Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 172783, 12 pages
http://dx.doi.org/10.1155/2013/172783

Research Article

Differential Evolution for Lifetime Maximization of Heterogeneous Wireless Sensor Networks

Yulong Xu,[1,2] Jian'an Fang,[1] Wu Zhu,[1,3] and Wenxia Cui[1]

[1] *College of Information and Technology, DongHua University, Shanghai 201620, China*
[2] *Institute of Information and Technology, Henan University of Traditional Chinese Medicine, Zhengzhou 450003, China*
[3] *Office of Putuo District, Ganquan Road Subdistrict, Shanghai 200065, China*

Correspondence should be addressed to Yulong Xu; flyxyl@126.com

Received 9 January 2013; Revised 1 February 2013; Accepted 12 February 2013

Academic Editor: Yang Tang

Maximizing the lifetime of wireless sensor networks (WSNs) is a hot and significant issue. However, using differential evolution (DE) to research this problem has not appeared so far. This paper proposes a DE-based approach that can maximize the lifetime of WSN through finding the largest number of disjoint sets of sensors, with every set being able to completely cover the target. Different from other methods in the literature, firstly we introduce a common method to generate test data set and then propose an algorithm using differential evolution to solve disjoint set covers (DEDSC) problems. The proposed algorithm includes a recombining operation, which performs after initialization and guarantees at least one critical target's sensor is divided into different disjoint sets. Moreover, the fitness computation in DEDSC contains both the number of complete cover subsets and the coverage percent of incomplete cover subsets. Applications for sensing a number of target points, named point-coverage, have been used for evaluating the effectiveness of algorithm. Results show that the proposed algorithm DEDSC is promising and simple; its performance outperforms or is similar to other existing excellent approaches in both optimization speed and solution quality.

1. Introduction

The past two decades have witnessed the boom of wireless sensor networks (WSNs) and their applications, such as battlefield surveillance, environment supervision, traffic control, animal tracking, and home applications [1–5]. A key technology for various applications that involve long-term and low-cost monitoring, in other words, the fundamental criterion for evaluating a WSN, is the network lifetime [6], which is defined as the period that the network satisfies the application requirements. Since most devices of WSNs are powered by nonrenewable batteries, studies for prolonging the network lifetime have become one of the most significant and challenging issues in WSNs. Thus, how to maximize the network's lifetime is a critical research topic in WSNs.

Various methods have been proposed for prolonging the lifetime of WSNs focusing on the issues of data processing [7], routing [8–11], device placement [12], topology management [13], and device control [14–16]. In a WSN where devices are densely deployed, a subset of the devices can already address the coverage and connectivity issues. Since it has been proven that the network connectivity of active sensors in complete coverage is guaranteed by having the communication range of each sensor at least twice of its sensing range [17, 18], thus, only the sensing coverage problem is researched in this paper.

In a WSN, coverage problem is an important issue, which determines how well an area of interest is monitored or tracked by sensors, and the device control approach that schedules the devices' sleep/wakeup activities has shown to be promising [19, 20]. In general, random deployment is performed to place the sensors (dropping from a plane) to the target area. And a sensor generally has two operation modes, active mode and sleep mode [4]. When in active mode, a sensor can carry out its full operations, such as sensing, computation, and communication. To maintain those operations, sensors need to consume a relatively large amount of energy. In contrast, a sensor in a sleep mode uses only a small amount of energy and can be awoken in

a scheduled working interval for full operations. When a subset of sensors in the area can already cover the target area completely, the other sensors can be scheduled to be in the sleep mode to save energy; thus, if the number of subset is maxed, the lifetime of WSN is prolonged. In other words, maximizing the number of completely cover subset is a more direct way to maximize the network lifetime [21]. Note that the target includes point target and area target; so the coverage problem is divided into point-coverage and area-coverage, respectively. Because the point target is the most common phenomenon in practical application, in this paper, the point-coverage problem is considered to maximize the lifetime of WSN by maximizing the number of completely cover subsets.

The problem of finding the maximum number of complete cover subsets is difficult because each subset must fulfill complete coverage to the target area, and the WSN can satisfy the surveillance task with only one subset of sensors active at any time. This problem in a WSN is called the disjoint set covers problem or the SET k-cover problem, which has been proven to be nondeterministic polynomial complete (NPC) complex problem [22]. In [22], a maximum cover using mixed integer programming (MC-MIP) algorithm is proposed to find the maximum number of complete cover subsets. In [23], Slijepcevic and Potkonjak proposed a greedy algorithm, which is named the most constrained minimally constraining covering (MCMCC) heuristic to completely cover the target area. In [24], Lai et al. firstly introduce gene algorithm to solve the point-coverage problems for WSN by finding the maximum number of complete cover subsets, termed GAMDSC. Recently, a novel hybrid genetic algorithm using a forward encoding scheme (STHGA) was proposed by Hu et al. [25] for lifetime maximization of WSN, which adopts a forward encoding scheme for chromosomes and sensor schedule transition operations.

Besides, from other perspectives to research wireless sensor networks such as evolutionary algorithm (EA) [26, 27] are promising direction. Furthermore, using other gene algorithms to research this problem is worth considering. Among the popular gene algorithms and their application in the recent years [28–31], differential evolution (DE) algorithm [32, 33] appears to be a competent candidate. Due to its original definition by Storn and Price [34], the DE algorithm and it's variant are perceived as a reliable and versatile population-based heuristic optimization technique, which exhibits remarkable performance in a wide variety of application problems [35–40].

In this paper, a DE-based algorithm is presented for aiming at solving disjoint set covers problems for maximizing the WSN lifetime. The proposed algorithm, termed the differential evolution for solving disjoint set covers (DEDSC) problems, comprises the following features. Firstly, before running of DEDSC, test data set is produced by a common generation method, which includes the targets and sensors coverage information and the upper bound of disjoint set covers number ($C_{\max}$). And then, at initialization of DEDSC, the population is generated randomly; in each chromosome the value of gene is a random integer between $[1, C_{\max}]$. After that, a recombination operation is performed to guarantee at least one critical target's sensor is divided into different disjoint subsets. Fourthly, there are just two parameters in DEDSC, and the mutation strategy of DEDSC is "DE/best/1" differential evolution; the round number style is ceiling. Lastly, the fitness computation in our proposed algorithm considers both the number of complete cover subset and the coverage percent of incomplete cover subset. In general, these distinct features of DEDSC make it simple and effective to implement.

The rest of this paper is organized as follows: Section 2 defines the problem addressed in this paper. Section 3 describes the implementation of the proposed algorithm in detail and includes the encoding method of chromosomes, the design of the fitness function, the recombining operation, the crossover operation, and mutation operations. Experimental results and discussions are presented in Section 4. Finally, Section 5 draws a conclusion and provides guidelines for future research.

2. Disjoint Set Covers Problem

In this section, the problem of finding the maximum number of disjoint set covers in WSN is defined. And then we introduce a method for estimating an upper bound of the number of disjoint set covers.

2.1. Problem Definition. Suppose that, in an $L \times W$ area, there are have a set of targets $T = \{t_1, t_2, t_3, \ldots, t_m\}$, and then randomly deploy a set of sensors $S = \{s_1, s_2, s_3, \ldots, s_n\}$ in this area to monitor the targets. All of the sensors have sleep mode and active mode: in the active mode sensors can sense information of target, and assume sensors have the same sensing region, but in sleep mode they cannot sense due to saving energy. A target is said to be covered by a sensor if it lies within the sensing region of the sensor. In order to prolong the lifetime of WSN, we need find the maximum number of disjoint sensor covers. The problem can be solved via transformation to the DSC problem [22], which is defined that to find the maximum number C of disjoint complete cover sets, and the corresponding cover set C_i is satisfied [24, 25].

Every cover C_i is a subset of $S, i \in [1, C_{\max}]$, where the $C_{\max}$ is the upper bound of disjoint set covers number C. Namely, $C_i \in S$, and each C_i can complete coverage to all of the targets. Beyond that, for every of t_i belongs to at least one member of C_i, and for any two different covers C_i and C_j, $C_i \cap C_j = \phi$. Take Figure 1 as an example; there are five sensors S1, S2, S3, S4, and S5 and four targets t1, t2, t3, and t4; every sensor has the same circular sensing region. Note that in the real applications, sensing region of a sensor can be an irregular shape [41].

According to Figure 1 it can be transformed into bipartite graph of Figure 2. The link line between sensor S_i and target t_j means S_i can sense t_j. For each sensor, which can sense targets set $S_1 = \{t_1, t_3\}$, $S_2 = \{t_1, t_2\}$, $S_3 = \{t_2, t_4\}$, $S_4 = \{t_3, t_4\}$, and $S_5 = \{t_1\}$.

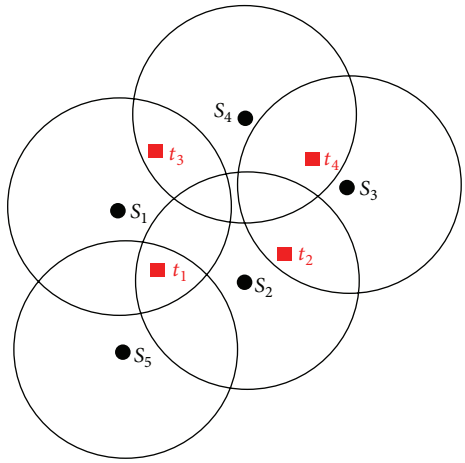

Figure 1: A randomly deployed figure of WSNs by five sensors and four targets.

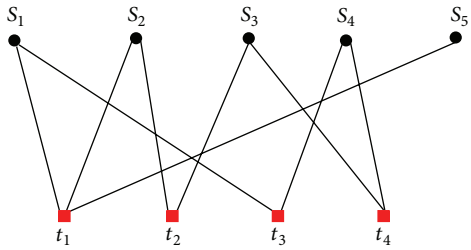

Figure 2: The bipartite graph of Figure 1.

2.2. Upper Bound of Disjoint Set Covers Number C. In WSNs, the maximum number of disjoint set covers cannot exceed the maximum number of full cover subsets that satisfy the coverage constraint. Thus, the maximum number of full cover subsets ($C_{\max}$) can be used as the upper limit of the number of disjoint set covers. It has been proven that finding the maximum number of full cover subsets is an NPC problem [21, 42], but we can compute the upper bound of the number of full cover subsets with the following method in test data sets.

When all the deployed sensors are active, all the targets are covered. For each target, it can be covered by at least one sensor. The targets which are covered by the minimum number of sensors is named *critical targets*, and their corresponding sensors are named *critical sensors*. It is worth noting that in the WSNs there maybe exists one or many *critical targets*, and each *critical target* has its own corresponding *critical sensors*. In the literature [24], the authors proposed term *critical sensors* which is just a special case; namely, there is just one *critical target* in WSNs. The number of sensors covering a *critical target* can be estimated by the upper bound of the number of full cover subsets [23, 25]. Therefore, the minimum number of sensors covered, which is denoted by $C_{\max}$, can be used as the upper bound of C.

Table 1: Notations.

S	Set of sensors deployed in the target area
S_i	Sensor i
T	Set of targets
t_i	Target i
C_i	The ith cover subset of S
C	The number of complete cover subset of S
C'	The number of incomplete cover subset of S
$C_{\max}$	The upper limit of C
CH_i	The ith chromosome
R	Sensing range
p	Average coverage percentage of incomplete cover subset
m	The number of sensor
n	The number of target
popsize	Size of population
pop	Current population
popold	Parent population
F	Factor of mutation
C_r	Crossover rate

Take the case in Figures 1 and 2 as an example; for these targets, the covered sets by sensors are $t_1 \in (S_1, S_2, S_5)$, $t_2 \in (S_2, S_3)$, $t_3 \in (S_1, S_4)$, and $t_4 \in (S_3, S_4)$. According to above analysis, there are three *critical targets* (t_2, t_3, and t_4), and each *critical target* is covered by 2 sensors; thus the upper of the C is 2 ($C_{\max} = 2$).

3. Differential Evolution for Maximizing the Disjoint Set Covers Problems

In this section, we will describe the DEDSC approach for maximizing the number of disjoint set covers problems in point-coverage targets WSN. Firstly, the test set generation algorithm is given. Then the implementation of the proposed DEDSC is presented step by step, including the initialization, the evaluation of the population, the recombination operations, and the crossover, mutation, and selection operations. For easy understanding, the notations used in this paper are shown uniform in Table 1.

3.1. The Test Data Set Generation Algorithm. According to the application of WSN in the real-world, sensors and targets are randomly deployed in the specific area. Thus the test data set needs to be randomly generated by algorithm; however, in all of the above literature [23–25], the test data set generation algorithm is not introduced. Therefore, a general test data set generation algorithm is given in this subsection. The pseudocode of this algorithm is shown in Pseudocode 1. Note that the code is described by Matlab language, and the bold words are inner functions or keywords in Matlab. For example, the **cell** is the cell structure type.

The test data set is obtained after running this algorithm, which contains the sensing information of every sensor (Scell), the information of every target covered (Tcell), the

```
/* generate test data set algorithm */
(1) randomly generate m sensors S_i, each S_i has abscissa S_ix
    and ordinate S_iy.
(2) randomly generate n targets t_j, each t_j has abscissa t_jx
    and ordinate t_jy.
(3) Scell = cell(m, 1); % save every sensor can sense
    the targets
(4)  for i =1: m
(5)    for j = 1: n
(6)      if sqrt((S_ix − t_jx)^2 + (S_iy − t_jy)^2) ≤ R
(7)            St{i} = [St{i} j]; % sensor i can sense
                 target j
(8)      end if
(9)    end for
(10) end for
/* compute upper of disjoint complete cover sets (C_max) */
(11) Tcell = cell(n, 1)
(12) for j = 1: n
(13)    for i = 1: m
(14)      if (find(Scell{i} == j))
(15)         count++;
(16)         Tcell{j} = [Tcell{j} i]; % target j is covered
              by sensor i
(17)      end
(18)    end
(19)    c = [c count];
(20) end
(21) C_max = min(c);
/* count all of critical targets and corresponding sensors */
(22) for t = 1: n
(23)    if C_max == size(Tcell{t}, 2)
(24)      CritialTarget = [CritialTarget Tcell{t}];
(25)      numOfCritialTarget++;
(26)    end
(27) end
(28) CritialTargetMatrix = reshape(CritialTarget);
```

Pseudocode 1: Pseudocode of generated test data set algorithm.

matrix of all *critical targets* and their corresponding critical sensors (Critial Target Matrix), and the upper limit of the number of disjoint set covers ($C_{\max}$).

3.2. The Representation of Chromosomes. It is known that 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 subset that the sensor joins [24], and thus the sensors with the same index number form a disjoint cover set.

For example, suppose a chromosome is CH_1 = (2, 1, 2, 3, 1, 3, 2, and 1). It means that there are eight sensors and set 1, set 2, and set 3 three disjoint subsets in WSN. The set 1 contains sensor S_2, S_5, and S_8, set 2 contains sensor S_1, S_3, and S_7, and set 3 contains S_4, S_6. If each of the above three sets can completely cover all targets, it means that the number of disjoint set cover is 3. In application, the sensors with the gene

```
/* recombination operation */
(1) for i = 1: popsize % for every chromosome
      % randomly generates C_max different integers and
      each integer value belong [1, C_max].
(2)   sequ = randperm(C_max);
      % randomly choose one critical target
(3)   rnum = randint(1,1,[1, numOfCritialTarget]);
      % find the corresponding sensors
(4)   rt = CritialTargetMatrix(rnum,:);
(5)   for j = 1: C_max % all corresponding sensors
(6)     pop(i,rt(j)) = sequ(j); % recombination
(7)   end for
(8) end for
```

Pseudocode 2: The pseudocode of recombination operation.

value 1 are scheduled to be activated at first, the other sensors will be kept in a sleep mode until the second set of sensors is activated, and the sensors in set 3 are activated in the last scheduling.

3.3. Initialization and Recombination. According to the representation of chromosomes, we randomly generate an integer between 1 and $C_{\max}$ as each gene value (subset number) in the initialization. The sensors with same gene value mean they form a subset. Take Figure 1 as an example, there are five sensors, and $C_{\max}$ = 2; therefore after initialization a chromosome maybe is CH_2 = (2, 1, 1, 1, 2), each gene value is random, and sensors S_2, S_3, and S_4 form subset 1; sensors S_1, S_5 form subset 2.

After initialization a recombination operation is performed, which guarantees at least one critical target's sensor is divided into different disjoint sets (subsets). Note that the recombination operation is different from the scattering in paper [24]; in this literature the authors assume there is only one *critical target* and scatter the corresponding sensors into different subsets. However, there maybe exist many *critical targets* in WSNs, and it is very difficult to scatter all of corresponding sensors into different subsets, this is likely an NP problem. Thus, our recombination operation considers every chromosome, randomly chooses a *critical target*, and recombines different gene values to their corresponding sensors. Take the chromosome CH_2 as an example; after initialization the chromosome CH_2 = (2, 1, 1, 1, and 2); then doing recombination operation, according to above analysis in Section 2.2, there are three *critical targets* (t_2, t_3, and t_4), which $t_2 \in (S_2, S_3)$, $t_3 \in (S_1, S_4)$, and $t_4 \in (S_3, S_4)$. Thus randomly choose one *critical target* from t_2, t_3, and t_4, suppose the target t_4 is chosen, and its corresponding sensors are scatted into different subset; one case is S_3 = 2, S_4 = 1. Namely, the chromosome CH_2 became CH_2 = (2, 1, 2, 1, 2); it is obvious that after recombination the CH_2 has better fitness. The pseudocode of recombination is given in Pseudocode 2. It needs to explain that the code is described by Matlab language, and the bold words are inner functions or keywords in Matlab.

3.4. Objective Function. The objective fitness function of a chromosome CH_i in the population is defined as follow:

$$f_i = C + p, \tag{1}$$

where C ($C \geqslant 1$) is the number of disjoint complete cover sets and p ($p \in (0, 1)$) is the average coverage percentage of incomplete cover subset.

Actually, it is not difficult to calculate the average coverage percentage of incomplete cover subset in the point-coverage problem. Assume in a chromosome, the number of disjoint complete cover is C, and then the number of incomplete cover subset is $C'(C_{\max} - C)$. For the ith incomplete cover, if it can cover k targets, then the average coverage of ith incomplete cove subset is k/n.

3.5. Differential Evolution for DSC Problem. Combined with the above analysis, this subsection introduces the detail of using differential evolution to solve DSC problem. Differential evolution [34] is a novel evolutionary algorithm, which has the following different mutation strategies which are frequently used in the literature:

(1) "DE/rand/1"

$$v_{i,g} = x_{r1,g} + F_i \cdot \left(x_{r2,g} - x_{r3,g}\right), \tag{2}$$

(2) "DE/best/1"

$$v_{i,g} = x_{\mathrm{best},g} + F_i \cdot \left(x_{r1,g} - x_{r2,g}\right), \tag{3}$$

(3) "DE/current-to-best/1"

$$v_{i,g} = x_{i,g} + F_i \cdot \left(x_{\mathrm{best},g} - x_{i,g}\right) + F_i \cdot \left(x_{r1,g} - x_{r2,g}\right), \tag{4}$$

where the indices r_1, r_2, and r_3 are distinct integers uniformly chosen from the set $\{i = 1, 2, \ldots, \mathrm{NP}\} \setminus \{i\}$, $(x_{r1,g} - x_{r2,g})$ is a difference vector to mutate the corresponding parent $x_{i,g}$, $x_{\mathrm{best},g}$ is the best vector at the current generation g, and F_i is the mutation factor which usually ranges on the interval $(0, 1)$. In classic DE, $F_i = F$ is a fixed parameter used to generate all mutation vectors at all generations, while in many adaptive DE algorithms each individual i is associated with its own mutation factor F_i.

Following the mutation stage, a binomial crossover operator is implemented to enhance diversity of the population. In crossover operation, the target vector $x_{i,g}$ is connected with its corresponding mutant vector $v_{i,g}$ by using binomial crossover to generate a trial vector $u_{i,g}$. The aforementioned scheme can be outlined in

$$u_{i,G}^{j} = \begin{cases} v_{i,g}^{j}, & \text{if } \mathrm{rand}_{i,j}[0, 1] \leq C_r \text{ or } j = j_{\mathrm{rand}}, \\ x_{i,g}^{j}, & \text{otherwise.} \end{cases} \tag{5}$$

The $\mathrm{rand}_{i,j}[0, 1]$ is a uniformly distributed random number with the range $[0, 1]$, different for each j. C_r denotes a user-defined parameter, called crossover probability, which determines how similar the trial vector will be with respect to the mutant vector. $j_{\mathrm{rand}} \in [1, D]$ indicates a randomly selected integer which ensures at least one dimension of the trial vector $u_{i,g}$ will differ from its associated target vector $x_{i,g}$. It has been proven that large values of C_r improve variation and hence accelerate convergence speed, while small values promote exploitation.

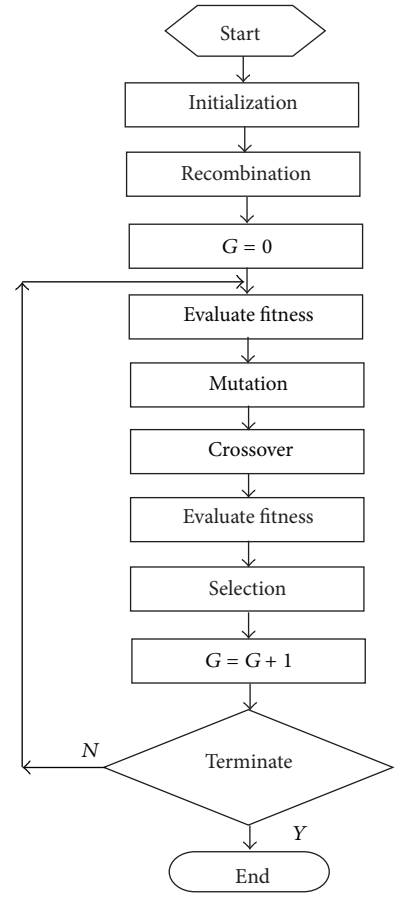

Figure 3: Flowchart of the proposed DEDSC.

Finally in selection operation, the selection operator is a one-to-one spawning strategy. In selection, the target vector competes against the trial vector according to their fitness values, and the better one will be selected to enter the next generation:

$$x_{i,g+1} = \begin{cases} u_{i,g}, & \text{if } f\left(u_{i,g}\right) \leq f\left(x_{i,g}\right), \\ x_{i,g}, & \text{otherwise.} \end{cases} \tag{6}$$

For using DE to solve the disjoint set cover problem systemically and concisely, we provide the flow chart and the pseudocode of basic DEDSE in Figure 3 and Pseudocode 3, respectively. Figure 3 contains all operation modules, and Pseudocode 3 shows the corresponding pseudocode.

From Figure 3 and Pseudocode 3, we summarize that the distinct features of DEDSC are as follows. Firstly, test data set is produced by generation algorithm before running DEDSC, and the overall ideal of DEDSC is differential evolution

```
/* proposed DEDSC algorithm */
(1) Load test data set.
(2) Initialization parameters F, C_r, et. al
    % Initialization population
(3) popold = round(unifrnd(1, C_max, popsize, m));
(4) Recombination operation
(5) FES = 0; % the generation of evolution
(6) while FES < m * 100
(7)    pop = popold;
(8)    v_{i,g} = x_{r1,g} + F_i · (x_{r2,g} − x_{r3,g}) % mutation
(9)    for i = 1: popsize
(10)       for j = 1: m
               v_{i,g}(j) = ceil(v_{i,g}(j)); % round numbers
(11)           if v_{i,g}(j) < 1 or v_{i,g}(j) > C_max % boundary
                 checkout
                       % regenerate a random integer
                       % between [1,C_max] to update v_{i,g}(j)
(12)             v_{i,g}(j) = randint(1,1,[1, C_max]);
(13)           end
(14)       end
(15)   end
(16)   Crossover between v_{i,g} and x_{i,g} by (5)
       % Evaluate fitness for x_{i,g} and u_{i,g}
(17)   for i = 1: popsize % for each chromosome i
(18)      for j = 1: C_max
(19)         find those sensors with same subset index j.
(20)         for every sensors k.
(21)            count each k cover the targets and
                  union them.
(22)         end
(23)         if the number of targets z is equal n
(24)            this subset j can complete coverage;
(25)         else
(26)            this subset j coverage percentage is z/n;
(27)         end
(28)      end
(29)      count fitness of chromosome i by (1);
(30)   end
(31)   Selection operation by (6);
(32)   Update popold;
(33)   FES = FES + popsize; % next generation
       % Judgment of terminal condition
(34)   if FES ≥ m * 100 or the best fitness equals C_max
(35)      break;
(36)   end
(37) end
```

Pseudocode 3: The pseudocode of proposed DEDSC algorithm. Note that the pseudo code is described by Matlab language, and the **bold** words are inner functions or keywords in Matlab.

algorithm. Second is initialization, the population is generated randomly, and thus the value of genes in chromosome is a random integer between 1 and $C_{\max}$. After that, a recombination operation is performed to guarantee at least one critical target's sensor is divided into different disjoint subsets.

Fourthly, there are just two parameters in DEDSC, and the mutation strategy of DEDSC is "DE/best/1" differential evolution; the round numbers style is ceiling. In addition, the fitness computation in our proposed algorithm considers not only the number of complete cover sets but also the coverage percent of incomplete cover sets. Lastly, termination condition of DEDSC is that the best fitness in current population equals $C_{\max}$ or the generation of evolution greater than $m * 100$.

3.6. Runtime Complexity Analysis of DEDSC Algorithm. Runtimecomplexity analysis of the population-based stochastic search techniques, like DE, GA, and so forth, is a critical issue by its own right. In this section, we examine the computational complexity of the proposed DEDSC algorithm, which can be estimated based on the number of calculations of metrics and reproduction that are required.

Assuming NP is the number of individuals, namely, size of population, and D is the dimension of problem; the serial DE model needs $O(\mathrm{NP} \cdot D)$ time algorithm for upgrading population and computing values and fitness object function. Hence, if the algorithm is terminated after a fixed number of generations $G_{\max}$, the overall runtime is $O(\mathrm{NP} \cdot D \cdot G_{\max})$.

In DEDSC, besides the fundamental operations for two parts of vectors, we have to take into account both of them. In our DEDSC, a recombination operation after initialization is proposed, which just performsone time in every time run. For each chromosome in NP, the best and the worst run complexity is $O(1)$ and $O(D)$, respectively. Thus, the average complexity of recombination is $O(0.5 \cdot (1+D) \cdot \mathrm{NP})$. Therefore, the time complexity of DEDSC is $O(0.5 \cdot (1+D) \cdot \mathrm{NP}) + O(\mathrm{NP} \cdot D \cdot G_{\max})$.

The above formula reveals the proposed DEDSC is slightly higher than original algorithm on runtime complexity.

4. Simulation Experiment and Analysis

In this section, a series of experiments are performed to evaluate the performance of DEDSC. Since the proposed approach is for maximizing the number of disjoint sets cover in point-average WSNs, the state-of-the-art algorithms, that is, GAMDSC [24] and STHGA [25], are used for comparison.

4.1. Parameter and Test Environment Setting. It needs to explain that, if not specially stated, the experiments for DEDSC use the same parameters settings as the population size *popsize* = 10; the rate of mutation and crossover are initialized $F = 0.5$ and $C_r = 0.3$, respectively, in which value of F and C are empirical values. And in DEDSC, the mutation strategy is DE/best/1, the maximum number of evolution generation function evaluations (FES) is $m \times 100$, and if the number of disjoint complete cover sets reaches $C_{\max}$, the algorithm also terminates. These parameters influence the performance of DEDSC, which will be analyzed in Sections 4.3 and 4.4.

Parameter settings of GAMDSC and STHGA can be referred in [24, 25]. For DEDSC and other referred algorithms, each case is tested 100 times independently, the sensors are deployed in a 50×50 rectangle area, and the coordinates

Table 2: The generated test cases index 1–7.

Case index	Sense radius of sensor	Number of sensor	Number of target	C_{max}
1	22	90	10	30
2	22	100	10	23
3	22	110	10	21
4	22	120	10	35
5	22	130	10	41
6	22	140	10	44
7	22	150	10	42

of sensors' locations are randomly generated as float-point values in [0, 50]. All cases are run by a computer with a Core I3 2.8 GHz CPU. We use the proposed generated test data set algorithm to obtain the test cases 1–7 and show the details of these test cases in Table 2.

4.2. Overall Test for DEDSC and Referred Algorithm. In this subsection, the overall test for DEDSC and comparison of referred optimization algorithms are provided in Table 3. The best result among those methods is shown by Boldface in the table. The "ok%" indicates the percent to find C_{max} in 100 runs, "Best" is the best result in algorithm, "Mean" indicates the mean value of 100 runs, and the "avg$E$" denotes the average of evolution generation when finding C_{max}.

From Table 3, it can be observed that the proposed DEDSC is significant improvement rather than GAMDSC in cases 1-2 and 4–7; in case 3 they have equal performance. Moreover, the proposed DEDSC algorithm almost has the same performance as the STHGA, and it can reach a hundred percent to find C_{max} in all cases except for case 6. However, the result shows that our proposed DEDSC need the least evolution generation to find C_{max} in cases 1–5 and case 7, which implies the DEDSC has the most fast convergence rate.

In order to analyze the performance of DEDSC detailed, the average optimization curves of DEDSC in cases 1 and 6 are provided in Figures 4 and 5, respectively. From Figures 4 and 5 we can conclude that (1) DEDSC finds the best result much faster than other two methods which benefit from differential evolution and the comprehensive evaluation of fitness; (2) due to the recombination operation, DEDSC has the best initial population. In general, the traditional intelligence method, that is, Genetic Algorithm, has better initial population than STHGA, that is because the GA randomly generates group number, but STHGA generates index increase by degrees. It is can be seen from Figures 4 and 5 STHGA has the least number of complete cover sets in initialization; (3) the performance of STHGA, stable progressive increase, and it easily finds global optima because of the forward encoding scheme. DEDSC has better convergence rate and initial population than STHGA. However, in the later stage of evolution it may fall into the local optimal solution. That is, the reason for our proposed DEDSC is not better than STHGA in case 6.

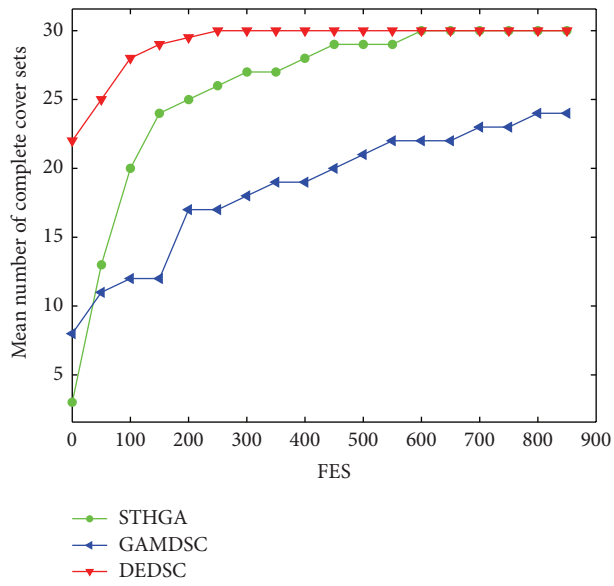

Figure 4: Average optimization curves of DEDSC, STHGA, and GAMDSC in case 1. Each of 50 generations evaluates one time.

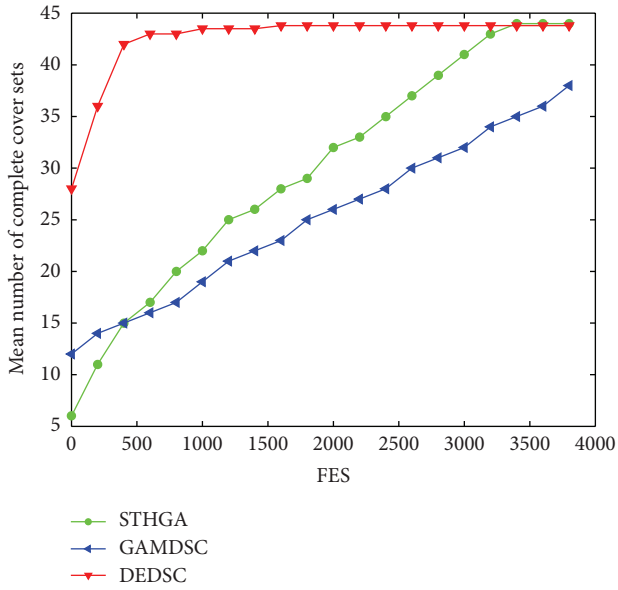

Figure 5: Average optimization curves of DEDSC, STHGA, and GAMDSC in case 6. Each of 20 generations evaluates one time.

4.3. Test for the Method of Round Numbers. In this subsection, the different way of round numbers and mutation are tested. The used test data is cases 1 to 7 are shown in Table 2.

After mutation operation, the value of gene maybe is float type, but the index of subset is integer type; therefore, it needs to do round number operation for population. There are there popular ways for round number operation, ceiling, floor, and round. In order to find which method is more suitable for DEDSC, all of three ways are tested and compared in Table 4.

Table 4 reports the experimental results of the three methods for round numbers in 100 independent runs. And

Table 3: Experimental results of cases 1–7, averaged over 100 independent runs.

	DEDSC				GAMDSC				STHGA			
	ok %		Mean	avg E	ok %	Best	Mean	avg E	ok %	Best	Mean	avg E
1	**100**	30	30	**1.24e + 02**	8	30	28.63	1.83e + 04	**100**	30	30	5.96e + 02
2	**100**	23	23	**10.1**	84	23	22.83	1.19 + 04	**100**	23	23	2.14e + 02
3	**100**	21	21	**10**	**100**	21	21	6.14e + 03	**100**	21	21	1.56e + 02
4	**100**	35	35	**3.71e + 01**	5	35	33.5	1.98e + 04	**100**	35	35	4.22e + 02
5	**100**	41	41	**4.01e + 02**	99	41	40.99	1.03e + 04	**100**	41	41	1.86e + 03
6	87	44	43.93	**5.84e + 02**	0	43	40.72	2.01e + 04	**100**	44	44	3.57e + 03
7	**100**	42	42	**4.24e + 01**	24	42	40.82	1.90e + 04	**100**	42	42	5.32e + 02

Table 4: Experimental results of cases 1–7, averaged over 100 independent runs.

	Ceiling				Round				Floor			
	ok %	Best	Mean	avg E	ok %	Best	Mean	avg E	ok %	Best	Mean	avg E
1	100	30	30	1.44e + 02	100	30	30	**1.22e + 02**	100	30	30	1.33e + 02
2	100	23	23	**10.1**	100	23	23	1.02e + 01	100	23	23	1.02e + 01
3	100	21	21	10	100	21	21	1.00e + 01	100	21	21	1.00e + 01
4	100	35	35	3.07e + 01	100	35	35	3.05e + 01	100	35	35	**3.03e + 01**
5	**99**	41	40.99	**3.33e + 02**	97	41	40.97	4.11e + 02	98	41	40.99	3.62e + 02
6	**87**	44	43.93	5.96e + 02	85	44	43.91	5.72e + 02	83	44	43.92	**5.22e + 02**
7	100	42	42	**3.84e + 01**	100	42	42	5.00e + 01	100	42	42	5.32e + 01

these tests are used in mutation strategy "DE/rand/1"; parameter value are $F = 0.5$, $C_r = 0.3$. The best result among those methods is indicated by Boldface in the table. The results show that different round number methods have different performances. Specifically, with ceiling methods the performance and convergence rate of DEDSC is better than other methods in all cases especially in cases 5 and 6; thus as a conclusion, the ceiling round number methods are applied suitably in our DEDSC algorithm.

4.4. Test for Mutation Strategy and F, C_r. There are many mutation strategies in DE, such as "DE/rand/1," "DE/best/1," and "DE/current-to-best/1." Meanwhile there are two important parameters F and C_r in DE. Therefore, finding the suitable mutation strategy and F, C_r are significant to the performance of DEDSC. In this subsection, we apply many popular mutations and F, C_r for testing DEDSC. The used test data sets are cases 1 to 7, and each test averaged over 100 independent runs with $m * 100$ FES, and the used round numbers method is ceiling.

Here, we provide an overall performance comparison between these tests in Tables 5, 6, and 7. The better result among those is indicated by Boldface in the table. Storn and Price in [34] have indicated that a reasonable value for F is usually between 0.4 and 1, and a good initial choice of F was 0.5. The parameter C_r controls how many parameters in expectation are changed in a population member. Feoktistov and Janaqi [43] claimed that a plausible choice of the crossover rate C_r is between [0.3, 0.9]. Recently, the authors in [44] state that C_r should lie in (0, 0.2) when the function is separable, while in (0.9, 1) when the function's parameters are dependent. For the disjoint set covers problem it likely belongs to separable function; so we apply $C_r = 0.3$ and $C_r = 0.5$ as comparison, respectively, and provide the detail results in Tables 5–7.

From Tables 5–7, it can be seen that the "DE/best/1" strategy with $F = 0.5$ and $C_r = 0.3$ provides the best performance on cases 1–7. To be specific, considering the values of F and C_r in each mutation strategy, it is observed that using $F = 0.5$ and $C_r = 0.3$ the results are better than that using $F = 0.9$ and $C_r = 0.5$. In especially the cases 1, 5, and 6, the performance of using the former parameter combinations significantly improves.

On the other hand, consider the mutation strategy with different values of F and C_r. Tables 5–7 show that only using "DE/best/1" strategy can reach 100 percent success rates to find $C_{\max}$ in case 5. And in case 6 the performance of using "DE/best/1" strategy is 87, which is the best in the three mutation strategies.

As a conclusion, according to the experimental results in Tables 5–7 and the above analysis, the "DE/best/1" mutation strategy using $F = 0.5$ and $C_r = 0.3$ has the best performance in DEDSC.

4.5. Test for DE Variants with Self-Adaptive Parameters. In this subsection, we study the effect of DE's variants which with self-adaptive parameter control strategy.

Inspired by "jDE" [39] we humbly employ the self-adaptive scheme to update parameters for solving the DSC problem, which can self-update to F and C_r among perform. Moreover, various mutation strategies are tested, and the comparison results are provided in Table 8.

In this subsection test, we not only apply the idea of self-adaptive in "jDE" to update F and C_r, but also test some

Table 5: Test "DE/rand/1" strategy on cases 1–7.

Case index	DE/rand/1							
	$F = 0.5, C_r = 0.3$				$F = 0.9, C_r = 0.5$			
	ok %	Best	Mean	avg E	ok %	Best	Mean	avg E
1	**100**	30	30	**146.5**	91	30	29.96	286.6
2	100	23	23	**10.6**	100	23	23	10.2
3	100	21	21	**10**	100	21	21	10
4	100	35	35	**28.1**	100	35	35	72.8
5	**99**	41	40.99	**407.3**	57	41	40.72	605.8
6	**81**	44	43.96	**561.8**	34	44	43.5	564.1
7	100	42	42	**39.7**	100	42	42	94.3

Table 6: Test "DE/current-to-best/1" strategy on cases 1–7.

Case index	DE/current-to-best/1							
	$F = 0.5, C_r = 0.3$				$F = 0.9, C_r = 0.5$			
	ok %	Best	Mean	avg E	ok %	Best	Mean	avg E
1	**99**	30	29.99	**203.5**	98	30	29.98	275.32
2	100	23	23	**10.1**	100	23	23	10.3
3	100	21	21	10	100	21	21	10
4	100	35	35	**51.42**	100	35	35	61.63
5	**83**	41	40.91	**524.7**	57	41	40.69	579.42
6	**61**	44	43.69	**687.7**	35	44	43.47	723.49
7	100	42	42	**76.81**	100	42	42	98.72

Table 7: Test "DE/best/1" strategy on cases 1–7.

Case index	DE/best/1							
	$F = 0.5, C_r = 0.3$				$F = 0.9, C_r = 0.5$			
	ok %	Best	Mean	avg E	ok %	Best	Mean	avg E
1	**100**	30	30	**125.8**	94	30	29.96	279.2
2	100	23	23	10.1	100	23	23	10.1
3	100	21	21	10	100	21	21	10
4	100	35	35	**37.14**	100	35	35	68.72
5	**100**	41	41	**400.8**	60	41	40.71	566.2
6	**87**	44	43.92	**584.1**	29	44	43.51	666.2
7	100	42	42	**42.41**	100	42	42	102.6

Table 8: Effects of paramaters on performance of jDE, averaged over 100 independent runs with $m * 100$ FES.

	DEDSC with self-adaptive update F and C_r (jDE)													
	jDE/rand/1				jDE/current-to-best/1				jDE/best/1				DEDSC	
	ok %	Best	Mean	avg E	ok %	Best	Mean	avg E	ok %	Best	Mean	avg E	ok %	avg E
1	94	30	29.95	173.14	84	30	29.87	220.92	**100**	30	30	131.53	**100**	**125.8**
2	100	23	23	10.1	100	23	23	10.7	100	23	23	10.2	100	10.1
3	100	21	21	10	100	21	21	10	100	21	21	10	100	10
4	100	35	35	**35.25**	100	35	35	48.14	100	35	35	48.09	100	37.14
5	97	41	40.97	**145.23**	77	41	40.81	224.53	90	41	40.92	356.47	**100**	400.8
6	90	44	43.92	**240.86**	83	44	43.84	301.42	83	44	43.87	281.43	**87**	584.1
7	100	42	42	43.91	100	42	42	53.88	100	42	42	60.81	100	**42.41**

variants of "jDE," such that the various mutation strategies in "jDE" are tested. Note that in original "jDE," the initialized $F = 0.5$, $C_r = 0.5$, and the mutation strategy is "DE/rand/1." Therefore in this subsection test, $F = 0.5$ and $C_r = 0.5$ are used in all "jDE" and their variants, but different mutation strategies are applied in different variants.

Table 8 shows that in the case 5, DEDSC outperforms "jDE" and its variants significantly, only DEDSC can attain 100 percent success rate to find $C_{\max}$. In case 1, both DEDSC and "jDE/best/1" can achieve 100% success rate; the "jDE/current-to-best/1" and original "jDE" only reach 84% and 94%, respectively. This means that the original "jDE" maybe is not very suitable for this problem, and the mutation strategy "/best/1" is better than others. Consider case 6, all of those algorithms cannot reach 100% success rate, original "jDE" shows the best performance, and our proposed DEDSC ranks second. It implies that our algorithm maybe lost in local optimum and needs more suitable control parameters to improve its performance.

As a conclusion, DEDSC is the best among the three algorithms in comparison to cases 1, 5, and 6. However, because the performance of DEDSC is not better than STHGA in case 6, we intend to study more suitable parameters in the future work.

5. Conclusion

Considering maximizing the lifetime of wireless sensor networks, our paper here proposes using differential evolution to solve this optimization problem, in which algorithm is termed DEDSC.

In DEDSC, there are just two parameters, mutate factor rate F and crossover rate C_r. The major features of DEDSC are recombination operation and fitness computation, and the former guarantees at least one *critical target*'s sensor is divided into different disjoint sets. This is a very important operation; especially there are many *critical targets* in test data set. Our method for computing fitness not only considers the number of complete cover subset, but also contains the cover percent of those incomplete cover subsets. To verify the DEDSC's effectiveness, an extensive performance comparison has been conducted over 7 commonly used test cases.

The experimental results suggested that its overall performance was better than GAMDSC and almost same as STHGA. Considering on the convergence rate, DEDSC needs the least evolution generations to find $C_{\max}$ among the above algorithms. Moreover, a scalability study test was carried out using the different parameters and mutation strategies for DEDSC. In addition, for updating the parameter self-adaptive we tentatively combine "jDE" with DEDSC for comparing, and three "jDE" mutation strategies are tested. From the above results we obtain that the DEDSC with "DE/best/1" mutation strategy and $F = 0.5$, $C_r = 0.3$ have promising performance to maximize the lifetime of WSN.

However, using other self-adaptive parameter schemes, such as "JADE" [29], to improve the performance of DEDSC is a future direction for the algorithm.

Furthermore, there are many evolutionary algorithms such as ant colony optimization (ACO) have been applied to research the problem of WSN. Therefore, our further research work is possible to study ACO [21, 26] for improving the performance of our method. Some possible experimental research is now under our investigation in algorithm design and test.

Authors' Contribution

Y. Xu and J. Fang contributed equally to this work.

Acknowledgments

This paper was supported by the Key Creative Project of Shanghai Education Community (13ZZ050) and the Key Foundation Project of Shanghai (12JC1400400). The authors are grateful to the Editor-in-Chief, Associate Editor, and anonymous reviewers for their constructive suggestions that helped to improve the content as well as the quality of the paper.

References

[1] T. D. Raty, "Survey on contemporary remote surveillance systems for public safety," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 40, no. 5, pp. 493–515, 2010.

[2] I. F. Akyildiz, T. Melodia, and K. R. Chowdury, "Wireless multimedia sensor networks: a survey," *IEEE Wireless Communications*, vol. 14, no. 6, pp. 32–39, 2007.

[3] M. Younis and K. Akkaya, "Strategies and techniques for node place-ment in wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 6, no. 4, pp. 621–655, 2008.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[5] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[6] I. Dietrich and F. Dressler, "On the lifetime of wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 5, no. 1, article 5, 2009.

[7] F. Marcelloni and M. Vecchio, "A simple algorithm for data compression in wireless sensor networks," *IEEE Communications Letters*, vol. 12, no. 6, pp. 411–413, 2008.

[8] S. Yang, H. Cheng, and F. Wang, "Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks," *IEEE Transactions on Systems, Man and Cybernetics Part C*, vol. 40, no. 1, pp. 52–63, 2010.

[9] S. Okdem and D. Karaboga, "Routing in wireless sensor networks using an Ant Colony optimization (ACO) router chip," *Sensors*, vol. 9, no. 2, pp. 909–921, 2009.

[10] F. Ge, Y. Wang, Q. Wang, and J. Wang, "Energy efficient broadcasting based on ant colony optimization in wireless sensor networks," in *Proceedings of the 3rd International Conference on Natural Computation (ICNC '07)*, pp. 129–133, Haiko, China, 2007.

[11] A. Acharya, A. Seetharam, A. Bhattacharyya, and M. K. Naskar, "Balanc-ing energy dissipation in data gathering wireless sensor networks using ant colony optimization," in *Proceedings of*

the International Conference on Distributed Computing and Networking (ICDCN '09), pp. 437–443, Hyderabad, India, 2009.

[12] C.-Y. Chang, J.-P. Sheu, Y.-C. Chen, and S.-W. Chang, "An obstacle-free and power-efficient deployment algorithm for wireless sensor networks," *IEEE Transactions on Systems, Man, and Cybernetics Part A*, vol. 39, no. 4, pp. 795–806, 2009.

[13] H. Chen, C. K. Tse, and J. Feng, "Impact of topology on performance and energy efficiency in wireless sensor networks for source extraction," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 6, pp. 886–897, 2009.

[14] Y. Liang, J. Cao, L. Zhang, R. Wang, and Q. Pan, "A biologically inspired sensor wakeup control method for wireless sensor networks," *IEEE Transactions on Systems, Man and Cybernetics Part C*, vol. 40, no. 5, pp. 525–538, 2010.

[15] R. Tharmarasa, T. Kirubarajan, and M. L. Hernandez, "Large-scale op-timal sensor array management for multi-target tracking," *IEEE Transactions Systems, Man, and Cybernetics, Part C*, vol. 37, no. 5, pp. 803–814, 2007.

[16] R. Tharmarasa, T. Kirubarajan, J. Peng, and T. Lang, "Optimization-based dynamic sensor management for distributed multitarget tracking," *IEEE Transactions on Systems, Man and Cybernetics Part C*, vol. 39, no. 5, pp. 534–546, 2009.

[17] D. Tian and N. D. Georganas, "Connectivity maintenance and coverage preservation in wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 6, pp. 744–761, 2005.

[18] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc and Sensor Wireless Networks*, vol. 1, no. 1-2, pp. 89–124, 2005.

[19] T. R. Park, K. J. Park, and M. J. Lee, "Design and analysis of asynchronous wakeup for wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 11, pp. 5530–5541, 2009.

[20] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor net works: a survey," *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009.

[21] Y. Lin, J. Zhang, S. H. Chung, and Y. Li, "An ant colony optimization approach for maximizing the lifetime of heterogeneous wireless sensor networks," *IEEE Transactions on Systems, Man, and Cybernetics Part C*, vol. 42, no. 3, pp. 408–420, 2012.

[22] M. Cardei and D. Z. Zhang, "Improving wireless sensor network lifetimethrough power aware organization," *Wireless Networks*, vol. 11, no. 3, pp. 333–340, 2005.

[23] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC '01)*, vol. 2, pp. 472–476, Helsinki, Finland, 2001.

[24] C. C. Lai, C. K. Ting, and R. S. Ko, "An effective genetic algorithm to improve wireless sensor network lifetime for large-scale surveillance applications," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 3531–3538, 2007.

[25] X. M. Hu, J. Zhang, Y. Yu et al., "Hybrid genetic algorithm using a forward encoding scheme for lifetime maximization of wireless sensornetworks," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 766–781, 2010.

[26] Y. P. Zhong, P. W. Huang, and B. Wang, "Maximum lifetime routing based on ant colony algorithm for wireless sensor networks," in *Proceedings of the IET Conference on Wireless, Mobile and Sensor Networks*, pp. 789–792, Shanghai, China, 2007.

[27] Q. Zhao and M. Gurusamy, "Lifetime maximization for connected target coverage in wireless sensor net works," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1378–1391, 2008.

[28] Y. Tang, Z. Wang, and J. A. Fang, "Controller design for synchronization of an array of delayed neural networks using a controllable probabilistic PSO," *Information Sciences*, vol. 181, no. 20, pp. 4715–4732, 2011.

[29] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperativelearning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[30] Y. Tang, Z. Wang, and J. Fang, "Feekback learning particle swarm optimization," *Applied Soft Computing*, vol. 11, pp. 4713–4725, 2011.

[31] Y. Tang and H. J. Gao, "Distributed synchronization in networks of agent systems with nonlinearities and random switchings," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 43, no. 1, pp. 358–370, 2013.

[32] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[33] W. Zhu, J. A. Fang, Y. Tang, W. B. Zhang, and Wei Du, "Digital IIR filters design using differential evolution algorithm with a controllable probabilistic population size," *PLoS ONE*, vol. 7, no. 7, Article ID e40549, 2012.

[34] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[35] Y. Tang, Z. Wang, H. Gao, S. Swift, and J. Kurths, "A constrained evolutionary computation method for detecting controlling regions of cortical networks," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 9, pp. 1569–1581, 2012.

[36] W. Zhu, Y. Tang, J. A. Fang, and W. B. Zhang, "Adaptive population tuning scheme for differential evolution," *Information Sciences*, vol. 223, pp. 164–191, 2013.

[37] W. Zhu, J. A. Fang, Y. Tang, W. B. Zhang, and Y. L. Xu, "Identification of fractional-order systems via a switching differential evolution subject to noise perturbations," *Physics Letters A*, vol. 376, pp. 3113–3120, 2012.

[38] Y. Tang, H. J. Gao, J. Kurths, and J. A. Fang, "Evolutionary pinning control and its application in uav coordination," *IEEE Transactions on Industrial Informatics*, vol. 8, pp. 828–838, 2012.

[39] J. Brest, S. Greiner, B. Boskovic et al., "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[40] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[41] Q. Cao, T. Yan, J. Stankovic, and T. Abdelzaher, "Analysis of target detection performance for wireless sensor networks," in *Proceedings of the 1st IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '05)*, pp. 276–292, July 2005.

[42] A. Boukerche, X. Fei, and R. B. Araujo, "An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange," *Computer Communications*, vol. 30, no. 14-15, pp. 2708–2720, 2007.

[43] V. Feoktistov and S. Janaqi, "Generalization of the strategies in differential evolution," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS '04)*, p. 165, April 2004.

[44] J. Ronkkonen, S. Kukkonen, and K. V. Price, "Real parameter optimiza-tion with differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1, pp. 506–513, 2005.