large sensing region of each sensor can increase ub, but it also increases energy consumption. Thus, there should be a trade-off between the size of sensing region and the sensor network lifetime. However, the topic is out of the scope of this paper since what we concern are the effectiveness and efficiency of finding sensor covers with the same given topology. The second approach is controlling the amount of the deployed sensors. Statistically, more deployed sensors could get higher ub.

A. Simulation 1 : adjusting the sensing radius

In the first simulation, 90 sensors and 10 targets are randomly deployed in a 500×500 region. For a given distribution of sensors and targets, we increase sensing radius from 100 to 300 by 20 for observing the performance with different values of ub. Here we use uniform disk as the model of sensing region to generate test cases. Table II and Fig. 5 show the comparison of number of covers found by MC-MIP, MCMCC, GAMDSC in each case. Note that, as mentioned above, MC-MIP will always result in the optimal solutions. When $ub < 11$, both GAMDSC and MCMCC can get optimal number of covers. When ub grows, the MCMCC performance deteriorates, while GAMDSC keeps getting near-optimal number of covers. In the worst case which $ub = 14$, GAMDSC still finds more than 92% of optimal number of covers on average. Moreover, we perform one-tailed t-test on the numbers of covers obtained by GAMDSC and MCMCC. With confidence level $\alpha = 0.05$, the improvements of GAMDCS on MCMCC are significant in most cases. In the case of $ub = 36$, GAMDSC can even improve by 16% the number of covers obtained from MCMCC. Note that Fig. 5 also shows that the difference between the number of covers found by GAMDSC and MCMCC grows as ub grows, which may imply GAMDSC may perform better with high ub for the same spatial distribution. Besides, the column MAX. of Table II lists the maximum covers obtained by GAMDSC. It represents that GAMDSC probably find the optimumal solutions. In practical use, we can run GAMDSC more than once to get the optimal solution.

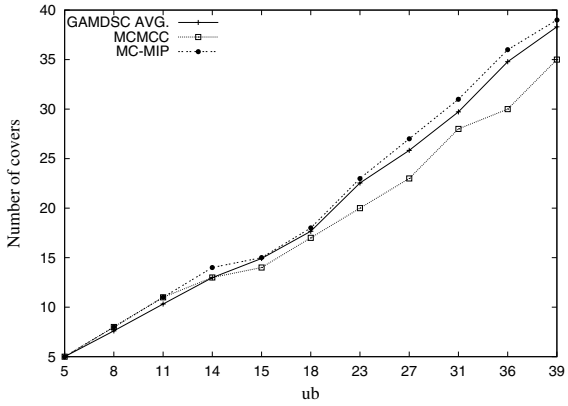

Fig. 5. Comparison of the number of covers obtained by MC-MIP, MCMCC, GAMDSC in simulation 1. As ub increases, the MCMCC performance deteriorates, while GAMDSC keeps getting near-optimumal number of covers.

Figure 6 shows the running time comparison. The results are depicted in logarithmic scale (Fig. 6(a)) and normal scale (Fig. 6(b)). As expected, MCMCC requires the least time. The logarithmic value of MC-MIP running time increases rapidly as shown in Fig. 6(a), which means the running time of MC-MIP grows exponentially. In the case with highest $ub = 39$, the running time of MC-MIP surpasses 10 hours. On the other hand, Fig. 6(b) shows that running time of GAMDSC grows linearly. It coheres with the result of time complexity analysis since for each fitness evaluation, $|S|$ should increase proportionally to ub.

TABLE II

COMPARISON OF THE NUMBER OF COVERS OBTAINED BY MC-MIP, MCMCC, AND GAMDSC IN SIMULATION 1. (POSITIVE P-VALUES INDICATE GAMDSC IS SUPERIOR TO MCMCC; NEGTIVE P-VALUES INDICATE GAMDSC IS INFERIOR TO MCMCC. THE P-VALUES IN BOLDFACE INDICATE SIGNIFICANT IMPROVEMENT OF GAMDSC ON MCMCC.)

		# of obtained covers					
sensing radius	ub	MC-MIP	MCMCC	GAMDSC MAX.	GAMDSC AVG.	GAMDSC STDEV.	p-value
100	5	5	5	5	5	0	
120	8	8	8	8	7.61	0.49	-1.49E-12
140	11	11	11	11	10.32	0.54	-3.56E-22
160	14	14	13	14	12.98	0.71	-0.38E00
180	15	15	14	15	14.92	0.27	**2.10E-56**
200	18	18	17	18	17.66	0.57	**2.62E-20**
220	23	23	20	23	22.54	0.59	**5.46E-66**
240	27	27	23	27	25.83	0.75	**1.03E-60**
260	31	31	28	31	29.72	0.76	**6.51E-41**
280	36	36	30	36	34.79	0.76	**1.66E-81**
300	39	39	35	39	38.31	0.74	**2.45E-67**

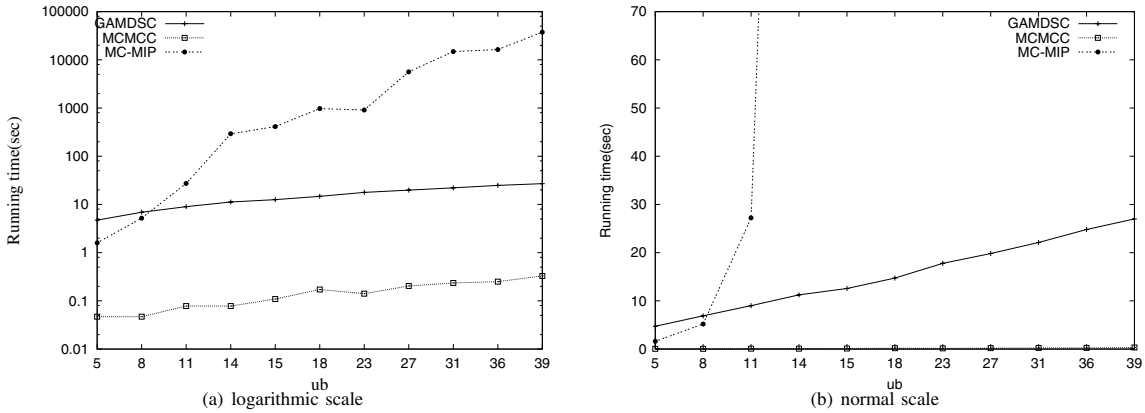

Fig. 6. Comparison of running time in simulation 1. As shown in (b), computation time of GAMDSC grows linearly as ub grows. As shown in (a), computation time of MC-MIP grows exponentially as ub grows.

Despite guarantee of optimal solutions, MC-MIP takes exponential computation time and is impractical in large-scale applications. From another aspect, the execution of long-running, resource-intensive MC-MIP may exhaust sensor energy before finding the solution. Oppositely, GAMDSC can get near-optimal solutions with acceptable computation time and sensor energy.

Figure 7 depicts the any-time behavior of GAMDSC as $ub = 5$, 14, 31, and 39. As minimum covered degree grows, the time to convergence increases.

B. Simulation 2 : controlling the number of sensors

In the second simulation, we fix the sensing radius as 220 and randomly deployed 90 to 140 sensors to cover 10 targets. Since the MC-MIP always return the optimum solution, we focus on GAMDSC and MCMCC. Table III and Fig. 8 show the results of GAMDSC and of MCMCC. The value of ub

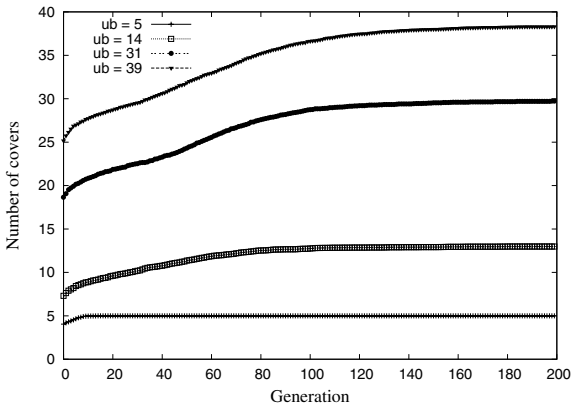

Fig. 7. Any-time behavior of GAMDSC in simulation 1. As ub grows, the time to convergence also grows.

TABLE III

COMPARISON OF THE NUMBER OF COVERS OBTAINED BY MCMCC, AND GAMDSC IN SIMULATION 2. (THE P-VALUES IN BOLDFACE INDICATE SIGNIFICANT IMPROVEMENT OF GAMDSC ON MCMCC)

		# of obtained covers				
Number of sensors	ub	MCMCC	GAMDSC MAX.	GAMDSC AVG.	GAMDSC STDEV.	p-value
90	17	15	17	16.99	0.10	**5.576E-131**
95	24	22	24	23.94	0.23	**4.477E-093**
100	24	21	24	23.90	0.33	**1.249E-095**
105	27	24	27	26.95	0.21	**2.951E-114**
110	30	26	30	29.08	0.77	**5.389E-063**
115	30	26	30	29.53	0.52	**4.990E-085**
120	33	30	33	32.81	0.41	**1.291E-084**
125	27	24	27	26.79	0.45	**9.067E-081**
130	32	31	32	31.97	0.17	**1.682E-077**
135	31	30	31	31	0.00	
140	39	33	39	37.86	0.76	**2.587E-082**

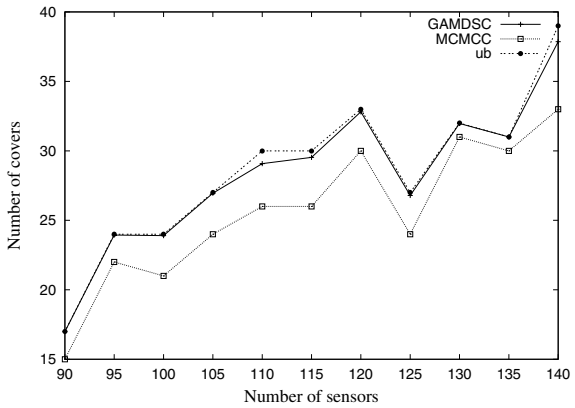

Fig. 8. Comparison of the number of found covers in simulation 2. GAMDSC outperforms MCMCC in all test cases and can get near-optimal solutions.

trends high as the number of deployed sensors increases. In the case of 125 sensors, ub sharply descends because the random deployment generates non-uniform positions of the sensors. Again, we perform one-tailed t-test on the numbers of covers obtained by GAMDSC and MCMCC. The result shows that GAMDSC significantly improves MCMCC in all test cases and gets near-optimal solutions. In the case $ub = 39$, GAMDSC improves MCMCC by 14% in terms of the number of covers.

VI. CONCLUSION AND FUTURE WORK

For surveillance applications, each sensor cover must satisfy the coverage constraint that all targets must be covered. An important scheme for extending the sensor network lifetime is dividing sensors into maximum disjoint sensor covers and letting the sensor covers work by turns. The more sensor covers can be found, the more network life time can be prolonged. This problem can be solved by transformation to a combinatorial optimization problem, disjoint set covers (DSC).

This paper proposes a genetic algorithm, called GAMDSC, to solve the DSC problem. GAMDSC uses integer representation to encode the grouping combination of sensors. Such an one-to-one mapping of genes and sensors has the advantage that the whole genotypic space corresponds to feasible solutions.

Simulation results show that GAMDSC can stably get near-maximal number of covers and can improve MCMCC in the number of obtained covers by 16%. That is to say, for a surveillance system with the lifetime 180 days by MCMCC, the proposed GAMDSC can extend the lifetime for more 28.8 days. Compared to the exponential time complexity of MC-MIP, our approach can get near-optimal results with only polynomial computation time, which is acceptable in large-scale sensor networks.

Although GAMDCS can get near-optimal solutions, there is still room for improvement. In the future, we will investigate the possibility of reaching closer optimum by combining GAMDSC with local search techniques, and will try to derive a parallel genetic algorithm to distribute the computation load evenly to each sensor for further prolonging the network lifetime.

REFERENCES

[1] B. Badrinath, M. Srivastava, K. Mills, J. Scholtz, and K. S. Eds., "Special issue on smart spaces and environments," *IEEE Personal Communications*, vol. 7, no. 5, 2000.

[2] I.F.Akyildiz, W. Su, Y.Sankarasubramaniam, and E.Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, 2002.

[3] D. Li, K. Wong, Y. Hu, and A. Sayeed, ""detection, classification and tracking of targets in distributed sensor networks"," *IEEE Signal Processing Magazine*, vol. 19, 2002.

[4] P. Varshney., "Distributed detection and data fusion," Spinger-Verlag, New York, 1996.

[5] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy-aware wireless microsensor networks," *IEEE Signal Processing*, vol. 19, no. 2, pp. 40–50, Mar. 2002.
[6] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Optimizing sensor networks in the energy-latency-density design space," *IEEE Transactions on Mobile Computing*, vol. 1, no. 1, pp. 70–80, Jan. 2002.
[7] J. A. Stine and G. D. Veciana, "Improving energy efficiency of centrally controlled wireless data networks," *Wireless Networks*, vol. 8, no. 6, pp. 681–700, Nov. 2002.
[8] M. Cardei and D. Z. Du, "Imporving wireless sensor network lifetime through power aware organization," *Wireless Networks*, pp. 333–340, 2005.
[9] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," IEEE international conference on Wireless Communications, Helsinki, Finland, June 2001.
[10] J. Holland, "Adaptation in natural and artificial systems," University of Michigan Press, Ann Arbor. 1975.
[11] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of np-completeness," Freeman, New York, 1979.
[12] Z. Abrams, A. Goel, and S. Plotkin, "Set k-over algorithms for energy efficient monitoring in wireless sensor networks," Third International Symposium on Information Processing in Sensor Networks, 2004.
[13] P. Berman, G. Calinescu, C. Shah, and A. Zelikovsky, "Power efficient monitoring management in mensor metworks," Proceedings of IEEE Wireless Communication and Networking Conference, Atlanta, Georgia, USA, Mar. 2004, pp. 2329-2334.
[14] M. Qin and R. Zimmermann, "Studying Upper Bounds on Sensor Network Lifetime by Genetic Clustering," *Lecture Notes in Computer Science*, vol. 3560, p. 408, 2005.
[15] K. P. Ferentinos and T. A. Tsiligiridis, "Adaptive design optimization of wireless sensor networks using genetic algorithms," *Comput. Networks*, vol. 51, no. 4, pp. 1031–1051, 2007.
[16] Q. Cao, T. Yan, J. A. Stankovic, and T. F. Abdelzaher., "Analysis of target detection performance for wireless sensor networks," In International Conference on Distributed Computing in Sensor Networks (DCOSS), 2005.