
This version is available at https://strathprints.strath.ac.uk/65130/

Strathprints is designed to allow users to access the research output of the University of Strathclyde. Unless otherwise explicitly stated on the manuscript, Copyright © and Moral Rights for the papers on this site are retained by the individual authors and/or other copyright owners. Please check the manuscript for details of any other licences that may have been applied. You may not engage in further distribution of the material for any profitmaking activities or any commercial gain. You may freely distribute both the url (https://strathprints.strath.ac.uk/) and the content of this paper for research or private study, educational, or not-for-profit purposes without prior permission or charge.

Any correspondence concerning this service should be sent to the Strathprints administrator: strathprints@strath.ac.uk
Kuhn-Munkres Parallel Genetic Algorithm for the Set Cover Problem and Its Application to Large-Scale Wireless Sensor Networks

Xin-Yuan Zhang, Yue-Jiao Gong, Member, IEEE, Zhi-Hui Zhan, Member, IEEE, Wei-Neng Chen, Member, IEEE, Yun Li, Member, IEEE, and Jun Zhang, Senior Member, IEEE

Abstract—Operating mode scheduling is crucial for the lifetime of wireless sensor networks. However, the growing scale of networks has made such a scheduling problem more and more challenging, as existing set cover and evolutionary algorithms become unable to provide satisfactory efficiency due to the curse of dimensionality. In this paper, a Kuhn-Munkres parallel genetic algorithm is developed to solve the set cover problem and is applied to lifetime maximization of large-scale wireless sensor networks. The proposed algorithm schedules the sensors into a number of disjoint complete cover sets and activates them in batch for energy conservation. It uses a divide-and-conquer strategy of dimensionality reduction, and the polynomial Kuhn-Munkres algorithm are hence adopted to splice the feasible solutions obtained in each subarea to enhance the search efficiency substantially. To further improve global efficiency, a redundant-trend sensor schedule strategy is developed. Additionally, we mellowate the evaluation function through penalizing incomplete cover sets, which speeds up convergence. Eight types of experiments are conducted on a distributed platform to test and inform the effectiveness of the proposed algorithm. The results show that it offers promising performance in terms of the convergence rate, solution quality, and success rate.

Index Terms—parallel genetic algorithm, set cover problem, large-scale wireless sensor networks, Kuhn-Munkres algorithm.

I. INTRODUCTION

WIRELESS sensor networks (WSNs) have been widely used in a number of fields to satisfy various requirements, such as road traffic monitoring [11], environmental observation [12], healthcare sensing [3], and asset monitoring. Typically, hundreds or even thousands of sensors, each with a series of transceivers, a battery and a micro central processing unit, are deployed in a target area. Since it is impossible to recharge or replace the battery in some scenarios, how to extend the lifetime of WSNs becomes a critical task.

Existing ways for lifetime enhancement are classified into five categories: operation mode control [6], data processing [7][8], sink relocation [9][11], topology control [12][13], and optimal routing [14][16]. There are various definitions of the network lifetime. In this paper, the lifetime of a WSN refers to the duration of time that the network is able to carry out its set mission. Normally, the networks can fulfill their mission if it can guarantee the specified coverage requirements by the sensors deployed, i.e., the set cover condition is satisfied [17].

As summarized in [18][20], the deployment methods for sensors in WSNs vary with applications, which can be categorized into deterministic deployment and random deployment. Deterministic deployment is applied to a small- or medium-scale network in a friendly sensory environment [21][23]. The set cover problem here can be transformed into a minimum set cover problem or its dual problem. There are some certain theoretical developments [24][26] and optimization algorithms [18][21] related to this field. Since this problem is NP-hard, evolutionary-computation based solvers are potentially promising because of their powerfulness in dealing with NP-hard problems. However, the optimal number of sensors cannot be known in advance, which increases the difficulty of applying an evolutionary algorithm, such as the genetic algorithm (GA). The variable length chromosome puts a great challenge to the crossover operation of GA. Nevertheless, this issue has been well solved recently by using a bi-objective GA [27].

When the environment is inaccessible or unfriendly, or the number of sensors is too large, sensors are often scattered from an aircraft or by other means of transportation, which in effect results in random deployment. In order to guarantee coverage and connectivity, the sensors are to be densely deployed in target areas. To construct an energy-efficient WSN in this case, sensors are assigned to different cover sets independent of one another [28]. Activating them in batch ensures that only one cover set is active at a time and the others are scheduled to sleep. This scientific problem is known as the Set K-cover problem [29] or Disjoint Set Covers problem [30], which is a nondeterministic polynomial complete (NP) problem and hence its optimization is NP-hard.

A common objective of solving the Set K-cover problem is to maximize the lifetime of the WSN, but they differ slightly in terms of coverage constraints. In [31], sensors are aimed to be scheduled into K disjoint sets while guaranteeing that the coverage ratio of each set is as high as possible by modeling...
the Set K-cover problem as an N-person card game, and solutions are obtained after a gaming process. A heuristic method, termed the Most Constrained-Minimally Constraining Covering (MCMCC), is proposed in [29]. The essence of MCMCC is to minimize the coverage of sparsely covered areas within one cover set. It requires that each cover set is able to cover the target area completely. We focus on complete cover sets in this paper.

Owing to their success in solving nondeterministic polynomial problems, GAs [32]-[34] and other evolutionary algorithms (EAs) [35]-[37] have been applied to the lifetime problem in WSNs recently. Lai et al. [38] propose a GA for maximum disjoint set covers (GAMDSC), which applies a scattering operator to the EA offspring to keep critical sensors from joining the same cover set. Hu et al. [39] propose a schedule transition hybridized genetic algorithm (STHGA), which adopts a forward encoding scheme for chromosomes and utilizes redundancy information via designing a series of transition operations. Ant-colony optimization for maximizing covered areas within one cover set. It requires that each cover set is able to cover the target area completely. We focus on complete cover sets in this paper.

In this section, we introduce the sensor model adopted in our algorithm, which is crucial for the coverage, connectivity, and energy consumption.

II. PROBLEM FORMULATION

A. Sensor Model

In this section, we introduce the sensor model adopted in our algorithm, which is crucial for the coverage, connectivity, and energy consumption.
there is still room for KMSPGA to adapt this framework to fitting new forms of energy constraints.

B. Set K-Cover Problem

In this paper, we focus on maximizing the lifetime of a large-scale WSN using a static optimization strategy. The proposed KMSPGA is an off-line algorithm that pre-calculates K disjoint complete cover sets at a time. By alternatively activating the K cover sets in batch, the lifetime of the WSN can be K times larger than the lifetime of a single cover set. Hence, maximizing the lifetime of the WSN is equivalent to finding the maximum disjoint complete cover sets.

We consider the Set K-cover problem a scheduling problem. For a positive integer K, sensors are judiciously scheduled into K disjoint cover sets such that each cover set is able to meet the coverage requirement. In this paper, we focus on a complete coverage. The premise of finding K disjoint complete cover sets is that every element of the target area is covered by at least K sensors. Sufficiency of this premise is proven in the following of this subsection. Fig. 2 illustrates the concept of an element and a K-covered element. In Fig. 2 (a), the grey square area is divided into eight elements. Element Eᵢ in Fig. 2 (b) is covered by four sensors; hence it is called 4-covered.

Without loss of generality, assume that a target area Γ is a rectangle and that N sensors S₁, S₂, ..., S₇ are randomly deployed in Γ. A constraint of the coverage requirements is a complete coverage. Given a positive number K, sensors are scheduled into K cover sets C = {C₁, C₂, ..., C₇}. For each cover set Cᵢ (i = {1, 2, ..., K}), if every element of C is covered by at least one sensor in Cᵢ, then C would be considered as a feasible solution of the Set K-cover problem. The Set K-cover problem can be formalized as:

\[ \bigcup_{S_k \in C_i} E(S_k) \geq \Gamma \]  \hspace{1cm} (3)

\[ \bigcup_{i=1}^{K} C_i \subseteq S \]  \hspace{1cm} (4)

\[ C_i \cap C_j = \emptyset, \quad i \neq j, \quad 1 \leq i, j \leq K \]  \hspace{1cm} (5)

where E(Sₖ) represents the element sensed by sensor Sₖ, k is the sensor index, S is the collection of sensors. Assume that the target area is partitioned into M elements E₁, E₂, ..., Eₘ. To make a clear explanation of how to estimate the upper bound of K, we first give the following proposition and its proof.

Proposition 1: The prerequisite of finding K disjoint cover sets is that each element is covered by at least K sensors.

Proof: Assume that Eᵢ is covered by Q sensors \(S_{c,i} = \{S_{c,i}1, S_{c,i}2, ..., S_{c,i}Q\}\), where \(S_{c,i}\) is the collection of sensors covering element Eᵢ. There still exist K disjoint complete cover sets with \(Q < K\).

Let C be a feasible solution of the Set K-cover problem. Then, we have \(|C| = K\). Considering the same element Eᵢ in the assumption, Eᵢ is expected to be covered in each \(C_i \subseteq C\) according to (3). Then, K cover sets are considered as K covering tasks, and we have Q sensors that can perform this task. Then, K tasks are assigned to Q sensors. Since we have \(|C| = K > |SC| = Q\), there exist cover sets \(C_p, C_q \in C\), and sensor \(S_{m,m} \in SC_m\) \((m \in \{1, 2, ..., Q\})\) satisfying \(S_{m,m} \in C_p \cap C_q\) according to the drawer principle, and therefore it contradicts

![Fig. 2. Notion of an element and a K-covered element. (a) The shaped area is divided into eight elements E₁, E₂, ..., E₇, where four elements are covered by two sensors and the remaining four by one only. (b) Element Eᵢ is called K-covered, if it is covered by K sensors, in which case K = 4.](image)

![Fig. 3. Elements E₁, E₅, E₆, E₈, and E₁₅ are called critical elements, as they are covered by the minimum number of sensors.](image)
\[
\delta = \frac{1}{T} \sum_{i=1}^{N} |N_g(S_i)| \\
N_g(S_i) \cap N_g(S_j) = \emptyset, \quad i \neq j, \; i, \; j \in \{1, \ldots, N\}
\]

The coverage criteria stipulates that grid \( g \) is covered by sensor \( S_i \) only if all its four vertices are within the sensing range of \( S_i \). Fig. 4 shows how to calculate the coverage ratio. This calculation method is widely used to estimate the coverage ratio. However, the grid width can influence the computational complexity of the coverage ratio in terms of computational complexity and accuracy. Fig. 5 gives an example to explain this special case, where the grey grid is apparently covered by the WSN. Unfortunately, it is regarded as uncovered due to the above coverage criteria whether a grid is covered. If the width of this grid is halved, two resultant grids become covered. However, the calculation of the coverage ratio is of an \( O(N \times T) \) computational complexity. A shorter width means a higher computational complexity according to (7).

In our work, we adopt a simple strategy to determine the grid width in a preprocessing step. Given \( \mathcal{P} \) kinds of \( d_i \) \( (i = 1, 2, \ldots, \mathcal{P}) \), \( d_i < d_{i+1} \) in the process of estimating the upper limit of \( K \), we choose the smallest \( d_i \) first to obtain an exact value \( \hat{U}_i \), because \( d_i \) is small enough to guarantee accuracy. Then \( d_i \) \((i = 1, 2, 3, \ldots, \mathcal{P})\) is adopted to work out \( \hat{U}_i \) in sequence. The largest \( d_i \) (ensuring \( \hat{U}_i = \hat{U}_i \)) is used for calculating the coverage ratio. Algorithm 1 presents a set of pseudocode of this preprocess of choosing a proper grid width. In this paper, we adopt 5 kinds of grid widths: \( (d_1, d_2, d_3, d_4, d_5) = (0.625, 0.78125, 1, 1.25, 1.5625) \).

III. PROPOSED PARALLEL GENETIC ALGORITHM

A. Kuhn-Munkres Parallel Genetic Approach

KMSPGA is designed on a divide-and-conquer strategy, and the polynomial KM algorithm is adopted to splice the feasible solutions obtained in each subarea. The framework of KMSPGA is shown in Fig. 6. In the first step, we uniformly divide the target area into a number of subareas and encode them. Assuming that the number of sensors within subarea \( A_i \) is \( N_i \), it satisfies:

\[
\sum_{i=1}^{L \times W} N_i = N
\]

where \( L \) and \( W \) denote the number of partitions along horizontal and vertical directions respectively, and hence \( L \times W \) denotes the number of subareas obtained. Therefore, the Set K-cover problem size of subarea \( A_i \) is whittled down to \( N_i \). After the partition process, the small-scale Set K-cover problem within each subarea is separately solved by a parallel processing module. When local solutions of each small-scale Set K-cover problem reach a predefined state, they are spliced through a KM combination operation to achieve global optimization efficiently. There are two termination conditions: 1) FEs reaches its upper limit and/or 2) the number of complete cover sets reach \( \hat{U} \). KMSPGA deploys a termination-controller in its master process. The controller checks whether the current process has reached the termination condition at the end of every generation. If so, the master process broadcasts a termination signal to all the slave processes and the master process begins to control processes. Fig. 4 gives an example to show how the grid width influences the computation of the coverage ratio. (a) The light grey grids are covered while the black one is not covered because one of its vertices is beyond the sensing range of the sensor. (b) The black grid is covered by two sensors \( S_{i_1} \) and \( S_{i_2} \) but only belongs to either of \( N_{i_1}(S_i), i \in \{p, q\} \), in the case of a repeat count. (c) The coverage ratio is 11/25 = 0.44.

Algorithm 1: Preprocessing of choosing a proper grid width

1: Procedure WIDTHCHOOSE \( \{d_1, \ldots, d_\mathcal{P}\} \)
2: \( d \leftarrow d_1 \); 
3: Compute \( \hat{U}_1 \); 
4: for \( i = \mathcal{P} \rightarrow 2 \) do 
5: \( d \leftarrow d_i \); 
6: Compute \( \hat{U}_i \); 
7: if \( \hat{U}_i = \hat{U}_i \) then 
8: \( d \leftarrow d_i \); 
9: break 
10: end if 
11: end for 
12: return \( d \); 
13: end procedure

Fig. 5. An example to show how the grid width influences the computation of the coverage ratio. (a) The grey grid is regarded as uncovered due to the coverage criteria. (b) The grey grid becomes covered by either of the two sensors after the grid width is shortened.
processes and, afterwards, all the processes of KMSPGA terminate. Pseudocode of this divide-and-conquer strategy is given in Algorithm 2.

Divide() contains two steps. Firstly, the target area $\Gamma$ is uniformly partitioned into $L \times W$ subareas $A = (A_1, A_2, \ldots, A_{L \times W})$. Fig. 7 gives four examples where $\Gamma$ is divided into 2, 8, 32, and $L \times W$ subareas in Fig. 7 (a), (b), (c), and (d), respectively. Then, the centers of all sensors are traversed to obtain a classification $SE = \{SE_1, SE_2, \ldots, SE_{L \times W}\}$, where $SE_i$ is the collection of sensors falling in subarea $A_i$. Every sensor within $SE_i$ satisfies that its central coordinate $(s_x, s_y) \in A_i$ ($k = 1, 2, \ldots, N_i$).

Additionally, if the center of a sensor falls on the boundaries of two or more subareas, it will be randomly scheduled into any one of the subareas. In order to keep the concision of KMSPGA, we adopt a uniform partition here instead of other ways such as clustering techniques. Besides, the uniform partition is convenient for the following combination operation.

In the parallel processing module, each process evolves a sub-population to obtain a feasible solution. The collection of sub-populations is formulated as $SP = (SP_1, SP_2, \ldots, SP_{L \times W})$. Each sub-population size is $N_p$. They are evolved independently through a selection, crossover, mutation, and RTSS operation in processing(). We estimate the state of each sub-population through periodically sampling the best individual at a sampling frequency $f_s$. A state factor $\zeta$ is computed as follows:

$$\zeta = \frac{1}{U} \sum_{i=1}^{L \times W} U_i$$

where $U_i$ is the number of disjoint complete cover sets obtained by the best individual of $SP_i$. Since $U_i \leq U$ ($i = 0, 1, \ldots, L \times W$), we have $\zeta \leq L \times W$. The upper limit of the threshold value, $\zeta^*$, is set to $L \times W - 1$. The value of $\zeta$ determines whether the KM combination operation will be applied. The independent evolution process will be terminated until $\zeta$ reaches $\zeta^*$. Therefore, instead of performing the KM operation every generation, the execution timing of KM is adaptively adjusted based on the state factor. This way, the effectiveness of the operation is improved, and hence the computational cost is substantially reduced. The threshold $\zeta^*$ determines the frequency of performing the KM operation (merging the local feasible solutions) and then checking for the termination condition. This procedure however does not affect the main evolution process of solutions to much extent. Thus, different settings of $\zeta^*$ will not change the output solution quality of the proposed algorithm, but only influence the execution time. The standard uniform crossover and uniform mutation are adopted in the evolutionary process. RTSS is conducted right after mutation, and before fitness evaluation, which is introduced in detail in Subsection D. The tournament selection is adopted because of its efficiency, with a tournament size $T_c$.

Conquer() is a combination operation in order to merge the feasible local solutions $C = (C_1, C_2, \ldots, C_{L \times W})^T$, $C_i = (C_{i,1}, C_{i,2}, \ldots, C_{i,17})$. $C_i$ is the best solution of the small-scale Set K-cover in subarea $A_i$. Fig. 8 shows an example of this merging process, where $\Gamma$ is divided into 2x4 subareas. We need three steps to merge $(C_1, C_2, \ldots, C_8)$ into $C_{1,8}$. Considering a couple of neighboring sub-populations $p$ and $q$, Conquer() is expected

![Fig. 7. Division of the target area and encoding of the subareas.](image)

![Fig. 8. Combination of the subareas.](image)

![Fig. 9. Bipartite combinations of solutions $C_p$ and $C_q$ obtained by neighbor sub-populations, such that the Kuhn-Munkres algorithm can be applied.](image)
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2511142, IEEE Transactions on Evolutionary Computation

Algorithm 3 Fitness evaluation

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><strong>Procedure</strong> CHROMOSOME EVALUATION ((x_1, \ldots, x_n))</td>
</tr>
<tr>
<td>2.</td>
<td>for (i = 1 \rightarrow N) do</td>
</tr>
<tr>
<td>3.</td>
<td>(SF_i \leftarrow 0);</td>
</tr>
<tr>
<td>4.</td>
<td>(RF_i \leftarrow \text{true}; // S_i \text{ is redundant if } RF_i \text{ is true} )</td>
</tr>
<tr>
<td>5.</td>
<td>end for</td>
</tr>
<tr>
<td>6.</td>
<td>for (i = 1 \rightarrow T) do</td>
</tr>
<tr>
<td>7.</td>
<td>for (j = 1 \rightarrow N) do</td>
</tr>
<tr>
<td>8.</td>
<td>(k \leftarrow x_j);</td>
</tr>
<tr>
<td>9.</td>
<td>if (g_i \subseteq N_k(S_i)) and (SF_k = 0) then</td>
</tr>
<tr>
<td>10.</td>
<td>(CN_k \leftarrow CN_k + 1);</td>
</tr>
<tr>
<td>11.</td>
<td>(SF_k \leftarrow 1);</td>
</tr>
<tr>
<td>12.</td>
<td>(RF_j \leftarrow \text{false}; )</td>
</tr>
<tr>
<td>13.</td>
<td>end if</td>
</tr>
<tr>
<td>14.</td>
<td>end for</td>
</tr>
<tr>
<td>15.</td>
<td>end for</td>
</tr>
<tr>
<td>16.</td>
<td>(f \leftarrow 0);</td>
</tr>
<tr>
<td>17.</td>
<td>for (i = 1 \rightarrow \hat{U}) do</td>
</tr>
<tr>
<td>18.</td>
<td>(\delta_i \leftarrow CN_i T^{-1});</td>
</tr>
<tr>
<td>19.</td>
<td>(f \leftarrow f + \delta_i P(\delta_i));</td>
</tr>
<tr>
<td>20.</td>
<td>end for</td>
</tr>
<tr>
<td>21.</td>
<td>end procedure</td>
</tr>
</tbody>
</table>

where \(n\) is eight and \(\hat{U}\) is four. On the contrary, \(X\) can be easily transformed from \(C_X\). Therefore, \(X\) and \(C_X\) are equivalent on representing an individual. In the remainder of this section, we adopt the \(C_X\) structure in representing a chromosome because this form is more convenient for introducing and describing the operations of KMSPGA while \(X\) is adopted in the practical implementation of KMSPGA.

### C. Improved Fitness Index

In STHGA\(^{39}\), the evaluation function is defined as (15), where \(\delta_i\) represents the coverage ratio of cover set \(C_i\). The computation of \(\delta_i\) is shown in (16), where the value of \(\delta_{i,k}\) is 1 if grid \(k\) is covered by \(C_i\). The value of \(\delta_i\) \((i \in \{1, 2, \ldots, \hat{U}-1\})\) is 1 in STHGA because of the forward encoding scheme. The evaluation function of GAMDSC\(^{38}\)is shown in (18), where \(f_B\)represents the number of disjoint complete cover sets and \([x]\) denotes the floor of \(x\).

\[
f_A = \frac{1}{T} \sum_{i=1}^{\hat{U}} \delta_i \tag{15}
\]

\[
\delta_i = \frac{L \cdot W}{T} \sum_{k=1}^{(LW)^{-1}} \delta_{i,k} \tag{16}
\]

\[
\delta_{i,j} = \begin{cases} 1, & \text{if grid } j \text{ is covered by } C_i \\ 0, & \text{otherwise} \end{cases} \tag{17}
\]

\[
f_B = \sum_{i=1}^{\hat{U}} [\delta_i] \tag{18}
\]

In order to improve the convergence rate, we adopt a penalty function \(P(\delta_i)\) of (19), where \(\lambda\) is the penalty coefficient. The fitness evaluation function of KMSPGA is given in (20). Hence, the contribution of an incomplete cover set is lower than a complete one because of this penalty. Therefore, individuals with more incomplete cover sets are
eliminated more easily than those with more complete cover sets. Algorithm 3 shows the pseudocode of the fitness evaluation in KMSPGA.

\[
P(\delta_i) = \begin{cases} 
1, & \text{if } d_i = 1 \\
\lambda, & \text{otherwise} 
\end{cases} \tag{19}
\]

\[
f = \sum_{i=1}^{U} \delta_i \cdot P(\delta_i) \tag{20}
\]

D. Redundant-Trend Sensors Schedule Operation

In RTSS, the redundant information is indirectly utilized in order to improve search efficiency. The redundant information is collected in the fitness evaluation process. In Algorithm 3, steps 2 to 15 give this collection process. As can be noted from the pseudocode, the collection process is embedded in the fitness evaluation in case of increasing computational complexity. Note that after collecting the redundant information, RTSS is not applied directly after the fitness evaluation, but, as introduced in subsection A, it is performed after crossover and mutation operations, the landscape of the chromosome may change. Thus, the redundant information utilized in RTSS is hysteretic.

Considering that \( S_k \) is a member of \( C_j \), whether \( S_k \) is redundant for \( C_j \) depending on its contributions to \( C_j \). \( S_k \) is considered to be redundant only if it has no contributions to \( C_j \), which is judged in the fitness evaluation. However, \( S_k \) may not still be redundant, because crossover and mutation operations may change the members of \( C_j \). Therefore, \( S_k \) is called redundant-trend sensors in RTSS due to this uncertainty of the redundant state. Fig. 11 shows an example of this uncertainty. The grey sensor is considered to be redundant after the fitness evaluation operation. However, it is uncertain whether it is still redundant after crossover or mutation.

The process of RTSS is described as follows. Suppose that the cover set is \( C = \{ C_1, C_2, \ldots, C_m \} \). Firstly, we traverse the redundant information of the sensors in \( C_i \). Assuming \( S_{i,k} \) is the \( k \)th member of \( C_i \), if \( S_{i,k} \) is judged to be redundant for \( C_i \) in Algorithm 3, we then consider it a redundant-trend sensor in RTSS. A cover set \( C_m (m \in \{1, 2, \ldots, U\}) \) will receive \( S_{i,k} \) through a tournament selection, where \( N_c \) candidates are randomly selected and the one with the lowest coverage ratio is chosen to receive \( S_{i,k} \) as one of its members. Fig. 12 illustrates this schedule strategy between disjoint cover sets. In Fig. 12, different polygons represent different cover sets, the grey sensors are redundant-trend sensors, and the direction of arrow represents the schedule direction. RTSS has twofold functions. It helps enhance the coverage ratio through the schedule strategy if the redundant-trend sensor is actually a redundant one. However, if the redundant sensor is no longer redundant for the current cover set, the scheduling operation becomes a disturbance for the population. This kind of stochastic disturbance enriches the diversity of the population.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

In this section, experiments are conducted to ascertain the performance of KMSPGA. In Subsection B, we compare KMSPGA with the state-of-the-art algorithms. MCMCC, GAMDSC, and STHGA are serial algorithms which perform well in solving the Set K-cover problem. The experiments and comparisons are used to verify the effectiveness of our proposed KMSPGA algorithm for lifetime maximization of large-scale WSNs. In Subsection C, we compare KMSPGA with a traditional pure parallel genetic algorithm (PGA). Further, the performance of the PGA embedding only RTSS (SPGA) or KM combination (KMPGA) are also tested in order to study the effectiveness of the two operations. Experiments in Subsection D and E are designed to evaluate the robustness of KMSPGA with different partitions and the redundant rates. In Subsection F, we conduct parameter investigation and give their suggested values. Finally, experiments in Subsections G-I are conducted with new and different testing scenarios to further verify the performance of KMSPGA.

KMSPGA and other algorithms are tested on a computer cluster of 25 nodes (with a total of 100 processing cores), which is homogenous with the same Intel core i3-3240 CPU running at 3.40 GHz, 4GB memory and Ubuntu 12.04 LTS.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P )</td>
<td>Number of grid width classification.</td>
<td>5</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>Penalty coefficient in cost evaluation.</td>
<td>0.2</td>
</tr>
<tr>
<td>( L )</td>
<td>Number of partitions along horizontal direction.</td>
<td>[2, 4]</td>
</tr>
<tr>
<td>( W )</td>
<td>Number of partitions along vertical direction.</td>
<td>[2, 4, 8]</td>
</tr>
<tr>
<td>( f_s )</td>
<td>Sampling frequency.</td>
<td>3000</td>
</tr>
<tr>
<td>( P_c )</td>
<td>Crossover probability.</td>
<td>0.6</td>
</tr>
<tr>
<td>( P_m )</td>
<td>Mutation probability.</td>
<td>0.001</td>
</tr>
<tr>
<td>( N_c )</td>
<td>Number of candidates in RTSS.</td>
<td>( 3^U )</td>
</tr>
<tr>
<td>( N_p )</td>
<td>Sub-population size.</td>
<td>15</td>
</tr>
<tr>
<td>( T_s )</td>
<td>Tournament size.</td>
<td>5</td>
</tr>
<tr>
<td>( \zeta )</td>
<td>Threshold of the state factor with 2x2 partitions.</td>
<td>3.0</td>
</tr>
<tr>
<td>( \xi )</td>
<td>Threshold of the state factor with 2x4 partitions.</td>
<td>7.0</td>
</tr>
<tr>
<td>( \eta )</td>
<td>Threshold of the state factor with 4x4 partitions.</td>
<td>15.0</td>
</tr>
<tr>
<td>( \psi )</td>
<td>Threshold of the state factor with 4x8 partitions.</td>
<td>30.0</td>
</tr>
</tbody>
</table>
obtained by GAMDSC are fewer than Ũ. STHGA possesses a strategy to handle such a large number of sensors, solutions whose dimensionality is small or medium. However, the computational complexity of MCMCC becomes so high in terms of performance of KMSPGA in comparison with the serial B.

MCMCC is \(O(N^2)\) for a single run and \(O(N^4)\) for an auxiliary parallel run. In this subsection, we investigate the effectiveness and reliability of KMSPGA with the same maximum number of function evaluations so that the comparison is fair. Some of the compared algorithms, e.g., STHGA, are not suitable for parallelism. The reasons are as follow. As can be noted from our description of the proposed parallel genetic framework, the disjoint cover sets within the same chromosome are supposed to be peer to each other. In STHGA, \(C_i\) is a complete cover set while \(C_{i \cup j}\) is incomplete due to the forward encoding scheme, which makes \(C_i\) (i.e., \(i, j \in U\)) not peer to cover set \(C_{i \cup j}\). Combination between any \(C_{i \cup j}\) and \(C_{i\cup j}\) (i, j < U) makes no sense to these already complete cover sets. The complicated auxiliary search

### Table II

<table>
<thead>
<tr>
<th>Instance</th>
<th>(N)</th>
<th>(R)</th>
<th>(\eta)</th>
<th>(U)</th>
<th>GAMDSC</th>
<th>STHGA</th>
<th>KMSPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FE Mean</td>
<td>Std</td>
<td>t-test*</td>
</tr>
<tr>
<td>I1-1</td>
<td>5000</td>
<td>5</td>
<td>4.760</td>
<td>33</td>
<td>15000</td>
<td>2.0</td>
<td>0.0</td>
</tr>
<tr>
<td>I1-2</td>
<td>8</td>
<td>8</td>
<td>4.845</td>
<td>83</td>
<td>15000</td>
<td>7.4</td>
<td>0.67</td>
</tr>
<tr>
<td>I2-1</td>
<td>10000</td>
<td>5</td>
<td>4.488</td>
<td>70</td>
<td>30000</td>
<td>1.8</td>
<td>0.50</td>
</tr>
<tr>
<td>I2-2</td>
<td>8</td>
<td>8</td>
<td>4.497</td>
<td>179</td>
<td>30000</td>
<td>10.2</td>
<td>0.57</td>
</tr>
<tr>
<td>I3-1</td>
<td>15000</td>
<td>5</td>
<td>4.446</td>
<td>106</td>
<td>45000</td>
<td>1.3</td>
<td>0.50</td>
</tr>
<tr>
<td>I3-2</td>
<td>8</td>
<td>8</td>
<td>4.435</td>
<td>272</td>
<td>45000</td>
<td>12.5</td>
<td>0.57</td>
</tr>
<tr>
<td>I4-1</td>
<td>20000</td>
<td>5</td>
<td>4.303</td>
<td>146</td>
<td>60000</td>
<td>1.1</td>
<td>0.35</td>
</tr>
<tr>
<td>I4-2</td>
<td>8</td>
<td>8</td>
<td>4.347</td>
<td>370</td>
<td>60000</td>
<td>13.2</td>
<td>0.76</td>
</tr>
<tr>
<td>I5-1</td>
<td>25000</td>
<td>5</td>
<td>4.462</td>
<td>176</td>
<td>75000</td>
<td>1.6</td>
<td>0.51</td>
</tr>
<tr>
<td>I5-2</td>
<td>8</td>
<td>8</td>
<td>4.458</td>
<td>451</td>
<td>75000</td>
<td>16.1</td>
<td>0.57</td>
</tr>
<tr>
<td>I6-1</td>
<td>30000</td>
<td>5</td>
<td>4.446</td>
<td>212</td>
<td>90000</td>
<td>2.5</td>
<td>0.50</td>
</tr>
<tr>
<td>I6-2</td>
<td>8</td>
<td>8</td>
<td>4.570</td>
<td>528</td>
<td>90000</td>
<td>20.8</td>
<td>0.92</td>
</tr>
</tbody>
</table>

1 Two tailed t-test of the null hypothesis that GAMDSC is equal to KMSPGA. 2 Two tailed t-test of the null hypothesis that STHGA is equal to KMSPGA.

---

**Fig. 13.** Convergence curves of the compared algorithms. (a) I3-2. (b) I6-1. (c) I3-1. The number of sensors is less than 5000. The complicated local search operations help STHGA search the problem space efficiently. However, the performance of this serial GA is badly influenced by curse of dimensionality. The success rate of instance I3-1 obtained by STHGA is 0.03. At the same time, the large number of function evaluations indicates that the searching efficiency reduces due to the curse of dimensionality. The success rate of instance I3-1 obtained by STHGA is 0.03. At the same time, the large number of function evaluations indicates that the searching efficiency reduces due to the curse of dimensionality.

In this subsection, we investigate the effectiveness and reliability of KMSPGA and the other compared algorithms with the same maximum number of function evaluations so that the comparison is fair. Some of the compared algorithms, e.g., STHGA, are not suitable for parallelism. The reasons are as follow. As can be noted from our description of the proposed parallel genetic framework, the disjoint cover sets within the same chromosome are supposed to be peer to each other. In STHGA, \(C_i\) is a complete cover set while \(C_{i \cup j}\) is incomplete due to the forward encoding scheme, which makes \(C_i\) (i.e., \(i, j \in U\)) not peer to cover set \(C_{i \cup j}\). Combination between any \(C_{i \cup j}\) and \(C_{i\cup j}\) (i, j < U) makes no sense to these already complete cover sets. The complicated auxiliary search
obtains larger mean U than KMPGA in the majority of the KMSPGA outperforms KMPGA and SPGA in all of the solution quality, which is made available by KMSPGA. KMSPGA are suitable for the problems of a high dimensionality or of large-scale function optimization [54]. They are adopted to either speed up the optimization or enhance the solution quality through a dimensionality reduction strategy. In this paper, the dimensionality and computational complexity of Set K-cover problem become so high in large-scale WSN that we adopt a parallel evolutionary algorithm for performance enhancement.

C. Comparison with Parallel Algorithms

In this subsection, we compare KMSPGA with the pure PGA, KMPGA, and SPGA. The PGA combined only with the KM combination or RTSS form KMPGA or SPGA. The experimental instances in Subsection B are adopted here. We conduct the experiments under two different partitions: P_{2×4} and P_{4×4}.

The experimental results are given in Table III and Table IV. KMSPGA outperforms KMPGA and SPGA in all of the instances. Although S_r of KMSPGA and SPGA is 0 in the majority of the instances, the mean U obtained by KMPGA and SPGA is much larger than PGA, which reveals that the KM combination and RTSS contribute to the enhancement of the solution quality, which is made available by KMSPGA. Take instance I4-2 of partition 2×4 as an example, where U is 370. The mean U obtained by PGA is 156.0, accounting for only 42.16%. As for KMPGA and SPGA, the mean U obtained are 290 and 358.2, accounting for 78.38% and 96.81%. SPGA obtains larger mean U than KMPGA in the majority of the instances, and therefore contribution of RTSS is larger than that of KM combination when it comes to the degree of the improvement of solution quality.

It is worth mentioning that parallel evolutionary algorithms are suitable for the problems of a high dimensionality or of complex and time-consuming computation features such as large-scale air traffic flow optimization [52] resource allocation in classic economic field [53] and large-scale function optimization [54][55]. They are adopted to either speed up the optimization or enhance the solution quality through a dimensionality reduction strategy. In this paper, the dimensionality and computational complexity of Set K-cover problem become so high in large-scale WSN that we adopt a parallel evolutionary algorithm for performance enhancement.

D. Discussion on Correlation between Sensors and Subareas

We investigate how the number of partitions affects the performance of parallel processing module and the KM combination. Fig. 14 gives a visual illustration of a special kind of correlative area (marked as grey), and sensors falling in grey areas are relevant with the largest number of subareas. Sensors falling in grey areas are correlation between 2, 4, 6 and 8 subareas in (a), (b), (c), and (d) respectively.
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2511142, IEEE Transactions on Evolutionary Computation

IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION

TABLE V

<table>
<thead>
<tr>
<th>Instance</th>
<th>P_{2x2} (\alpha_{x}=1.38, \alpha_{s}=1.67)</th>
<th>P_{2x4} (\alpha_{x}=1.79, \alpha_{s}=2.41)</th>
<th>P_{4x4} (\alpha_{x}=2.30, \alpha_{s}=3.43)</th>
<th>P_{4x8} (\alpha_{x}=3.41, \alpha_{s}=5.55)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE</td>
<td>Mean</td>
<td>Mean</td>
<td>Std</td>
<td>S</td>
</tr>
<tr>
<td>11-1</td>
<td>4965</td>
<td>33</td>
<td>1.00</td>
<td>2688</td>
</tr>
<tr>
<td>11-2</td>
<td>2730</td>
<td>83</td>
<td>1.00</td>
<td>2922</td>
</tr>
<tr>
<td>12-1</td>
<td>14604</td>
<td>69.9</td>
<td>0.57</td>
<td>8418</td>
</tr>
<tr>
<td>12-2</td>
<td>9010</td>
<td>179</td>
<td>1.00</td>
<td>9324</td>
</tr>
<tr>
<td>13-1</td>
<td>41490</td>
<td>1014</td>
<td>1.24</td>
<td>23850</td>
</tr>
<tr>
<td>13-2</td>
<td>3718</td>
<td>272</td>
<td>1.00</td>
<td>15488</td>
</tr>
<tr>
<td>14-1</td>
<td>59445</td>
<td>143.8</td>
<td>0.44</td>
<td>33086</td>
</tr>
<tr>
<td>14-2</td>
<td>52500</td>
<td>369.7</td>
<td>0.71</td>
<td>35838</td>
</tr>
<tr>
<td>15-1</td>
<td>67695</td>
<td>174.5</td>
<td>1.98</td>
<td>26761</td>
</tr>
<tr>
<td>15-2</td>
<td>66675</td>
<td>449.3</td>
<td>1.82</td>
<td>36955</td>
</tr>
<tr>
<td>16-1</td>
<td>87660</td>
<td>209.6</td>
<td>2.22</td>
<td>22845</td>
</tr>
<tr>
<td>16-2</td>
<td>78615</td>
<td>526.1</td>
<td>2.83</td>
<td>34785</td>
</tr>
</tbody>
</table>

Table V lists the results with different partitions: P_{2x2}, P_{2x4}, P_{4x4}, and P_{4x8}, where \( \alpha_{s} \) represents the value of \( \omega \) with sensing radius \( r \). P_{4x4} achieves the best performance in the majority of test instances in terms of convergence rate, solution quality, and success rate. The success rate is low considering the performance of P_{2x2} and P_{4x8}. However, the reason is completely different. As for P_{2x2}, the partition quantity is not enough to reduce the dimensionality to an acceptable level. On the contrary, the partition quantity of P_{4x8} is so large that the correlation between sensors and subareas becomes too high to apply the divide-and-conquer strategy. As can be noted in instance Ix-1 and Ix-2, the success rate of the former is clearly higher than the latter because of the smaller value of \( \omega \) in Ix-1 than that in Ix-2. Consequently, the number of partitions is restricted by the value of \( \omega \). In order to help the divide-and-conquer strategy work efficiently, the partition quantity is limited to an appropriate range.

E. Experiments with Different Redundant Rate

In this subsection, we study the influence of different redundant rates on the solution quality. Two groups of experimental instances are adopted in this experiment, where the sensor radius is 5. Instances prefixed by “J” represent the number of sensors is 20,000, whereas instances prefixed by “H” represent the number of sensors is 25,000. Although the number and the sensing radius of sensors are fixed, the redundant rate can be different because of the random deployment strategy. It is quite difficult to generate an instance with a specified \( \bar{U} \). Instead, to generate this test set with different redundant rates, we create a relatively large number of candidate instances, calculate their \( \bar{U} \)'s, and then select the candidate instances with appropriate \( \bar{U} \) to the test set. The redundant rate ranges from 3.997 to 5.003.

Performance of different partitions, i.e. P_{2x2}, P_{4x4}, and P_{4x8}, is tested. STHGA is also adopted for comparison. Results are summarized in Table VI. KMSPGA (P_{2x4}, P_{4x4}, and P_{4x8}) achieves a high success rate in a large range of redundant rates, which indicates that KMSPGA offers very promising performance with robustness. P_{4x4} achieves the fastest convergence rate, the largest mean \( U \), and the highest \( S \) in the majority of the instances in Table VI. As far as STHGA is concerned, the success rate declines sharply when the redundant rate decreases. The x-axis of Fig. 15 (a) and (b) is the redundant rate, the y-axis of Fig.15 (a) represents mean \( S \), and the y-axis of Fig. 15 (b) represents the ratio of mean \( U \) to \( \bar{U} \). The best solution obtained by KMSPGA among different partitions in each instance is represented by \( \bar{U} \). In Fig. 15 (a), KMSPGA (P_{2x4}, P_{4x4}, and P_{4x8}) obtains a high \( S \) in most of the instances while the \( S \) of STHGA declines sharply as \( \eta \) decreases. Fig. 15 (b) also indicates that KMSPGA possesses high solution quality within a large-scale range of \( \eta \). In fact, \( \bar{U} \) maintains a value of 1.00 in both Fig. 15 (a) and (b).

F. Parameter Investigation

We investigate the penalty coefficient \( \lambda \) and the threshold value of state factor \( \zeta \) in this subsection in order to show how these parameters influence the performance of KMSPGA. The penalty coefficient \( \lambda \) is adopted to improve the convergence rate. We adopt test instances I4-2 and I5-1 with partition P_{4x4}. The value of \( \lambda \) is increased from 0 to 1 with step 0.05. Based on the experimental results, we give the relationship between \( \lambda \) and FEs in Fig. 16. As can be noted from the results, the value of FEs when \( \lambda \) is set within [0.05, 0.95], the required FEs is significantly less than that when \( \lambda \) is set to 1 (no penalty). The experimental results indicate that the penalty coefficient \( \lambda \) has an effect of improving the convergence rate of KMSPGA. Meanwhile, KMSPGA is generally insensitive to the value of \( \lambda \), and it works identically well when \( \lambda \) is set to [0.2, 0.7].
Influence on the mean success rates. (b) Influence on the mean ratios.

Fig. 15. Influence of redundant rate on the performance of KMSPGA. (a) Influence on the mean success rates. (b) Influence on the mean ratios.

instance I2-2 to investigate the effect of this parameter. For partition $P_{2\times4}$ and $P_{4\times4}$, $\zeta_*$ is increased from 0 to 7 with step 0.5 and from 0 to 15 with step 1, respectively. The experimental results show that, KMSPGA achieves a 100% success rate with different $\zeta_*$. Therefore, $\zeta_*$ has no influence on the solution quality. Fig. 17 shows the influence of the value of $\zeta_*$ on the FEs and a practical execution time of KMSPGA. It can be observed that $\zeta_*$ has negligible influence on the convergence rate. Based on the experimental results, we suggest that the value of $\zeta_*$ is set to $[0.94 \times L \times W]$.

### G. Proof-of-Principle Experiments

To conduct proof-of-principle experiments, we first apply a deterministic deployment strategy to generate $K$ complete cover sets. The superposition of these $K$ complete cover sets results in a Set $K$-cover instance, to which an optimal solution is known. Then KMSPGA and other compared algorithms are applied to solving this instance.

Here, it is to be noted that deterministically deploying the least number of circles to cover any polygon is an NP-hard problem as discussed in [21], [56] and [57]. Nevertheless, if we relax the requirement of that the layout should be “theoretically best with the least circles”, there exists an efficient deterministic node placement strategy to cope with this issue. Fig. 18 shows a “tessellation” placement strategy. First, the target area is completely tiled by a number of compact polygons. Then, sensors are placed at the vertices of polygons. If each polygon is covered by at least one sensor, the

### TABLE VI

<table>
<thead>
<tr>
<th>INSTANCE</th>
<th>$\overline{\mu}$</th>
<th>$\eta$</th>
<th>$\overline{\sigma}$</th>
<th>$\overline{\epsilon}$</th>
<th>$\overline{\tau}$</th>
<th>$\overline{\omega}$</th>
<th>$\overline{\kappa}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>STHGA</td>
<td>157.5003</td>
<td>56796</td>
<td>157</td>
<td>0</td>
<td>1.00</td>
<td>2674</td>
<td>157</td>
</tr>
<tr>
<td>KMSPGA</td>
<td>159.4900</td>
<td>62349</td>
<td>158.9</td>
<td>0.35</td>
<td>0.86</td>
<td>3524</td>
<td>159</td>
</tr>
<tr>
<td>KMSPGA</td>
<td>162.4848</td>
<td>63270</td>
<td>161.9</td>
<td>0.31</td>
<td>0.90</td>
<td>3954</td>
<td>162</td>
</tr>
<tr>
<td>KMSPGA</td>
<td>164.7893</td>
<td>63223</td>
<td>163.9</td>
<td>0.31</td>
<td>0.90</td>
<td>5054</td>
<td>164</td>
</tr>
<tr>
<td>KMSPGA</td>
<td>157.4703</td>
<td>67287</td>
<td>166.9</td>
<td>0.35</td>
<td>0.87</td>
<td>7305</td>
<td>167</td>
</tr>
<tr>
<td>KMSPGA</td>
<td>163.4549</td>
<td>72310</td>
<td>172.5</td>
<td>0.63</td>
<td>0.57</td>
<td>15330</td>
<td>173</td>
</tr>
<tr>
<td>KMSPGA</td>
<td>175.4488</td>
<td>73608</td>
<td>174.1</td>
<td>0.92</td>
<td>0.40</td>
<td>19980</td>
<td>175</td>
</tr>
<tr>
<td>KMSPGA</td>
<td>178.4142</td>
<td>74741</td>
<td>176.4</td>
<td>1.04</td>
<td>0.10</td>
<td>19685</td>
<td>178</td>
</tr>
<tr>
<td>KMSPGA</td>
<td>181.4340</td>
<td>74003</td>
<td>179.9</td>
<td>0.96</td>
<td>0.33</td>
<td>20730</td>
<td>181</td>
</tr>
<tr>
<td>KMSPGA</td>
<td>185.4245</td>
<td>74570</td>
<td>183.4</td>
<td>1.10</td>
<td>0.13</td>
<td>41595</td>
<td>185</td>
</tr>
</tbody>
</table>

Fig. 16. Influence of $\lambda$ on the performance of KMSPGA. (a) I4-2. (b) I5-1.

Fig. 17. Influence of threshold value $\zeta_*$ on the performance of KMSPGA. (a) $P_{2\times4}$. (b) $P_{4\times4}$.

Fig. 18. Illustration of “tessellation” placement strategy, and the polygons adopted here is triangle.
Two tailed t-test of the null hypothesis that GAMDSC is equal to KMSPGA. Two tailed t-test of the null hypothesis that STHGA is equal to KMSPGA.

The performance of KMSPGA. Five new instances are generated in this way with K setting to 100, 300, and 600. The numbers of sensors covering the target area are 45 and 22 when R_s is 5 and 8, respectively.

The experimental results are given in Table VII. Both STHGA and KMSPGA achieve 100% success rate in the tests. However, KMSPGA achieves a higher convergence rate than STHGA. The experimental results also indicate that these ideal and regular test instances are easy to solve by the two algorithms. The reasons are presented as follow. Aiming at using the least number of circles to realize complete coverage, the deterministic deployment strategy tends to minimize the overlapping area of neighboring circles. Thus, for each sensor, the coverage ratio of each cover set will be quite different considering whether or not the sensor is assigned to the right cover set. This feature makes the fitness evolution possess good differentiation for different individuals (candidate solutions) and hence provides a promising guidance for the search. Furthermore, this feature also benefits the proposed redundancy-based schedule strategy, owing to the uniqueness of sensor within each unit set.

H. Experiments with Sensors of Different Radiuses

Although in the above experiments, sensors of identical radius are assumed for simplicity. However, as the proposed KMSPGA algorithm does not contain any radius-related parameters or operators, it is a generic algorithm suitable for both application scenarios with homogenous or heterogeneous sensors deployed. In this subsection, we conduct experiments using sensors of different radiuses to investigate the performance of KMSPGA. Five new instances are generated and tested, in which the radiuses of sensors follow Gaussian distribution with different mean values and standard deviations. The radius of sensor j in instance i is set to R_s+r_j, where r_j is a random number following a standard normal distribution. For i \in \{0, 1, 2\}, R_s is set to 6, 7, and 8, respectively. For i = 3 and i = 4, R_s is randomly chosen within an integer interval [5, 7] and [6, 8], respectively. The number of each instance is 15000.

From the experimental results given in Table VIII, it is still observed that KMSPGA achieves significantly higher solution quality and success rate among the compared algorithms in most of the instances. Besides, KMSPGA achieves smallest standard deviations, which indicates that KMSPGA possesses high stability. The results confirm that KMSPGA works well with sensors of different radiuses.

I. Experiment in a 3-D Environment

In the literature of WSN lifetime maximization, sensors are assumed to be uniformly distributed on an ideal square plane. However, in practice, this is not always the case. Instead, sensors are often deployed on a 3-D surface so that the sensor distribution is no longer uniform, but it is highly dependent to the shape/landscape of the surface [58]. This uneven distribution makes it even harder to optimally schedule the sensors into the right complete cover sets. In this subsection, we implement KMSPGA to test its effectiveness and reliability on this application scene.

In Table IX, the experimental results in 3-D environment on different instances are given. The results show that KMSPGA still obtains the highest quality and success rate among the compared algorithms. Aiming at using the least number of cubes to realize complete coverage, the deterministic deployment strategy tends to minimize the overlapping area of neighboring cubes. Thus, for each sensor, the coverage ratio of each cover set will be quite different considering whether or not the sensor is assigned to the right cover set. This feature makes the fitness evolution possess good differentiation for different individuals (candidate solutions) and hence provides a promising guidance for the search. Furthermore, this feature also benefits the proposed redundancy-based schedule strategy, owing to the uniqueness of sensor within each unit set.

In Table IX, the experimental results in 3-D environment on different instances are given. The results show that KMSPGA still obtains the highest quality and success rate among the compared algorithms. Aiming at using the least number of cubes to realize complete coverage, the deterministic deployment strategy tends to minimize the overlapping area of neighboring cubes. Thus, for each sensor, the coverage ratio of each cover set will be quite different considering whether or not the sensor is assigned to the right cover set. This feature makes the fitness evolution possess good differentiation for different individuals (candidate solutions) and hence provides a promising guidance for the search. Furthermore, this feature also benefits the proposed redundancy-based schedule strategy, owing to the uniqueness of sensor within each unit set.

In Table IX, the experimental results in 3-D environment on different instances are given. The results show that KMSPGA still obtains the highest quality and success rate among the compared algorithms. Aiming at using the least number of cubes to realize complete coverage, the deterministic deployment strategy tends to minimize the overlapping area of neighboring cubes. Thus, for each sensor, the coverage ratio of each cover set will be quite different considering whether or not the sensor is assigned to the right cover set. This feature makes the fitness evolution possess good differentiation for different individuals (candidate solutions) and hence provides a promising guidance for the search. Furthermore, this feature also benefits the proposed redundancy-based schedule strategy, owing to the uniqueness of sensor within each unit set.
Future work includes the development of an improved matching algorithm for the combination operation, of distributed, cloud and multi-objective versions of KMSPGA and their applications to various kinds of real-world problems.

REFERENCES

IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION


Xin-Yuan Zhang (S’14) received the B. S. degree from Sun Yat-sen University, China, in 2014, where he is currently pursuing the Ph. D. degree. His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, their applications in real-world problems, and smart grid.

Jue-Foong Gong (S’10-M’15) received the Ph.D. degree in Computer Science from Sun Yat-sen University, China, in 2014. She is currently a Post-Doctoral Research Fellow with the Department of Computer and Information Science, University of Macau, Macau.

Her research interests include evolutionary computation, swarm intelligence, and their applications to intelligent transportation scheduling, wireless sensor network, and image processing. She has published over 30 papers, including ten IEEE Trans. papers, in her research area. Dr.

**Zhi-Hui Zhan** (S’09-M’13) received the Bachelor’s degree and the Ph. D degree in 2007 and 2013, respectively, from the Department of Computer Science of Sun Yat-Sen University, Guangzhou, China. He is currently an associate professor with the School of Advanced Computing, Sun Yat-sen University.

His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, and their applications in real-world problems, and in environments of cloud computing and big data. Dr. Zhan’s doctoral dissertation was awarded the China Computer Federation Outstanding Dissertation in 2013. Dr. Zhan received the Natural Science Foundation for Distinguished Young Scientists of Guangdong Province, China in 2014 and was awarded the Pearl River New Star in Science and Technology in 2015. Dr. Zhan is listed as one of the Most Cited Chinese Researchers in Computer Science.

**Wei-Neng Chen** (S’07-M’12) received the Bachelor’s degree and the Ph.D. degree from the Department of Computer Science of Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively. He is currently an associate professor with the School of Advanced Computing, Sun Yat-sen University, China.

His current research interests include swarm intelligence algorithms and their applications on cloud computing, financial optimization, operations research and software engineering. He has published more than 30 papers in international journals and conferences. His doctoral dissertation was awarded the China Computer Federation (CCF) outstanding dissertation in 2012.

**Yun Li** (S’87-M’90) received his Ph.D. in computing and control in 1990. He is currently a professor with Department of Systems Engineering, University of Glasgow, U.K. He served as Founding Director of University of Glasgow Singapore during 2011-2013 and acted as Founding Director of the University’s international joint program with University of Electronic Science and Technology of China (UESTC) in 2013. He was invited to Kumamoto University, Japan, as Visiting Professor in 2002 and is currently Visiting Professor to UESTC and Sun Yat-sen University, China, researching into smart design with market informatics via the cloud to complete the value chain for Industry 4.0.

Dr. Li is an Associate Editor of the IEEE Trans. on Evolutionary Computation and of the SM Journal of Engineering Sciences. He has 200 publications and is a Chartered Engineer.

**Jun Zhang** (M’02–SM’08) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002. He is currently a Changjiang Chair Professor with Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, China.

His research interests include computational intelligence, cloud computing, data mining, and power electronic circuits. He has published over 200 technical papers in his research area. Dr. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE Trans. on Evolutionary Computation, IEEE Trans. on Industrial Electronics and IEEE Trans. on Cybernetics.