
Zi-Jia Wang 1, Zhi-Hui Zhan 2,* and Jun Zhang 2

1 School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China; wangzjia@mail2.sysu.edu.cn
2 Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China; junzhang@ieee.org
* Correspondence: zhanapollo@163.com; Tel.: +86-138-2608-9486

Received: 30 October 2018; Accepted: 13 December 2018; Published: 18 December 2018

Abstract: This paper proposed a distributed genetic algorithm (DGA) to solve the energy efficient coverage (EEC) problem in the wireless sensor networks (WSN). Due to the fact that the EEC problem is Non-deterministic Polynomial-Complete (NPC) and time-consuming, it is wise to use a nature-inspired meta-heuristic DGA approach to tackle this problem. The novelties and advantages in designing our approach and in modeling the EEC problems are as the following two aspects. Firstly, in the algorithm design, we realized DGA in the multi-processor distributed environment, where a set of processors run distributed to evaluate the fitness values in parallel to reduce the computational cost. Secondly, when we evaluate a chromosome, different from the traditional model of EEC problem in WSN that only calculates the number of disjoint sets, we proposed a hierarchical fitness evaluation and constructed a two-level fitness function to count the number of disjoint sets and the coverage performance of all the disjoint sets. Therefore, not only do we have the innovations in algorithm, but also have the contributions on the model of EEC problem in WSN. The experimental results show that our proposed DGA performs better than other state-of-the-art approaches in maximizing the number of disjoint sets.

Keywords: wireless sensor networks; energy efficient coverage; distributed genetic algorithm

1. Introduction

Wireless sensor networks (WSN) have become a hot research topic and have been widely used in numerous real-world applications, such as traffic monitoring [1], mobile computing [2], environmental observation [3], and many others [4–6]. In these sophisticated environments, in order to make full coverage and get more accurate results, many nodes should be randomly deployed in the area, causing a waste of resources. Since the sensor nodes are equipped with limited battery resources and the replacement of the battery is not feasible in many applications, low power consumption has become a critical factor to be considered when designing the WSN. Therefore, research into energy saving to prolong the network lifetime has become one of the most significant issues in WSN. Moreover, the energy saving in WSN is a significant research topic in smart and sustainable energy systems and applications [7–9].

Due to the significance of the energy efficient problem in the WSN [10–12], many efforts have been made to tackle this problem. These proposed techniques are generally divided into two categories,
where one is to design efficient protocols to reduce the energy consumption and the other one is to schedule the nodes to work efficiently. For the first category, protocols such as for the medium access control [13], for transmission [14], and for communication [15] have been proposed in the literature for reducing energy consumption. While the second categorized technique concentrates on scheduling the nodes, resulting in energy preservation and longer network lifetime. These two categories both have contributions to the energy efficiency for WSN from different approaches. The focus of this paper belongs to the second category, which is the energy efficient coverage (EEC) problem in the WSN [16,17].

In EEC, the sensor nodes are divided into different disjoint sets with the constraint that each set can guarantee the full coverage of the whole monitored area. In that way, in any time, only the necessary sensor nodes in one set are activated while the other sensor nodes that monitor the same regions can be turned off. These different disjoint sets work one by one and therefore the network lifetime can be prolonged. Moreover, with more disjoint sets that can be formed, longer network lifetime can be obtained because the nodes are scheduled to work energy efficiently [18–20]. Therefore, it is very interesting and promising for the approaches in [21–23] to divide the deployed sensor nodes into maximal disjoint sets and schedule the sets to work in turn. In this paper, we also focus on maximizing the number of disjoint sets.

Even though the above methods have been applied to the EEC problem [21–23], they can easily get trapped into local optima and cannot achieve or guarantee the full coverage because the EEC problem is NPC [21]. Therefore, in this paper, we proposed distributed genetic algorithm (DGA) to solve the EEC problems in WSN to further improve the performance due to the superior adaptation and strong global search ability of GA. Moreover, two major novel designs and advantages of DGA are described as follows:

1. DGA is realized in the multi-processor distributed environment by the master-slave distributed model, where a set of processors run distributed to evaluate the fitness values in parallel to reduce the computational cost.

2. When we evaluate a chromosome, different from the traditional model of EEC problem in WSN that only calculates the number of disjoint sets, a hierarchical fitness evaluation mechanism is proposed and a two-level fitness function with biased attention to the sets with larger coverage percentage is designed.

Therefore, not only do we have the innovations in algorithm, but also have the contributions on the model of EEC problem in WSN. The experimental results show that our proposed DGA performs better than other state-of-the-art approaches in maximizing the number of disjoint sets.

The rest of this paper is organized as follows. Section 2 gives the problem description of the EEC in the WSN and reviews some related work. Section 3 proposes our methodology for solving EEC in the WSN by using DGA. Section 4 presents the experimental results between our approach and other state-of-the-art approaches in the literature. Conclusions are given in Section 5.

2. Energy Efficient Coverage Problem in WSN

The EEC problem is a fundamental and significant research topic in the WSN. In this section, we present the formulation of the EEC and review the related work on this problem.

2.1. Problem Formulation

Given an $L \times W$ (Length $\times$ Width) rectangle area $A$ and $D$ randomly deployed sensor nodes. The EEC is to divide the nodes into several disjoint sets and then schedule these sets to work one by one. As the nodes do not have to work all the time, the energy can be significantly preserved. The aim of the EEC is to maximize the number of disjoint sets, with the constraint that each set can provide the full coverage for the monitored area.

In order to know whether the sensor network provides full coverage in the area $A$, we assume that the location of each sensor node is known in advance [19,24]. Moreover, the area is divided into
many grids and the coverage issue can be transformed to check whether each grid is covered by at least one active sensor node [19].

All the D sensors form the sensors set \( S = \{s_1, s_2, \ldots, s_D\} \), where each sensor node \( s_i \) is with the location \((x_i, y_i)\) and the sensor radius \( R \). For any grid \( g(x, y) \in A \) in the monitored area, the relationship between the \( s_i \) and the \( g \) is defined as:

\[
P(s_i, g) = \begin{cases} 
1, & \text{if } (x - x_i)^2 + (y - y_i)^2 \leq R^2 \\
0, & \text{otherwise}
\end{cases}
\] (1)

where 1 means that the grid \( g \) is covered by the sensor \( s_i \) while 0 means the sensor \( s_i \) does not cover the grid \( g \). Therefore, for any grid \( g \) in the monitored area, if there exist at least one sensor \( s_i (1 \leq i \leq D) \) that makes \( P(s_i, g) = 1 \), we say that \( A \) is fully covered by the sensor network.

In the EEC, the set \( S \) is divided into \( M \) subsets \( S_j (1 \leq j \leq M) \), and with the objective of maximizing the value of \( M \), as:

\[
f = \max K \\
\text{subject to} \\
\text{(i)} \quad \bigcup_{i=1}^{K} S_i \subseteq S \\
\text{(ii)} \quad S_i \cap S_j = \emptyset \\
\text{(iii)} \quad \left( \bigoplus_{s_j \in S} P(s_j, g) \right) = 1, \quad \forall g \in A
\] (2)

Here, the constraint (i) means that the unitization of all the subsets \( S_j \) must belong to the original set \( S \). The (ii) indicates that there is no intersection between any two different subsets \( S_{j1} \) and \( S_{j2} \). The (iii) represents that for any grid \( g \) in area \( A \), there exists at least one sensor \( s_i \) in \( S_j \) which can cover the grid \( g \). Obviously, these constraints can guarantee that each subset \( S_j \) can fully cover the monitored area.

2.2. Related Work

With the development of evolutionary computations (ECs) like GA [25,26], ant colony optimization (ACO) [27,28], particle swarm optimization (PSO) [29,30], and differential evolution (DE) [31,32], many researchers have applied ECs into solving the EEC problems in WSN, such as PSO-based [33] and ACO-based [34] approaches. Specifically, in [33], Zhan et al. extended the binary PSO (BPSO) to solve the EEC problem by finding a minimal set of nodes again and again to maximize the number of disjoint sets. Lin et al. [34] proposed the ACO-based approach to maximize the number of connected covers, called ACO-MNCC, to maximize the lifetime of heterogeneous WSNs by transforming the search space of the problem into a construction graph. Besides, in [35], Yang et al. proposed a probabilistic model to tackle the EEC problem in WSN, which transforms the area converge into point converge. It greatly reduces the dimension of problem. Lee et al. [36] also tried to solve the EEC problem in the WSN by using the ACO, which designs three pheromones to balance the local exploitation and global exploration. Meanwhile, they also introduced a probabilistic sensor detection model, which makes the algorithms more realistic [37].

Some researchers have also used GA for solving the EEC in the WSN recently. For example, Lai et al. [25] proposed a GA-based method to maximize the disjoint sensor covers called GAMDSC. However, there is still much room for improvement. First, their works only solved the small-scale problems (up to 140 sensors). When the scales of problems increase, their performance often face difficulties. Second, parallel or distributed methods can further improve the performance and save the computational time.

3. Methodology: DGA for the EEC Problem

In this section, the DGA for solving the EEC problem in WSN is implemented and described. The master-slave distributed framework is first given. Then the chromosome representation and the hierarchical fitness evaluation mechanism are described. After that, the genetic operators, including the selection, crossover, and mutation are proposed. At last, the overall algorithm is presented.
3.1. Distributed Framework of DGA

In DGA, the master-slave distributed model is used. The processor whose role is to control the whole evolutionary process including crossover, mutation, and selection operations, is called MASTER. The other processors, which are responsible for chromosomes’ evaluation to reduce the computational costs, act as SLAVEs. To give an overview of the DGA, the master-slave distributed structure is depicted in Figure 1. In each generation, the master sends chromosomes to slaves, while the slaves evaluate the chromosomes and return their corresponding fitness values back to the master [38].

![Figure 1](Image)

**Figure 1.** The distributed framework of the distributed genetic algorithm (DGA).

3.2. Chromosome Representation

The aim of the EEC problem is to maximize the number of disjoint sets. Therefore, the first thing is to determine how many sets the nodes can be divided into at most. This is also the upper bound of the disjoint sets number $K$ [23]. Since the area has been divided into many grids, for each grid $i$, we find out the number of sensors $k_i$ that cover the grid $i$. Then, the minimal $k_i$ is the upper bound number of disjoint sets, i.e., $K = \min \{k_i\}$.

With the upper bound $K$ determined, we can encode the chromosome in a very intuitive manner that each gene is an integer in the range of $[1, K]$. Therefore, the chromosome is a string of integers with the length of $D$, where $D$ is the number of sensor nodes. The representation is

$$X = [x_1, x_2, \ldots, x_D], \text{ where } x_i \in \mathbb{Z} \text{ and } 1 \leq x_i \leq K$$

(3)

In (3), the value of $x_i$ denotes which set the $i$th sensor belongs to. Figure 2 gives two examples of the chromosome representation. In the examples, $D$ is 4 and $K$ is 2. In Figure 2a, the chromosome is $X^a = [1, 2, 1, 2]$, meaning that the first and the third sensors belong to the 1st set while the second and the fourth sensors belong to the 2nd set. In this case, it can be observed that both the 1st and the 2nd sets can provide full coverage for the monitored area. However, in Figure 2b, the chromosome is $X^b = [1, 2, 1, 1]$, meaning that the first, the third, and the fourth sensors belong to the 1st set while only the second sensor belongs to the 2nd set. In this case, only the 1st set can provide full coverage for the monitored area, but the 2nd set cannot.

![Figure 2](Image)

**Figure 2.** Illustration of the chromosome representation.
Hierarchical Fitness Evaluation

In order to evaluate the chromosome, herein, a hierarchical fitness evaluation mechanism and a two-level fitness function are proposed.

First, the genes are grouped into different sets according to their values. Then a check procedure is performed on each set to see whether the nodes in the set can provide full coverage. As we divide the monitored area into $L \times W$ grids, for each set $S_i$, we can count the grids number $G$ that is covered by the sensor nodes in set $S_i$. Therefore, we can obtain the coverage percentage $f_i$ as:

$$f_i = \frac{G}{L \times W}$$

Note that if $f_i$ is equal to 1, it means that the set $S_i$ can provide full coverage for the monitored area. In this way, we can count and obtain the number of disjoint sets (denoted as $M$) in the population which can achieve the full coverage. When two chromosomes $A$ and $B$ are compared, $A$ wins $B$ if $A$ has a larger disjoint set number (larger $M$). The number of disjoint sets $M$ is regarded as the first-level fitness evaluation.

However, to meet the case where the two chromosomes $A$ and $B$ have the same disjoint set number, we further designed a fitness function with biased attention to the sets with larger coverage percentage, so called the second-level fitness evaluation, which is calculated as:

$$F = \sum_{i=1}^{K} w_i \times f_i$$

where $K$ is the upper bound of the disjoint sets and $w_i$ is the weight for the set $S_i$. Herein, we give the biased attention to the sets with larger coverage percentage. That is because the set with a higher coverage percentage has a higher probability to achieve the full coverage. Therefore, we first sort the sets according to their coverage percentages in descending order, then the weight $w_i$ for the set $S_i$ is set as $w_i = 10,000.0/r_i$, where $r_i$ is the rank of the set $S_i$ in the sort of coverage percentage.

In that way, if two chromosomes $A$ and $B$ have the same disjoint set number, then the one with a larger fitness value is preferred (larger $F$).

Genetic Operators

The genetic operators in DGA consist of the selection, crossover, and mutation. In this subsection, we will briefly describe the implementations of these operators.

Selection

In the selection implementation, we use the tournament selection strategy. Specifically, in each selection, partial of the chromosomes are randomly selected to compete for survival. The winner enters into the next generation. Repeat the selection operator until a population size of chromosomes have been selected.

Crossover

After the selection, the survived chromosomes recombine to create offspring. First, for each chromosome, a random value in range $[0, 1]$ is generated. If the value is smaller than the crossover probability $p_c$, then the chromosome is used as one of the parents. After all the parents have been determined, every two parents are randomly mated to create two offspring. A crossover position $k$ is randomly generated and the genes from the $k$th position of the two parents are exchanged, as shown in Figure 3a. In this way, two new chromosomes, so called offspring are generated and enter the population.
3.4.3. Mutation

The chromosomes will perform a mutation operation after the crossover. For each gene in a chromosome, a random value in range [0, 1] is generated and compared with the mutation probability \( p_m \). If the random value is smaller than \( p_m \), then the mutation occurs. As shown in Figure 3b, when the mutation occurs, the gene is set as a random integer in the range [1, \( K \)].

![Crossover and Mutation](image)

**Figure 3.** The illustration of the crossover and mutation operators.

3.5. Complete Algorithm

With the designs of the chromosome representation, fitness function, and the genetic operators, the DGA for solving the EEC problem is shown in Figure 4 and is described as the following seven steps.

![Flowchart](image)

**Figure 4.** The flowchart of the DGA for solving the energy efficient coverage (EEC) problem in the wireless sensor networks (WSN).

Step 1: Initialization. The \( N \) chromosomes in the population are initialized at master. Each gene in the chromosome is set as a random integer value in the range [1, \( K \)] as (3).
Step 2: Evaluation. Each chromosome is evaluated as (5) at the slave and returns its fitness to the master. Then the best chromosome with the highest fitness can be determined. This chromosome is compared with the historical best one and will replace the historical best one if it is better. Otherwise, the historical best chromosome will replace the worst chromosome in the current generation.

Step 3: Selection. The selection operation has been discussed in Section 3.4.1.

Step 4: Crossover. The crossover operation has been discussed in Section 3.4.2.

Step 5: Mutation. The mutation operation has been discussed in Section 3.4.3.

Step 6: Termination check. If the termination criteria is met, go to Step 7. Herein, the termination criteria include the obtaining of the upper bound \( K \) or reaching the maximal generation number. Otherwise, the algorithm goes to Step 2 and continues the next generation evolution.

Step 7: Terminate. Output the best chromosome as the final solution.

3.6. Computational Complexity

Herein, we denote the population size and the dimension of problem (the number of sensors) as \( N \) and \( D \), respectively. First, in the initialization, the computational complexity of DGA is \( O(N \times D) \), which is obtained by step 1 in MASTER process of Figure 4. Then, as for the chromosomes evaluation, the computational complexity is \( O(N) \), as obtained by steps 3–4 in the MASTER process and steps 2–4 in the SLAVE process of Figure 4. After that, in the chromosome evolution, since we used the tournament selection, the computational complexity is \( O(N^2 \times D) \), as obtained by steps 6–8 in the MASTER process of Figure 4. Therefore, the overall computational complexity of DGA is \( O(N^2 \times D) \).

4. Experiments and Comparisons

4.1. Algorithms Configurations

The parameter configurations of the DGA are listed as Table 1 and are described as follows.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>40</td>
</tr>
<tr>
<td>Generation</td>
<td>200</td>
</tr>
<tr>
<td>( p_c )</td>
<td>0.8</td>
</tr>
<tr>
<td>( p_m )</td>
<td>0.01</td>
</tr>
</tbody>
</table>

The population size is 40. Several processors are used here to evaluate the fitness of chromosomes. In each tournament selection round, 20% of the chromosomes are randomly chosen and compete for survival. The crossover probability \( p_c \) and mutation probability \( p_m \) are 0.8 and 0.01, respectively. The algorithm terminates at the maximal generations of 200 or it obtains the upper bound \( K \) of the disjoint sets number.

4.2. Experimental Results and Comparisons

In our experiments, we choose several EA-based algorithms, including a PSO-based method (BPSO [33]) and a GA-based method (GAMDSC [25]) to compare with our proposed DGA. To have a reliable and fair comparison, the parameter configurations of all the competitor algorithms are set the same as suggested in their original papers.

Herein, the network topology configurations used in [19] are adopted. That is, with the monitored area 50 m by 50 m, the sensing range \( R \) spans 8 m, 10 m, and 12 m, whilst the deployed nodes number \( N \) can be 100, 150, 200, 250, and 300. For each case with \( N \) original deployed nodes, we randomly deploy the \( D \) nodes in the network to form the topology. Note that the \( D \) nodes must provide the full coverage. Otherwise, the nodes are randomly deployed again until the network is fully covered. On each case, we generate three different random topologies. Therefore, there are totally \( 3 \times 3 \times 5 = 45 \) cases in
this experiment. Due to the stochastic characteristic of the algorithms, we simulate each case on each topology 10 times and the mean results are recorded for statistics. All the approaches are dealing with the same test environments, resulting in a fair comparison.

The simulation results are presented and compared in Table 2. For clarity, the best results are highlighted in boldface. The results show that the DGA outperforms the other approaches on most of the cases. Also, DGA can maximize the number of disjoint sets and obtain the upper bound $K$ in most of the cases, except cases 34, 42–44, while other algorithms can only find few disjoint sets. Moreover, with the increase in the number of nodes, especially in cases 37–45, the superiority of DGA is increasingly obvious, which further indicates the dominance of DGA when solving the complicated EEC problem in WSN.

Table 2. Experimental Results of Comparisons.

<table>
<thead>
<tr>
<th>Case No.</th>
<th>Topology</th>
<th>Upper Bound $K$</th>
<th>All Combinations</th>
<th>Mean Disjoint Set Number $M$</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Node Range</td>
<td></td>
<td></td>
<td>GAMDSCE BPSO DGA GAMDSCE BPSO DGA</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>100 8</td>
<td>2 2$^{100}$</td>
<td>2.000 2.000 2.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>100 10</td>
<td>2 2$^{100}$</td>
<td>2.000 2.000 2.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>100 12</td>
<td>2 2$^{100}$</td>
<td>2.000 2.000 2.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>100 12</td>
<td>4 4$^{100}$</td>
<td>3.900 3.600 4.000</td>
<td>0.025 0.100 0.000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>150 8</td>
<td>2 2$^{150}$</td>
<td>2.000 2.000 2.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>150 8</td>
<td>2 2$^{150}$</td>
<td>2.000 2.000 2.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>150 10</td>
<td>5 5$^{150}$</td>
<td>4.200 5.000 5.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>150 12</td>
<td>3 3$^{150}$</td>
<td>3.000 3.000 3.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>200 8</td>
<td>2 2$^{200}$</td>
<td>2.000 2.000 2.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>200 10</td>
<td>3 3$^{200}$</td>
<td>3.000 3.000 3.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>200 12</td>
<td>5 5$^{200}$</td>
<td>5.000 4.600 5.000</td>
<td>0.000 0.080 0.000</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>200 12</td>
<td>7 7$^{200}$</td>
<td>6.300 6.600 7.000</td>
<td>0.100 0.057 0.000</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>250 8</td>
<td>3 3$^{250}$</td>
<td>3.000 3.000 3.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>250 10</td>
<td>4 4$^{250}$</td>
<td>3.500 3.200 4.000</td>
<td>0.125 0.200 0.000</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>250 12</td>
<td>3 3$^{250}$</td>
<td>3.000 3.000 3.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>300 8</td>
<td>6 6$^{300}$</td>
<td>6.000 6.000 6.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>300 10</td>
<td>7 7$^{300}$</td>
<td>7.000 6.800 7.000</td>
<td>0.000 0.029 0.000</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>300 12</td>
<td>9 9$^{300}$</td>
<td>8.600 8.700 9.000</td>
<td>0.044 0.033 0.000</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>300 12</td>
<td>11 11$^{300}$</td>
<td>9.900 9.700 10.00</td>
<td>0.100 0.118 0.091</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>300 12</td>
<td>12 12$^{300}$</td>
<td>10.80 10.90 11.30</td>
<td>0.100 0.092 0.058</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>300 12</td>
<td>9 9$^{300}$</td>
<td>8.800 8.600 9.000</td>
<td>0.022 0.044 0.000</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>350 8</td>
<td>6 6$^{350}$</td>
<td>6.000 6.000 6.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>350 10</td>
<td>7 7$^{350}$</td>
<td>7.000 6.800 7.000</td>
<td>0.000 0.029 0.000</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>350 12</td>
<td>9 9$^{350}$</td>
<td>8.600 8.700 9.000</td>
<td>0.044 0.033 0.000</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>350 12</td>
<td>11 11$^{350}$</td>
<td>9.900 9.700 10.00</td>
<td>0.100 0.118 0.091</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>350 12</td>
<td>12 12$^{350}$</td>
<td>10.80 10.90 11.30</td>
<td>0.100 0.092 0.058</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>350 12</td>
<td>9 9$^{350}$</td>
<td>8.800 8.600 9.000</td>
<td>0.022 0.044 0.000</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>400 8</td>
<td>6 6$^{400}$</td>
<td>6.000 6.000 6.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>400 10</td>
<td>7 7$^{400}$</td>
<td>7.000 6.800 7.000</td>
<td>0.000 0.029 0.000</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>400 12</td>
<td>9 9$^{400}$</td>
<td>8.600 8.700 9.000</td>
<td>0.044 0.033 0.000</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>450 8</td>
<td>6 6$^{450}$</td>
<td>6.000 6.000 6.000</td>
<td>0.000 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>450 10</td>
<td>7 7$^{450}$</td>
<td>7.000 6.800 7.000</td>
<td>0.000 0.029 0.000</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>450 12</td>
<td>9 9$^{450}$</td>
<td>8.600 8.700 9.000</td>
<td>0.044 0.033 0.000</td>
<td></td>
</tr>
</tbody>
</table>
We also compare the Error values of these approaches when dealing with the EEC problem. The Error is calculated as

\[ Error = (K - M)/K \]  

(6)

where \( K \) is the upper bound in each case and \( M \) is the mean disjoint set number obtained by the approach. The Error values of DGA are smaller than 0.1 in all the cases and most of the Error values are 0, indicating the DGA is very promising in producing high quality solutions.

4.3. Effects of Parameters

The DGA involves two parameters, the crossover probability \( p_c \) and the mutation probability \( p_m \). In this part, we investigate the two parameters based on the same topologies as in Section 4.2.

First, the \( p_c \) is set as 0.8 and the \( p_m \) varies from 0.01 to 0.09. The mean Error values of the 45 cases with different \( p_m \) are plotted in Figure 5. As we can see, the small value of \( p_m \) generally performs better than the large value of \( p_m \). That may because the larger \( p_m \) destroys the convergence ability, which makes the algorithm act like a random search and cannot converge to a promising region. Therefore, the investigated results indicate that the DGA obtains the best performance when the \( p_m \) is set as 0.01.

![Figure 5](image)

**Figure 5.** The mean Error values obtained by the DGA with different \( p_m \).

Then, the \( p_c \) is investigated, with the \( p_m \) set as 0.01. The value of \( p_c \) spans from 0.1 to 0.9, with the step 0.1. The mean Error values of the 45 cases with different \( p_c \) are plotted in the Figure 6. It can be observed that the large value of \( p_c \) may be generally better than the small value of \( p_c \). This may be due to large \( p_c \) can making the good information of the chromosome spread fast in the population. This is very helpful for the algorithm to use good information to search in the promising region, and therefore results in better performance. However, a too-large \( p_c \) value will sometimes make the algorithm converge too fast and be trapped into local optima. For example, when \( p_c = 0.9 \), the mean Error values are worse than that when \( p_c = 0.8 \). In general, \( p_c = 0.8 \) is promising for the DGA to obtain satisfying results.
4.4. Speedup Ratio

Another advantage of DGA is the parallel implementation, where several processors in DGA are used to evaluate the fitness of chromosomes independently. Table 3 shows the computational times of DGA on all the 45 cases with different processors.

Speedup ratio is the metric to measure the performance of a distributed or parallel algorithm, and it varies with the numbers of processors and the computation cost of the case. Herein, we use the small-scale case 1 and the large-scale case 45 as two representative instances. We test the speedup of DGA on these two cases and draw the speedup curve with the increasing processors, from 2 to 10, shown in Figure 7.

As we can see, the speedup ratio consistently grows with the increase of the number of processors on both of these two cases. However, the increase of speedup ratio slows down when the number of processors increases from 6 to 10. That may due to the huge communication cost among many processors. Even so, the speedup ratio still keeps growing with the increasing of the number of processors. Moreover, the speedup ratio of case 45 is larger the case 1. That may because case 45 is more complicated than case 1, which will cause more computation cost than case 1. Therefore, under a similar communication cost, the higher computation cost will directly lead to a higher speedup ratio in case 45.

Figure 6. The mean Error values obtained by the DGA with different $p_c$.

Figure 7. Speedup ratios achieved by DGA on two typical cases.
5. Conclusions

In this paper, DGA is proposed to solve the EEC problem in the WSN to achieve or guarantee the full converge. In the implementation, DGA is realized in the multi-processor distributed environment, where a set of processors run distributed to evaluate the fitness values in parallel to reduce the computational cost. Moreover, when we evaluate a chromosome, different from the traditional model of EEC problem in WSN that only calculates the number of disjoint sets, hierarchical fitness evaluation mechanism is proposed and a two-level fitness function with biased attention to the sets with larger coverage percentage is further designed. Therefore, not only do we have the innovations in the algorithm, but we also have the contributions to the model of the EEC problem in WSN. Simulations have been conducted and the experimental results confirm the effectiveness and efficiency of the proposed approach when compared with the state-of-the-art approaches.
For future work, we wish to further improve the performance of DGA by considering the potential uncertainties, so as to make DGA more applicable in the real applications. Moreover, we wish to apply the DGA into other more complicated optimization problems, like data routing [39], cloud resource scheduling [40], supply chain managing, even in a dynamic and multi-objective environment [41,42], not only in the WSN.

Author Contributions: Z.-J.W. conducted the experiments, performed the experiments, and wrote the draft of this paper. The idea of this paper was proposed by Z.-H.Z., who also made contributions to help organize and write the paper, together with J.Z.

Funding: This work was partially supported by the Outstanding Youth Science Foundation with No. 61822602, the National Natural Science Foundations of China (NSFC) with No. 61772207 and 61332002, the Natural Science Foundations of Guangdong Province for Distinguished Young Scholars with No. 2014A030306038, the Project for Pearl River New Star in Science and Technology with No. 201506010047, the GDUPS (2016), and the Fundamental Research Funds for the Central Universities.

Acknowledgments: The authors would like to thank the assistant editor and reviewers for their valuable comments and suggestions that improved the paper’s quality.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGA</td>
<td>distributed genetic algorithm</td>
</tr>
<tr>
<td>EEC</td>
<td>energy efficient coverage</td>
</tr>
<tr>
<td>WSN</td>
<td>wireless sensor networks</td>
</tr>
<tr>
<td>D</td>
<td>the number of sensor nodes</td>
</tr>
<tr>
<td>M</td>
<td>the number of disjoint subsets</td>
</tr>
<tr>
<td>K</td>
<td>the upper bound of the disjoint sets number</td>
</tr>
<tr>
<td>N</td>
<td>the number of chromosomes</td>
</tr>
<tr>
<td>p_c</td>
<td>crossover probability</td>
</tr>
<tr>
<td>p_m</td>
<td>mutation probability</td>
</tr>
</tbody>
</table>

References


5. Al-Jaoufi, M.A.A.; Liu, Y.; Zhang, Z. An active defense model with low power consumption and deviation for wireless sensor networks utilizing evolutionary game theory. *Energies* 2018, 11, 1281. [CrossRef]


