Maximizing lifetime of range-adjustable wireless sensor networks: a neighborhood-based estimation of distribution algorithm

Maximizing Lifetime of Range-Adjustable Wireless Sensor Networks: A Neighborhood-Based Estimation of Distribution Algorithm

Zong-Gan Chen, *Student Member, IEEE*, Ying Lin, *Member, IEEE*, Yue-Jiao Gong, *Senior Member, IEEE*, Zhi-Hui Zhan, *Senior Member, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Sensor activity scheduling is critical for prolonging the lifetime of wireless sensor networks (WSNs). However, most existing methods assume sensors to have one fixed sensing range. Prevalence of sensors with adjustable sensing ranges posts two new challenges to the topic: 1) expanded search space, due to the rise in the number of possible activation modes and 2) more complex energy allocation, as the sensors differ in the energy consumption rate when using different sensing ranges. These two challenges make it hard to directly solve the lifetime maximization problem of WSNs with range-adjustable sensors (LM-RASs). This article proposes a neighborhood-based estimation of distribution algorithm (NEDA) to address it in a recursive manner. In NEDA, each individual represents a coverage scheme in which the sensors are selectively activated to monitor all the targets. A linear programming (LP) model is built to assign activation time to the schemes in the population so that their sum, the network lifetime, can be maximized conditioned on the current population. Using the activation time derived from LP as individual fitness, the NEDA is driven to seek coverage schemes promising for prolonging the network lifetime. The network lifetime is thus optimized by repeating the steps of the coverage scheme evolution and LP model solving. To encourage the search for diverse coverage schemes, a neighborhood sampling strategy is introduced. Besides, a heuristic repair strategy is designed to fine-tune the existing schemes for further improving the search efficiency. Experimental results on WSNs of different scales show that NEDA outperforms state-of-the-art approaches. It is also expected that NEDA can serve as a potential framework for solving other flexible LP problems that share the same structure with LM-RAS.**

***Index Terms*—Estimation of distribution algorithm, lifetime maximization, neighborhood, wireless sensor network (WSN).**

Manuscript received June 11, 2019; revised October 8, 2019 and December 15, 2019; accepted February 25, 2020. Date of publication April 1, 2020; date of current version November 9, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61772569, Grant 61873095, and Grant U1611262. This article was recommended by Associate Editor R. Selmic. *(Corresponding author: Ying Lin.)*

Zong-Gan Chen, Yue-Jiao Gong, and Zhi-Hui Zhan are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, South China University of Technology, Guangzhou 510006, China.

Ying Lin is with the Department of Psychology, Sun Yat-sen University, Guangzhou 510006, China (e-mail: linying23@mail.sysu.edu.cn).

Jun Zhang is with Victoria University, Melbourne, VIC 8001, Australia (e-mail: junzhang@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2020.2977858.

Digital Object Identifier 10.1109/TCYB.2020.2977858

I. Introduction

WIRELESS sensor networks (WSNs) have been deployed for automation of various tasks, such as environmental monitoring, intrusion detection, and intelligent farming [1]–[3]. The lifetime of a WSN is the total amount of time that the WSN can stay functional and fulfill certain objectives [4]–[6]. Many functions of WSNs rely on full coverage of specified objects, which can be an area [7]–[9]; a barrier [10], [11]; or a set of targets [12]–[14]. In many scenarios, batteries of sensors have limited energy and recharging the batteries is impractical. Therefore, how to utilize sensor energy efficiently for guaranteeing full coverage becomes the key to prolong the lifetime of WSNs.

The sensor activity scheduling is an effective way to prolong the lifetime of WSNs. In order to provide high-quality service, sensors are usually densely deployed. Activating a subset of sensors is already sufficient for achieving full coverage of specified objects (called a coverage scheme), and the other sensors can switch to the sleep mode for saving energy. Therefore, the network lifetime can be prolonged by finding a set of disjoint coverage schemes that use different sensors and activating them in sequence. Following this idea, a wealth of sensor activity scheduling approaches has been proposed. Liao and Ting [15] proposed a memetic algorithm based on a genetic algorithm and a local search operator. Wan *et al.* [16] proposed a scheduling algorithm for the roadside sensor network, where the road should be monitored by multiple sensors simultaneously. Zhang *et al.* [17] proposed a Kuhn–Munkres parallel genetic algorithm that used a divide-and-conquer strategy for the large-scale WSNs. Chen *et al.* [18] proposed a hybrid memetic framework that considered the dynamic coverage maintenance.

All the above approaches are subject to the assumption that sensors have a fixed sensing range. However, in recent years, the usage of sensors with adjustable sensing ranges has become more prevalent [19], [20]. This posts new challenges to the sensor activity scheduling task. With adjustable sensing ranges, the sensor activation mode is no longer binary (sleep/activated) but multilevel. The search space for finding coverage schemes thus becomes larger. In addition, sensors activated at different sensing ranges consume different amounts of energy for the same length of time. Thus, sensors activated in one coverage scheme may have

different remaining energy left after activation. The sensors with abundant remaining energy can continue to monitor targets, which should be considered in the formation of other coverage schemes for the sake of efficient energy utilization. However, the fact that one sensor can be used in multiple coverage schemes makes energy allocation more complex. To deal with these new challenges, a few new scheduling approaches have been proposed. Chang *et al.* [21] proposed a sensing radius adaptation approach based on a weighted Voronoi diagram for area coverage. Zhang *et al.* [22] proposed a multiobjective algorithm for barrier coverage to minimize the total power consumption, the number of active sensors, and the maximum sensing range of sensors. Since area and barrier coverage can both be converted into the special cases of target coverage, we focus on solving the lifetime maximization problem of WSNs with range-adjustable sensors (LM-RASs) in a target-coverage scenario. In this regard, Rossi *et al.* [23] proposed a column generation approach assisted by a genetic algorithm. However, it suffers from poor efficiency due to the time-consuming steps in generating new columns, particularly when facing large-scale WSNs. Cerulli *et al.* [24] proposed a heuristic approach based on a greedy algorithm and a local search procedure, but it is easy to get trapped into local optima. Despite these pioneering efforts, an effective and efficient algorithm for LM-RAS is still lacking.

This article proposes a novel neighborhood-based estimation of distribution algorithm (NEDA) to address the LM-RAS problem. Instead of solving LM-RAS directly, the NEDA divides LM-RAS into two subproblems: 1) finding a set of coverage schemes and 2) determining the activation time of each coverage scheme so that sensors' energy is best allocated to maximize the network lifetime. Clearly, solutions to the first subproblem compose the premise for the second one. In fact, supposing that all the possible coverage schemes in a WSN have been found, the second subproblem can be modeled as a linear programming (LP) problem, where the activation time of each scheme can be optimally assigned so that their sum, the network lifetime, is maximized. However, due to the large search space of LM-RAS, it is difficult or even impossible to enumerate all the coverage schemes. The essential idea behind NEDA is to address LM-RAS by solving the two subproblems recursively. In NEDA, each individual in the population represents a coverage scheme. An LP model is built to assign activation time to each coverage scheme in such a way that the network lifetime is maximized conditioned on the current population. The activation time is used as the individual's fitness, which drives NEDA to generate coverage schemes that are promising for prolonging the network lifetime. The new schemes, which form a new population, are again fed into the LP model for activation time allocation that may further prolong the network lifetime. As such, by repeating the steps of the coverage scheme evolution and LP model solving, NEDA can gradually approach the globally optimal network lifetime.

Maintaining high-population diversity is the key for NEDA to achieve good performance. This is because using similar coverage schemes that have many common sensors can induce imbalanced energy consumption across the WSN, resulting in a short lifetime. Therefore, to provide a set of diverse coverage schemes for efficiently utilizing each sensor, a neighborhood sampling strategy (NSS) is developed, which generates new coverage schemes based on the probabilistic models derived from neighborhoods. Moreover, a heuristic repair strategy (HRS) is also introduced, which uses the remaining energy of sensors as heuristic information to fine-tune the coverage schemes and thus enhances the search efficiency. Experiments are performed on WSN instances of different scales. The experimental results are compared with state-of-the-art approaches for validating the effectiveness and efficiency of the proposed NEDA.

In real world, many problems share the same structure with the LM-RAS problem: they all contain two interdependent subproblems and the first one is the premise for optimally solving the second one in an LP manner. We name them as *flexible LP* (fLP) problems (refer to Section II-B for details). It is also expected that NEDA can serve as a potential framework for solving the other fLP problems.

The remainder of this article is organized as follows. Section II presents the formulation of the LM-RAS problem and its generalization as fLP problems. Section III introduces the proposed NEDA approach and its novelties in detail. Section IV presents the experimental study. Finally, Section V draws a conclusion to this article.

II. Problem Formulation and Generalization

A. Lifetime Maximization Problem of WSNs With Range-Adjustable Sensors

The optimization objective of the LM-RAS problem is to maximize the lifetime of WSN while satisfying two constraints: one is to guarantee that all targets are monitored simultaneously and consecutively; and the other is that the sensor activation time is limited by the battery energy.

1) Sensor Features: We define the set of sensors in WSN as $\boldsymbol{S} = \{s_1, s_2, s_3, \ldots, s_{|S|}\}$. All the sensors are of the same type. Each sensor's battery has an initial energy b_{ini} and cannot be recharged. As mentioned above, the sensing range of a sensor is adjustable. The set of selectable sensing ranges is a finite set of discrete values, defined as $\boldsymbol{R} = \{r_0, r_1, r_2, r_3, \ldots, r_K\}$, in which the elements, without loss of generality, satisfy $r_i < r_{i+1}$ $(i = 0, 1, \ldots, K-1)$. r_0 indicates inactivation (i.e., $r_0 = 0$). A tuple (s_j, r_i) represents the sensor s_j is activated at the sensing range r_i. The energy consumption rate corresponding to each sensing range, defined as $\boldsymbol{E} = \{e_0, e_1, e_2, e_3, \ldots, e_K\}$, is computed according to the quadratic model [23], as

$$e_i = e_K \times (r_i/r_K)^2 \tag{1}$$

where e_K, the energy consumption rate of the largest sensing range, is given in advance.

2) Coverage Scheme: A coverage scheme is a subset of sensors with selected sensing ranges that can achieve full coverage over all the targets. For example, in Fig. 1, there are three sensors s_1, s_2, and s_3 and three targets t_1, t_2, and t_3. $\{(s_1, r_2), (s_2, r_1), (s_3, r_2)\}$ is a coverage scheme. However, this coverage scheme is not considered energy efficient due to coverage redundancy. An improved version would be to reduce

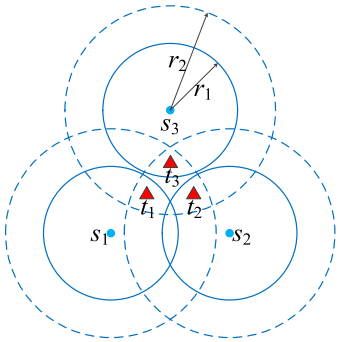

Fig. 1. WSN instance.

the sensing range of s_1 from r_2 to r_1 for energy saving. In the sense of energy efficiency, we define the nondominated coverage scheme (ndCS) as a coverage scheme whose sensing range reduction or inactivation of any involved sensors will cause the coverage scheme to be infeasible, that is, losing full target coverage. In the example of Fig. 1, $\{(s_1, r_1), (s_2, r_1), (s_3, r_1)\}$ is an ndCS. It is obvious that using ndCSs is always more energy efficient than using simply coverage schemes.

3) LP Model for Maximizing Network Lifetime: Given a WSN with fixed sensors and targets, we could maximize its lifetime if we can identify a specific set of ndCSs and activate them one by one in a way that utilizes energy of each sensor to the best. Considering the instance in Fig. 1, assume that each sensor can be activated for 2 h with the sensing range set at r_1, but for only 1 h at r_2. If we activate the ndCS $\{(s_1, r_1), (s_2, r_1), (s_3, r_1)\}$ till all the sensors run out of energy, the network lifetime is 2 h. However, there are also three other ndCSs: 1) $\{(s_1, r_2)\}$; 2) $\{(s_2, r_2)\}$; and 3) $\{(s_3, r_2)\}$. If we activate these three ndCSs in turn, the network lifetime is prolonged to 3 h.

Following the above idea, the LM-RAS problem can be divided into two interdependent subproblems: 1) identify a proper set of ndCSs and 2) determine the activation time of each identified ndCS. Let $\boldsymbol{CSS} = \{ndCS_1, ndCS_2, \ldots, ndCS_M\}$ denote the set of ndCSs, and ε_i be the activation time of the ith ndCS in $\boldsymbol{CSS}$, that is, $ndCS_i$ $(i = 1, 2, \ldots, M)$. The second subproblem can be formulated by an LP model for maximizing the network lifetime, as

$$\max \ NL = \sum_{i=1}^{M} \varepsilon_i \tag{2}$$

$$\text{s.t.} \begin{cases} \sum_{i=1}^{M} e_{ndCS_{i,j}} \times \varepsilon_i \le b_{\text{ini}} \quad \forall j \in \{1, 2, 3, \ldots, |\boldsymbol{S}|\} & (3) \\ \varepsilon_i \ge 0 \quad \forall i \in \{1, 2, 3, \ldots, M\}. & (4) \end{cases}$$

In this formulation, (2) describes the optimization objective, where the network lifetime NL is calculated as the sum of activation time of ndCSs in $\boldsymbol{CSS}$ because the ndCSs will be activated sequentially. Equation (3) reflects the constraint of battery energy for each sensor, where $ndCS_{i,j}$ represents the index of the selected sensing range for the sensor s_j in $ndCS_i$. Note that a sensor may be activated in multiple ndCSs, and the energy it consumes during the activation of each ndCS must be taken into account. Equation (4) is a boundary constraint by definition, which requires the activation time of each ndCS to be non-negative.

Given that the first subproblem is solved, that is, $\boldsymbol{CSS}$ is determined, the second subproblem (i.e., the LP model) can be addressed by a variety of matured techniques in polynomial time [25], [26]. Therefore, the key to solve the entire problem lies in the first subproblem. A naïve solution is to enumerate all ndCSs in the considered WSN. However, considering the numerous combinations of different sensors and different sensing ranges, ndCS enumeration is too time consuming to be practical. Moreover, a large $\boldsymbol{CSS}$ may lead to poor efficiency in solving the LP model. Therefore, we propose the NEDA to find a proper ndCS set, which will be introduced in Section III.

B. Flexible Linear Programming Problems: Generalization of LM-RAS

The LM-RAS problem can be generalized as a kind of problems that share the same structure. That is, they can be formulated as

$$\max f(\boldsymbol{\beta}|\boldsymbol{\alpha}) = \sum_{\alpha_i \in \boldsymbol{\alpha}} \beta_i \ \text{ s.t. } g(\boldsymbol{\beta}|\boldsymbol{\alpha}) \le 0 \tag{5}$$

where the decision variable vector $\boldsymbol{\beta}$ and the pattern set $\boldsymbol{\alpha}$ have the same number of elements (denoted as NE). Each decision variable β_i in $\boldsymbol{\beta}$ is related to the pattern α_i in $\boldsymbol{\alpha}$ $(i = 1, 2, \ldots, NE)$. $g(\boldsymbol{\beta}|\boldsymbol{\alpha})$ is a group of linear functions that represents problem constraints. For example, in the LM-RAS problem, $\boldsymbol{\alpha}$ is the set of ndCSs and $\boldsymbol{\beta}$ is the vector composed of their activation time. A salient feature of such problems is that once $\boldsymbol{\alpha}$ is determined, the problems are reduced to classical LP problems, and the exact optimal solution conditioned on $\boldsymbol{\alpha}$ can be deterministically obtained by LP techniques. We identify this kind of problems as fLP problems, where the term "flexible" reflects uncertainty in the size and content of the pattern sets, that is, $\boldsymbol{\alpha}$ in (5).

As mentioned above, the key to solve an fLP problem lies in the formation of a proper pattern set. To approximate the global optimal solution efficiently, it is expected that the set includes all the patterns used in the global optimal solution, but excludes unrelated patterns (i.e., those that are not used in the global optimal solution) as many as possible. Missing one or more patterns used in the global optimal solution will deviate the search procedure from the global optimum while including too many unrelated patterns will slow down the search procedure.

This article proposes the NEDA to address the pattern set formation in the LM-RAS problem. It can also be extended to other fLP problems for pattern set formation. Thus, it arises the main contribution of this article: the identification of fLP problems and a potential NEDA-based framework for solving them.

III. NEDA

EDA is a kind of evolutionary computation techniques [27]–[30]. It evolves the population by sampling the probabilistic model(s) constructed by promising individuals.

Algorithm 1 Basic Framework of EDA

1: Generate an initial population;
2: Evaluate the fitness of each individual in the population;
3: Select a set of promising individuals from the population;
4: Construct the probabilistic model based on the selected individuals;
5: Sample new individuals according to the probabilistic model;
6: Select individuals from those in the current population and those sampled by the probabilistic model to form a new population for the next generation;
7: Terminate if the termination criterion is met, otherwise jump back to Step 2;

Encoded Individual

dimension	1	2	3	4	5	6	7	8	9	10
value	2	0	3	1	0	3	2	1	3	1

Fig. 2. Example of encoded individual.

The basic framework of EDA is shown in Algorithm 1. As can be seen, after generating and evaluating the initial population (steps 1 and 2), a set of promising individuals (usually those with good fitness) is selected to construct a probabilistic model (steps 3 and 4), according to which new individuals are sampled (step 5). Then, the new population is formed by selecting from the union of the individuals in the current population and the new individuals sampled by the probabilistic model (step 6). By repeating the steps of model construction and sampling until the termination criterion is met (step 7), EDA can drive the population toward the optimum of the problem. EDAs have been successfully applied to a number of real-world optimization problems [31]–[33]. In this article, following the basic framework of EDA, the proposed NEDA utilizes a problem-dependent encoding scheme and incorporates the following three novel strategies to solve the LM-RAS problem: 1) an LP-based fitness evaluation (LPFE) strategy; 2) an NSS; and 3) an HRS.

A. Encoding Scheme

Each individual in the population of NEDA, denoted by X_i ($i = 1, 2, \ldots, N$ with N as the population size), is a sequence that represents an ndCS. Each element $X_{i,j}$ in X_i denotes the index of the sensing range at which the sensor s_j is activated ($j = 1, 2, \ldots, |\boldsymbol{S}|$). The number of dimensions is thus the number of sensors, that is, $|\boldsymbol{S}|$. As mentioned in Section II-A, r_0 represents inactivation. Therefore, s_j is not activated if $X_{i,j}$ equals zero. An example of the above encoding scheme is shown in Fig. 2. As can be seen, s_2 and s_5 are not activated; s_4, s_8, and s_{10} are activated at r_1; s_1 and s_7 are activated at r_2; and s_3, s_6, and s_9 are activated at r_3.

In this article, we assume that the selectable sensing ranges of each sensor are a finite set of discrete values, defined as $R = \{r_0, r_1, r_2, r_3, \ldots, r_K\}$. However, not all the sensing ranges are energy-efficient choices for each sensor. In some cases, activation at a larger sensing range, which consumes the sensor more energy, cannot bring more gain in target coverage. Therefore, a preprocessing procedure is performed before population initialization to find out the energy-efficient sensing ranges for each sensor according to the target distribution. If a sensing range allows the sensor to monitor more targets than its next smaller sensing range, it is considered an energy-efficient sensing range, otherwise, it is an energy-inefficient sensing range. Take sensor s_1 in Fig. 3 as an example. r_3 is an energy-efficient sensing range for s_1 because it allows s_1 to monitor two more targets (t_2 and t_3) compared to r_2. However, r_2 is an energy-inefficient sensing range for s_1 because it cannot allow s_1 to monitor more targets compared to r_1. Note that after conducting the preprocessing procedure, the energy-efficient sensing ranges of different sensors may be different. The individuals can only choose each sensor's energy-efficient sensing ranges in the corresponding dimension. By doing so, the search space of the problem can be reduced while all ndCSs are still preserved.

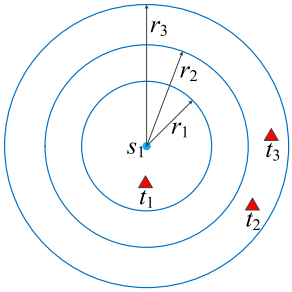

Fig. 3. Example of determining energy-efficient sensing ranges.

B. LP-Based Fitness Evaluation

NEDA defines the fitness of an individual as the activation time of its corresponding ndCS, which is derived from the solution of the LP model formulated in (2)–(4) with the current population fed as the ndCS set in (3). Longer activation time of an ndCS indicates a greater contribution to the network lifetime, and thus is a suitable fitness metric to guide the generation of promising ndCSs. Consider the example in Fig. 1. Assume that the population has four individuals: 1) $X_1 = \{(s_1, r_1), (s_2, r_1), (s_3, r_1)\}$; 2) $X_2 = \{(s_1, r_2)\}$; 3) $X_3 = \{(s_2, r_2)\}$; and 4) $X_4 = \{(s_3, r_2)\}$. Solving the LP model we obtain $\varepsilon_1 = 0$, $\varepsilon_2 = 1$, $\varepsilon_3 = 1$, and $\varepsilon_4 = 1$, which represents that $\{(s_1, r_1), (s_2, r_1), (s_3, r_1)\}$ is not activated while the other three are activated for 1 h, respectively. Therefore, the fitness values of the four individuals are $f(X_1) = 0$ and $f(X_2) = f(X_3) = f(X_4) = 1$.

Note that in NEDA, all the individuals act as the input to the LP model and after solving the LP model, the fitness of each individual is obtained. That is, the LPFE only needs to be carried out once to obtain all the individuals' fitness, which is distinct from the traditional EDAs.

C. Neighborhood Sampling Strategy

A promising ndCS set to prolong the network lifetime should make use of each sensor as fully as possible. Moreover, the overlap of the activated sensors across different ndCSs should be minimized to achieve relatively balanced energy utilization among the sensors. Herein, the word "overlap" indicates that one sensor can participate in multiple ndCSs simultaneously, using either the same or different sensing

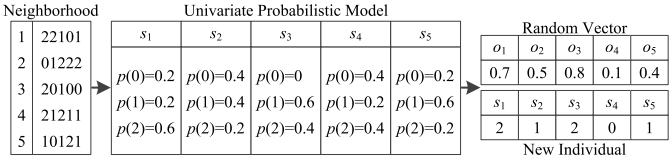

Fig. 4. Example of sampling a new individual by NSS.

Algorithm 2 NSS

1: ***SP*** $= \emptyset$
2: **While** $|\boldsymbol{SP}| < (N - |\boldsymbol{SI}|)$
3: Randomly select an individual $X_i \in \boldsymbol{SI}$;
4: Find the $n-1$ individuals nearest to X_i in ***SI***, incorporate them with X_i to construct X_i's neighborhood;
5: Construct a univariate probabilistic model based on X_i's neighborhood;
6: Sample one new individual from the above probabilistic model, repair it by HRS if it is not an ndCS;
7: If the new individual is not the same as the individuals in ***SI*** and ***SP***, add it to ***SP***, otherwise discard it;
8: **End While**

ranges. To achieve these two goals, the key is to enhance the population diversity of NEDA. The NSS is thus proposed.

Typically, EDAs select a proportion of the population to construct a probabilistic model from which new individuals are sampled. In NEDA, we select the individuals whose fitness is larger than zero, that is, those ndCSs that are used (activated) in the solution of the LP model. The set of selected individuals is denoted as

$$\boldsymbol{SI} = \{X_i \in \boldsymbol{P} \mid f(X_i) > 0\} \tag{6}$$

where ***P*** represents the population. Note that EDAs usually require a parameter to determine how many individuals are selected to construct the probabilistic model, while the proposed NEDA does not require such a parameter and determines adaptively based on the solution of the LP model.

NSS introduces the idea of neighborhood into the process of constructing the probabilistic models to enhance the population diversity. The procedure of NSS in one generation is shown in Algorithm 2. Let ***SP*** denote the set to store the new individuals sampled by NSS. At the beginning of the NSS procedure, ***SP*** is initialized as an empty set (step 1). Then, the new individuals are generated as follows. First, an individual is randomly selected from ***SI*** (step 3). Second, the neighborhood of the selected individual is formed, including itself as well as the other (n-1) individuals nearest to it according to the Manhattan distance in the search space (step 4). In NEDA, n is set to 5. A univariate probabilistic model is constructed based on the distribution of activation modes of each sensor across the individuals in the neighborhood (step 5). Finally, a new individual is sampled from this probabilistic model (step 6).

Fig. 4 exemplifies the above process of generating a new individual. Assume that the WSN has five sensors with two sensing range options. The left grid in Fig. 4 is a neighborhood that contains five individuals. The middle grid contains five subgrids, each of which shows the selection probabilities of the activation modes for one sensor. For example, three individuals in the neighborhood select r_2 for sensor s_1 while the other two select r_0 and r_1, respectively. Therefore, as shown in the first subgrid, the probabilities for s_1 to be inactivated, be activated at r_1, and be activated at r_2 are $p(0) = 0.2$, $p(1) = 0.2$, and $p(2) = 0.6$, respectively. With the probabilities of the three activation modes calculated in the same way for the other four sensors (as tabulated in the other four subgrids), a probabilistic model is built. A new individual can then be generated by choosing an activation mode for each sensor based on the above probabilistic model. To be more exact, for each sensor s_j, a random number o_j is first derived from the uniform distribution in (0, 1). Then, o_j is compared with the cumulative probability of each activation mode, and the first mode whose cumulative probability goes beyond o_j is chosen for the sensor. Still take Fig. 4 as an example. For sensor s_1, the cumulative probabilities of r_0, r_1, and r_2 are 0.2, 0.4, and 1, respectively. Given that $o_1 = 0.7$, sensor s_1 will be activated at r_2 since $o_1 \in (0.4, 1]$. With the activation mode for the other sensors selected in the same way, a new individual 21 201 is generated (as shown in the right grid). Such a method is better than pure random sampling as it reduces the blindness of the search process. Meanwhile, it is also better than maximum-likelihood sampling (i.e., greedily choosing the activation mode with the highest probability for each sensor) as it enables the generation of more diverse individuals and thus alleviates the risk of premature convergence caused by the loss of population diversity.

The new individual may not always be an ndCS, if it is not, repair it by HRS (refer to Section III-D for HRS). In addition, to prevent the duplicated individuals that may lead to premature convergence, the new individual is compared with the individuals in ***SI*** and the other new individuals (i.e., the individuals in ***SP***). If no duplication is detected, add the new individual to ***SP*** (step 7). The above procedure (steps 3–7) iterates until $(N - |\boldsymbol{SI}|)$ new individuals are generated (step 2). NEDA then forms the population of the next generation by bringing these new individuals together with the original individuals in ***SI***.

D. Heuristic Repair Strategy

In the proposed NEDA, each individual represents an ndCS, that is, a nondominated coverage scheme. However, the individuals sampled from the univariate probabilistic models are not always ndCSs. Therefore, we propose the HRS to transform an individual into an ndCS using the remaining energy of sensors as the heuristic information. The remaining energy of each sensor s_j, denoted by RE_j, is computed by applying the solution of the LP model in (2)–(4), that is

$$RE_j = b_{\text{ini}} - \sum_{i=1}^{N} e_{X_{i,j}} \times \varepsilon_i \quad j \in \{1, 2, 3, \ldots, |S|\}. \tag{7}$$

As shown in Algorithm 3, HRS conducts two operations: 1) refinement and 2) reduction. The refinement operation tunes the coverage scheme to make it feasible. That is, it activates some sensors or increases the sensing ranges of some activated sensors to achieve the coverage over all the targets. The refinement operation deals with each target in a random order (step 3). Then, it adopts a tournament selection strategy to choose a sensor for the unmonitored target. More

Algorithm 3 HRS

1: X_i is the individual to be transformed into an ndCS by HRS;
2: //refinement operation
3: **For Each** target t_k in a random order
4: **If** t_k is not monitored
5: Randomly select two sensors from $\boldsymbol{TS}_k$, denoted by s_a, s_b;
6: $sel = a$ if $RE_a > RE_b$, otherwise $sel = b$;
7: $X_{i,sel} = min\_r$ where r_{min_r} is the minimum sensing range for s_{sel} to monitor t_k;
8: Update the coverage information;
9: **End If**
10: **End For**
11: //reduction operation
12: **For Each** sensor s_j in the ascending order of $\boldsymbol{RE}$;
13: Iteratively do $X_{i,j} = X_{i,j}-1$ if $X_{i,j}>0$ and the WSN can still maintain
full target coverage after that;
14: **End For**

specifically, the set of sensors that can monitor a target t_k is defined as $\boldsymbol{TS}_k$. If t_k is not monitored (step 4), two sensors are randomly picked from $\boldsymbol{TS}_k$ and the one with higher remaining energy are selected to monitor t_k (steps 5 and 6). The sensing range of the selected sensor is set at the minimum that can monitor t_k (step 7). After that, the coverage information is updated immediately (step 8) because the selected sensor may also monitor some other unmonitored targets. The tournament selection strategy in the above refinement operation is intended for striking a balance between randomness and greediness. If the sensors are selected completely at random, it may improve the diversity in sensor usage but also increase the risk of selecting undesired sensors with low remaining energy. In contrast, selecting the sensors solely based on their remaining energy can avoid the energy risk, but results in a less diverse ndCS set.

After performing the refinement operation, the reduction operation is carried out to make the coverage scheme nondominated. For each activated sensor in the scheme, HRS checks whether it can reduce its sensing range or be inactivated while the WSN can still maintain full target coverage. If so, the sensing range of the sensor will be shrunk to the lowest possible level. Note that the checking order can affect the performance of the reduction operation, because several sensors may monitor the same target. Herein, we check the activated sensors in ascending order of their remaining energy. The sensor with less remaining energy thus has a higher chance to get its sensing range reduced, which eases the energy consumption of that sensor and possibly brings longer network lifetime. The reduction operation corresponds to steps 12–14 in Algorithm 3.

One note is that those randomly generated individuals in the initial population may also not ndCSs, but the remaining energy of sensors is not available since the LP model has not been solved. In this case, we simply use a random repair strategy. That is, randomly select sensors to monitor the unmonitored targets and follow a random order to check if each sensor can reduce its sensing range or be inactivated.

E. Overall Procedure of NEDA

The overall procedure of NEDA is shown in Algorithm 4. First, the population of N individuals is randomly generated.

Algorithm 4 NEDA

1: Randomly initialize the population $\boldsymbol{P}$ of N individuals;
2: Solve the LP model in (2)∼(4) with $\boldsymbol{P}$ as input;
3: Evaluate the individuals' fitness through LPFE;
4: Obtain $\boldsymbol{SI}$ according to (6);
5: Sample N–$|\boldsymbol{SI}|$ new individuals according to NSS, denoted by $\boldsymbol{SP}$;
6: $\boldsymbol{P} = \boldsymbol{SI} \bigcup \boldsymbol{SP}$;
7: Terminate if the termination criterion is met, otherwise jump back to Step 2;

Each individual, if not an ndCS, is repaired by the random repair strategy. The next step is solving the LP model in (2)–(4) with the population as input. After that, the individuals are evaluated through LPFE and $\boldsymbol{SI}$ is obtained according to (6). (N–$|\boldsymbol{SI}|$) new individuals are then sampled by NSS and repaired by HRS if necessary. The new individuals and those in the $\boldsymbol{SI}$ form the population of the next generation. Note that by keeping all the individuals in $\boldsymbol{SI}$ for the next generation, NEDA is able to preserve the best solution found so far.

The population size N in NEDA is adaptively set to $2 \times |\boldsymbol{S}|$ according to the number of sensors in WSNs. Based on the theory of LP, if an LP model has c constraints (excluding the non-negativity constraints), the number of variables in a feasible basis will be no larger than c [34], [35]. The LP model in (2)–(4) has $|\boldsymbol{S}|$ constraints apart from the non-negativity constraints. Therefore, at most $|\boldsymbol{S}|$ ndCSs are used (activated) in the solution of the LP model. Our NEDA has a population of $2 \times |\boldsymbol{S}|$ individuals, which can provide sufficient new ndCSs to avoid premature convergence. In extreme conditions that $|\boldsymbol{SI}| = |\boldsymbol{S}|$, NEDA can still generate $|\boldsymbol{S}|$ new ndCSs.

IV. Experimental Study

A. Experimental Settings

Two groups of instances are used to test the performance of the proposed NEDA. The first group contains 15 types of small-scale instances, where the number of sensors $|\boldsymbol{S}|$ varies in the set of {200, 300, 400} and the number of targets $|\boldsymbol{T}|$ varies in the set of {10, 20, 30, 40, 50}. The second group contains one type of large-scale instances, where $|\boldsymbol{S}| = 1000$ and $|\boldsymbol{T}| = 100$. For each type of instances, ten different instances are generated with sensors and targets randomly scattered in a 100×100 area. We name the instances as $|\boldsymbol{S}|\_|\boldsymbol{T}|\_l$, where l is the index of the instance in the type specified by $|\boldsymbol{S}|$ and $|\boldsymbol{T}|$. Each sensor has ten selectable sensing ranges, increasing from 2 to 20 with a step of 2. The initial battery energy of sensor is 1 and the energy consumption rate of the largest sensing range is also 1. That is, the sensor can be activated for 1 h at the sensing range of 20. The energy consumption rates of the other sensing ranges are computed according to (1).

We compare NEDA with three approaches, CGGA [23], AR-Iterative [24], and Greedy. Herein, CGGA and AR-Iterative are two state-of-the-art approaches proposed in [23] and [24], respectively. Greedy is a simple heuristic algorithm that acts as the baseline. It has two principles: 1) the target whose distance from its nearest available sensor is larger has higher priority to be handled and 2) the available

Algorithm 5 Greedy Algorithm

1: $\boldsymbol{AS} = \boldsymbol{S}$; $NL = 0$;
2: **While** the sensors in $\boldsymbol{AS}$ can achieve full target coverage
3: Sort the targets in descending order of the distance from their nearest available sensor;
4: **For each** target t_k in the order obtained by Step 3
5: **If** t_k is not monitored
6: Activate t_k's nearest sensor in $\boldsymbol{AS}$ at the minimum sensing range that can monitor t_k;
7: Update the coverage information;
8: **End If**
9: **End For**
10: Calculate the maximum activation time of the current coverage scheme, denoted by CL;
11: $NL = NL + CL$;
12: Update $\boldsymbol{AS}$ and the remaining energy of sensors;
13: **End While**
14: Output NL;

sensor nearest to the target is selected to monitor it. Herein, available sensors are those that have remaining energy. The detailed procedure of Greedy is shown in Algorithm 5. $\boldsymbol{AS}$ represents the set of available sensors. In steps 3–9, a coverage scheme is constructed according to the above two principles. In step 7, the selected sensor may also be able to monitor the other target(s) apart from t_k, so that the coverage information is updated immediately. Since the target that has larger distance from its nearest available sensor is handled at first, the coverage scheme obtained is guaranteed to be an ndCS. After obtaining an ndCS, its activation time is set at the maximum according to the remaining energy of the sensors (step 10). Then, the network lifetime NL is updated (step 11). In addition, $\boldsymbol{AS}$ and the remaining energy of sensors are also updated (step 12). The algorithm follows an iterative procedure and terminates until the sensors in $\boldsymbol{AS}$ are unable to achieve full target coverage.

NEDA and the compared algorithms are all iterative approaches, but it may not be fair to use the same number of iterations due to the difference in their iterative mechanisms. Instead, we force the algorithms to terminate within the same execution time on identical platforms. The maximum execution time is set to 1 and 20 h for small-scale and large-scale instances, respectively. All the experiments are conducted on PCs with Ubuntu 16.04 system, Intel i7-7700 CPU and 8-GB RAM. Among the tested algorithms, NEDA and CGGA involve stochastic factors, so that these two algorithms are run for ten independent times on each instance and the average results are presented. Unless otherwise stated, the differences between the experimental results are validated by Wilcoxon rank-sum test at the significance level of 0.05.

B. Experimental Results

1) Small-Scale Instances: Table I tabulates the average lifetime obtained on the ten instances of each small-scale instance type. The best results on each instance type are denoted in bold. The numbers of the format (a/b/c) show that comparing NEDA with the corresponding algorithm, NEDA performs significantly better on a instances, shows no significant difference on b instances, and performs significantly worse on c instances. "Total" sums up a, b, and c across all the 15 instance types for each compared algorithm.

TABLE I
EXPERIMENTAL RESULTS OF LIFETIME ON SMALL-SCALE INSTANCES

Instance Type	NEDA	CGGA	AR-Iterative	Greedy
200_10	**31.949**	**31.949** (0/10/0)	31.673 (7/3/0)	29.485 (7/3/0)
200_20	**33.452**	**33.452** (0/10/0)	32.955 (5/5/0)	31.398 (5/5/0)
200_30	**37.325**	**37.325** (0/10/0)	36.481 (8/2/0)	34.687 (8/2/0)
200_40	**35.448**	**35.448** (0/10/0)	34.893 (8/2/0)	32.456 (8/2/0)
200_50	**39.486**	**39.486** (0/10/0)	39.077 (7/3/0)	37.489 (8/2/0)
300_10	**56.041**	**56.041** (0/10/0)	55.711 (7/3/0)	52.394 (7/3/0)
300_20	**48.296**	**48.296** (0/10/0)	47.150 (10/0/0)	43.278 (10/0/0)
300_30	**45.374**	**45.374** (0/10/0)	44.239 (7/3/0)	40.602 (7/3/0)
300_40	**55.372**	**55.372** (0/10/0)	53.494 (7/3/0)	49.779 (8/2/0)
300_50	**44.587**	**44.587** (0/10/0)	43.632 (7/3/0)	40.430 (7/3/0)
400_10	**109.014**	**109.014** (0/10/0)	108.113 (8/2/0)	106.102 (8/2/0)
400_20	**73.195**	**73.195** (0/10/0)	71.642 (10/0/0)	64.286 (10/0/0)
400_30	**66.448**	**66.448** (0/10/0)	64.664 (9/1/0)	60.476 (9/1/0)
400_40	**68.476**	**68.476** (0/10/0)	65.009 (10/0/0)	59.248 (10/0/0)
400_50	**55.592**	**55.592** (0/10/0)	53.562 (8/2/0)	49.068 (8/2/0)
Total		0/150/0	118/32/0	120/30/0

Compared with AR-Iterative and Greedy, NEDA performs better on all instance types. In detail, NEDA performs significantly better than AR-Iterative on 118 out of the 150 small-scale instances, and better than Greedy on 120 instances. Moreover, AR-Iterative and Greedy do not show a significant advantage over NEDA on any instance. Comparing NEDA with CGGA, we can see that their average results are the same on all small-scale instances. That is, given a maximum execution time of 1 h, NEDA and CGGA achieve the same result on each instance of the 15 small-scale instance types. The reason for getting the same results is probably that NEDA and CGGA both have the ability to sufficiently explore the search space of small-scale instances and both find the global optimal solution.

To make a more detailed comparison between NEDA and CGGA, the "Time to Best" metric is employed and the experimental results are shown in Table II. The Time to Best metric represents the execution time consumed by the algorithm to reach its best solution. The same as in Table I, the average results on the ten instances of each instance type are presented. The unit of execution time is second. In the last row of the table, the Wilcoxon rank-sum tests show that NEDA performs significantly better than CGGA on all 150 instances. The last column shows the reduced time of NEDA to reach the best solution compared

TABLE II
COMPARISON OF "TIME TO BEST" BETWEEN NEDA AND CGGA ON SMALL-SCALE INSTANCES (THE UNIT OF EXECUTION TIME IS SECOND)

Instance Type	NEDA	CGGA	Reduced time of NEDA
200_10	0.520	3.978 (10/0/0)	**3.458**
200_20	0.766	7.248 (10/0/0)	**6.482**
200_30	1.152	10.666 (10/0/0)	**9.514**
200_40	0.913	12.240 (10/0/0)	**11.327**
200_50	0.755	11.283 (10/0/0)	**10.528**
300_10	1.248	14.530 (10/0/0)	**13.282**
300_20	8.594	78.257 (10/0/0)	**69.663**
300_30	10.549	83.539 (10/0/0)	**72.990**
300_40	17.663	157.153 (10/0/0)	**139.490**
300_50	6.520	129.536 (10/0/0)	**123.016**
400_10	7.361	71.611 (10/0/0)	**64.250**
400_20	43.281	291.629 (10/0/0)	**248.348**
400_30	342.954	1290.871 (10/0/0)	**947.917**
400_40	152.089	787.189 (10/0/0)	**635.100**
400_50	94.405	599.005 (10/0/0)	**504.600**
Total		150/0/0	

with CGGA. We can observe that NEDA has comprehensively better efficiency on all 15 instance types. Particularly on 400_30, NEDA is 15 min faster than CGGA to reach the best solution. These results indicate that although CGGA can achieve competitive results, it suffers from poor time efficiency.

2) Large-Scale Instances: The average lifetime obtained by the four algorithms on the ten large-scale instances are shown in Table III, with the best results denoted in bold. The values in the parentheses show the improvement rate of NEDA compared with the other algorithms. The last row summarizes the number of instances on which NEDA performs significantly better, shows no significant difference, or performs significantly worse compared with the corresponding algorithm.

From the table, the proposed NEDA achieves the best results on eight out of the ten large-scale instances. Only on 1000_100_2 and 1000_100_7 is the performance of NEDA worse than AR-Iterative, with the differences of -3.811% and -2.227%, respectively. However, NEDA performs significantly better than AR-Iterative on seven instances, particularly on 1000_100_9 and 1000_100_10, where NEDA's improvement rates against AR-Iterative are 5.092% and 7.883%, respectively. CGGA suffers from poor search ability on large-scale WSNs. We can see that NEDA performs better than CGGA on all the instances, with an improvement rate beyond 15% on seven out of the ten instances. In particular, the improvement rate of NEDA on 1000_100_8 reaches

TABLE III
EXPERIMENTAL RESULTS OF LIFETIME ON LARGE-SCALE INSTANCES

Instance	NEDA	CGGA	AR-Iterative	Greedy
1000_100_1	**158.939**	135.638 (+17.179%)	153.439 (+3.585%)	138.382 (+14.855%)
1000_100_2	162.399	137.487 (+18.120%)	**168.833** (−3.811%)	140.027 (+15.977%)
1000_100_3	**156.161**	128.453 (+21.571%)	152.435 (+2.445%)	126.138 (+23.802%)
1000_100_4	**143.865**	137.899 (+4.326%)	139.144 (+3.393%)	127.987 (+12.405%)
1000_100_5	**154.584**	133.248 (+16.012%)	153.671 (+0.594%)	144.572 (+6.925%)
1000_100_6	**119.627**	119.590 (+0.031%)	115.934 (+3.186%)	100.936 (+18.518%)
1000_100_7	159.961	128.932 (+24.066%)	**163.604** (−2.227%)	148.652 (+7.607%)
1000_100_8	**163.811**	126.720 (+29.270%)	156.624 (+4.589%)	142.546 (+14.918%)
1000_100_9	**128.925**	126.768 (+1.701%)	122.678 (+5.092%)	119.903 (+7.524%)
1000_100_10	**168.305**	139.753 (+20.431%)	156.007 (+7.883%)	151.057 (+11.418%)
Total		10/0/0	7/1/2	10/0/0

29.270%. Compared with Greedy, NEDA has comprehensive advantage on all instances.

Overall, NEDA has better performance than the other three tested algorithms on both small-scale and large-scale instances. There are mainly two reasons for this. First, NEDA benefits from the LP model in (2)–(4), which can obtain the optimal activation time of each ndCS to efficiently utilize sensor energy and maximize the network lifetime. Second, with LPFE, NSS, and HRS, NEDA can efficiently generate a high-quality set of ndCSs to prolong the network lifetime.

C. Effectiveness of LPFE

In NEDA, the LPFE evaluates the fitness of individuals based on the LP model in (2)–(4). To investigate the effectiveness of LPFE, a greedy fitness evaluation (GFE) is used for comparison. GFE defines the fitness of each individual as the maximum activation time of its corresponding ndCS. Note that in NEDA, those individuals whose fitness is larger than zero are selected to construct the $\boldsymbol{SI}$ for NSS. However, every individual's fitness is larger than zero by the GFE. In this case, $|\boldsymbol{S}|$ fittest individuals (random break in case of tie) are selected to construct the $\boldsymbol{SI}$.

Tables IV and V show the experimental results of these two fitness evaluation strategies on small-scale and large-scale instances, respectively. The better results are bolded. The numbers of the format (a/b/c) show that compared with GFE, LPFE performs significantly better on a instances, shows no significant difference on b instances, and performs significantly worse on c instances. In Table V, the values in the parentheses show the improvement rate of LPFE compared with GFE. From Table IV, we can see that LPFE performs comprehensively better than GFE on all instance types. In total, LPFE performs better on 123 out of the 150 small-scale instances. The advantage of LPFE is very obvious on large-scale instances, as shown in Table V. The reason for

TABLE IV
EXPERIMENTAL RESULTS OF LIFETIME OBTAINED BY LPFE AND GFE ON SMALL-SCALE INSTANCES

Instance Type	LPFE	GFE	Instance	LPFE	GFE
200_10	**31.949**	29.779 (6/4/0)	300_40	**55.372**	40.454 (9/1/0)
200_20	**33.452**	28.486 (6/4/0)	300_50	**44.587**	37.044 (8/2/0)
200_30	**37.325**	31.981 (7/3/0)	400_10	**109.014**	74.927 (9/1/0)
200_40	**35.448**	31.394 (8/2/0)	400_20	**73.195**	51.988 (10/0/0)
200_50	**39.486**	35.814 (8/2/0)	400_30	**66.448**	43.764 (9/1/0)
300_10	**56.041**	51.465 (7/3/0)	400_40	**68.476**	43.603 (9/1/0)
300_20	**48.296**	38.765 (9/1/0)	400_50	**55.592**	42.545 (9/1/0)
300_30	**45.374**	37.971 (9/1/0)	Total	123/27/0	

TABLE V
EXPERIMENTAL RESULTS OF LIFETIME OBTAINED BY LPFE AND GFE ON LARGE-SCALE INSTANCES

Instance	LPFE	GFE
1000_100_1	**158.939**	90.184 (+76.240%)
1000_100_2	**162.399**	90.318 (+79.807%)
1000_100_3	**156.161**	83.485 (+87.054%)
1000_100_4	**143.865**	90.717 (+58.587%)
1000_100_5	**154.584**	88.522 (+74.627%)
1000_100_6	**119.627**	86.142 (+38.871%)
1000_100_7	**159.961**	88.621 (+80.499%)
1000_100_8	**163.811**	88.696 (+84.687%)
1000_100_9	**128.925**	89.819 (+43.538%)
1000_100_10	**168.305**	98.187 (+71.413%)
Total		10/0/0

LPFE's effectiveness is as follows. The LP model can obtain the optimal activation time of each ndCS to maximize the network lifetime. Using the optimal activation time as the fitness can show each ndCS's contribution to the network lifetime, which helps guide the generation of more promising ndCSs. However, GFE's greedy principle makes it easy to find a local optimum.

D. Effectiveness of NSS

In this section, we investigate the performance of NEDA with different settings of the neighborhood size n in NSS, including $n = 5$, 10, 20, and $|\boldsymbol{SI}|$. Note that with the setting of $n = |\boldsymbol{SI}|$, NEDA actually omits the use of NSS and directly use $\boldsymbol{SI}$ to construct the univariate probabilistic model from which all new individuals are sampled. Thus, we term this setting as "no NSS." The experimental results show that the lifetime obtained by these four settings are not significantly different on all small-scale instances. Due to the page limit, we only present the convergence curves of NEDA under these four settings on some typical instances, as shown in Fig. 5.

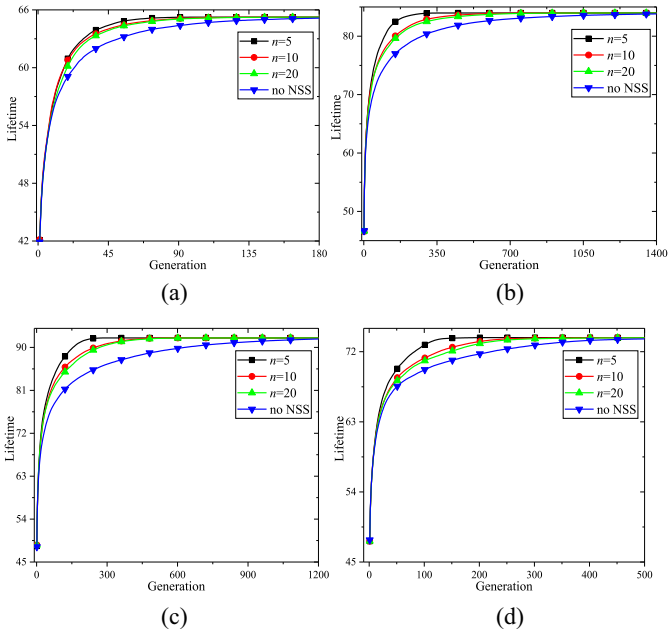

Fig. 5. Convergence curves of different neighborhood size settings on small-scale instances (a) 300_30_4, (b) 400_30_8, (c) 400_40_7, and (d) 400_50_7.

TABLE VI
EXPERIMENTAL RESULTS OF LIFETIME OBTAINED BY DIFFERENT NEIGHBORHOOD SIZE SETTINGS ON LARGE-SCALE INSTANCES

Instance	n=5	n=10	n=20	No NSS
1000_100_1	**158.939**	128.692 (+23.504%)	127.623 (+24.538%)	127.943 (+24.227%)
1000_100_2	**162.399**	125.935 (+28.954%)	124.702 (+30.230%)	125.332 (+29.575%)
1000_100_3	**156.161**	126.312 (+23.631%)	125.581 (+24.351%)	125.915 (+24.022%)
1000_100_4	**143.865**	129.006 (+11.518%)	128.097 (+12.309%)	128.632 (+11.842%)
1000_100_5	**154.584**	124.411 (+24.252%)	123.136 (+25.539%)	123.986 (+24.679%)
1000_100_6	**119.627**	**119.627** (≈)	119.614 (+0.011%)	**119.627** (≈)
1000_100_7	**159.961**	126.730 (+26.222%)	125.607 (+27.350%)	126.511 (+26.440%)
1000_100_8	**163.811**	126.717 (+29.274%)	125.493 (+30.534%)	126.192 (+29.811%)
1000_100_9	**128.925**	124.598 (+3.473%)	123.812 (+4.130%)	124.090 (+3.896%)
1000_100_10	**168.305**	133.400 (+26.165%)	132.381 (+27.137%)	133.279 (+26.280%)
Total		9/1/0	10/0/0	9/1/0

From these figures, we can see that although the differences in the obtained lifetime are not statistically significant, NSS does help accelerate the convergence. The reason is that NSS helps generate diverse ndCSs to efficiently utilize the energy of each sensor. As the neighborhood size increases, the acceleration effect is gradually compromised, implying that a larger neighborhood size may weaken the effectiveness of NSS in diversity improvement and thus reduce the efficiency for finding an ndCS set with longer lifetime.

The experimental results on large-scale instances are shown in Table VI, with the best results on each instance bolded. The

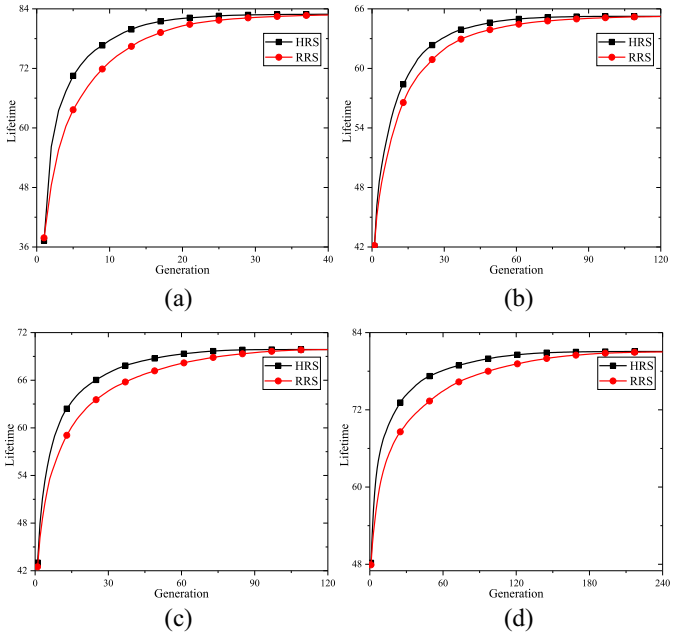

Fig. 6. Convergence curves of HRS and RRS on small-scale instances (a) 300_20_1, (b) 300_30_4, (c) 400_30_9, and (d) 400_40_10.

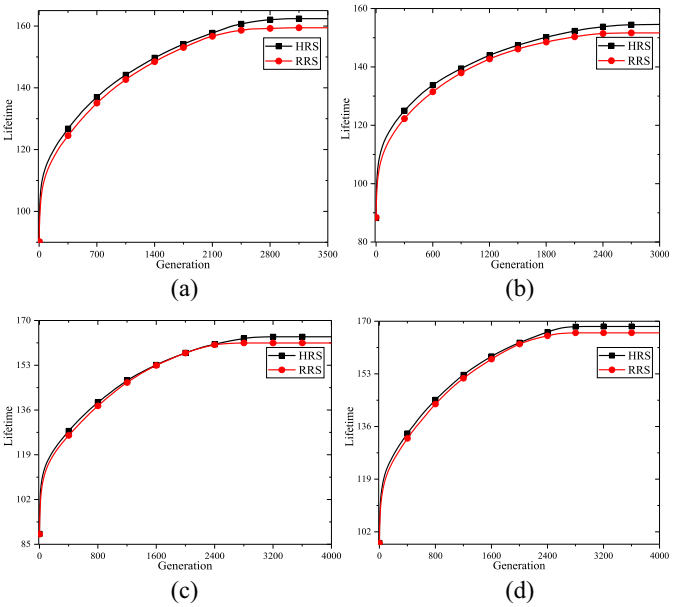

Fig. 7. Convergence curves of HRS and RRS on large-scale instances (a) 1000_100_2, (b) 1000_100_5, (c) 1000_100_8, and (d) 1000_100_10.

values in the parentheses show the improvement rate of the adopted setting, that is, $n = 5$, compared with the other tested settings. The comparison results on all ten tested instances are counted and presented in the bottom of each compared setting's column. We can see that apart from 1000_100_6 where the four settings have closed performance, the NSS with $n = 5$ can greatly improve the obtained results.

In conclusion, NSS can improve the search efficiency by sampling new individuals based on neighborhood. The reason is that the LM-RAS problem requires a set of diverse ndCSs to efficiently utilize sensors' energy and NSS helps conduct the distributed search to enhance the population diversity. Moreover, on both small-scale and large-scale instances, the adopted setting of n, that is, $n = 5$, performs the best among all tested settings.

E. Effectiveness of HRS

To validate the effectiveness of HRS, we compare the performance of HRS with the random repair strategy (denoted by RRS). Compared with HRS, RRS does not use the remaining energy of the sensors as the heuristic information, but randomly selects the sensors to monitor unmonitored targets and checks if each activated sensor can reduce its sensing range or be inactivated in a random order. The convergence curves on some small-scale instances are plotted in Fig. 6. We can see that HRS accelerates the convergence. The convergence curves on some large-scale instances are plotted in Fig. 7. We can see that HRS can not only accelerate the convergence but also improve the obtained lifetime. The above results confirm that the proposed HRS benefits from the usage of heuristic information, that is, the remaining energy of sensors, and helps improve the search efficiency of NEDA to prolong the network lifetime.

V. Conclusion

In this article, a novel NEDA is proposed to deal with the LM-RAS problem. Each individual in NEDA represents an ndCS. Based on the ndCS set represented by the population, an LP model is built to find the optimal activation time of each ndCS for maximizing the network lifetime. To generate more promising ndCSs for prolonging the network lifetime, three novel strategies, i.e., LPFE, NSS, HRS, are incorporated into NEDA. Individual fitness is evaluated by LPFE that shows an individual's contribution to the network lifetime. The NSS uses neighborhoods to construct the probabilistic models for sampling new individuals, which helps generate diverse ndCSs. The HRS uses the remaining energy of the sensors as heuristic information to fine-tune the coverage schemes by the refinement and reduction operations. Experimental results on WSN instances of different scales showed that the proposed NEDA outperforms the state-of-the-art approaches in terms of solution quality and efficiency. The LM-RAS problem is further generalized as a kind of problem that shares the same structure, namely, fLP problems. The NEDA is supposed to be an effective framework for solving fLP problems, which will be further examined in future work.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.

[2] B. Wang, X. Deng, W. Liu, L. T. Yang, and H. Chao, "Confident information coverage in sensor networks for field reconstruction," *IEEE Wireless Commun.*, vol. 20, no. 6, pp. 74–81, Dec. 2013.

[3] H. Huang, T. Gong, R. Zhang, L. Yang, J. Zhang, and F. Xiao, "Intrusion detection based on k-coverage in mobile sensor networks with empowered intruders," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12109–12123, Dec. 2018.

[4] H. Yetgin, K. T. K. Cheung, M. El-Hajjar, and L. H. Hanzo, "A survey of network lifetime maximization techniques in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 828–854, 2nd Quart., 2017.
[5] A. A. Aziz, Y. A. Sekercioglu, P. Fitzpatrick, and M. Ivanovich, "A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 121–144, 1st Quart., 2013.
[6] D. Ye and M. Zhang, "A self-adaptive sleep/wake-up scheduling approach for wireless sensor networks," *IEEE Trans. Cybern.*, vol. 48, no. 3, pp. 979–992, Mar. 2018.
[7] F. Li, J. Luo, W. Wang, and Y. He, "Autonomous deployment for load balancing *k*-surface coverage in sensor networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 279–293, Jan. 2015.
[8] J. Yu, S. Wan, X. Cheng, and D. Yu, "Coverage contribution area based *k*-coverage for wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 9, pp. 8510–8523, Sep. 2017.
[9] C. Sun, "A time variant log-linear learning approach to the SET K-COVER problem in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 48, no. 4, pp. 1316–1325, Apr. 2018.
[10] D. Kim, W. Wang, J. Son, W. Wu, W. Lee, and A. O. Tokuta, "Maximum lifetime combined barrier-coverage of weak static sensors and strong mobile sensors," *IEEE Trans. Mobile Comput.*, vol. 16, no. 7, pp. 1956–1966, Jul. 2017.
[11] Z. Wang, J. Liao, Q. Cao, H. Qi, and Z. Wang, "Achieving k-barrier coverage in hybrid directional sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 7, pp. 1443–1455, Jul. 2014.
[12] W. Wang, V. Srinivasan, B. Wang, and K.-C. Chua, "Coverage for target localization in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 2, pp. 667–676, Feb. 2008.
[13] Z. Lu, W. W. Li, and M. Pan, "Maximum lifetime scheduling for target coverage and data collection in wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 2, pp. 714–727, Feb. 2015.
[14] G. Han, L. Liu, J. Jiang, L. Shu, and G. Hancke, "Analysis of energy-efficient connected target coverage algorithms for industrial wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 13, no. 1, pp. 135–143, Feb. 2017.
[15] C. Liao and C. Ting, "A novel integer-coded memetic algorithm for the set *k*-cover problem in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 48, no. 8, pp. 2245–2258, Aug. 2018.
[16] X. Wan, J. Wu, and X. Shen, "Maximal lifetime scheduling for roadside sensor networks with survivability *k*," *IEEE Trans. Veh. Technol.*, vol. 64, no. 11, pp. 5300–5313, Nov. 2015.
[17] X. Y. Zhang, J. Zhang, Y.-J. Gong, Z.-H. Zhan, W.-N. Chen, and Y. Li, "Kuhn–Munkres parallel genetic algorithm for the set cover problem and its application to large-scale wireless sensor networks," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 695–710, Oct. 2016.
[18] C. Chen *et al.*, "A hybrid memetic framework for coverage optimization in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2309–2322, Oct. 2015.
[19] H. Mohamadi, S. Salleh, and M. N. Razali, "Heuristic methods to maximize network lifetime in directional sensor networks with adjustable sensing ranges," *J. Netw. Comput. Appl.*, vol. 46, pp. 26–35, Nov. 2014.
[20] K. M. Alam, J. Kamruzzaman, G. Karmakar, and M. Murshed, "Dynamic adjustment of sensing range for event coverage in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 46, pp. 139–153, Nov. 2014.
[21] C. Chang, C. Chang, S. Zhao, J. Chen, and T. Wang, "SRA: A sensing radius adaptation mechanism for maximizing network lifetime in WSNs," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9817–9833, Dec. 2016.
[22] X. Zhang, Y. Zhou, Q. Zhang, V. C. S. Lee, and M. Li, "Problem specific MOEA/D for barrier coverage with wireless sensors," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3854–3865, Nov. 2017.
[23] A. Rossi, A. Singh, and M. Sevaux, "An exact approach for maximizing the lifetime of sensor networks with adjustable sensing ranges," *Comput. Oper. Res.*, vol. 39, no. 12, pp. 3166–3176, 2012.
[24] R. Cerulli, R. De Donato, and A. Raiconi, "Exact and heuristic methods to maximize network lifetime in wireless sensor networks with adjustable sensing ranges," *Eur. J. Oper. Res.*, vol. 220, no. 1, pp. 58–66, 2012.
[25] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, 1984.
[26] M. Todd, "The many facets of linear programming," *Math. Program.*, vol. 91, no. 3, pp. 417–436, 2002.
[27] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Boston, MA: Kluwer Academic, 2002.
[28] Z.-G. Chen, Z.-H. Zhan, H. Wang, and J. Zhang, "Distributed individuals for multiple peaks: A novel differential evolution for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, early access, doi: 10.1109/TEVC.2019.2944180.
[29] P. Yang, K. Tang, and X. Lu, "Improving estimation of distribution algorithm on multimodal problems by detecting promising areas," *IEEE Trans. Cybern.*, vol. 45, no. 8, pp. 1438–1449, Aug. 2015.
[30] Z.-G. Chen *et al.*, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019.
[31] B. Yuan, B. Li, H. Chen, and X. Yao, "A new evolutionary algorithm with structure mutation for the maximum balanced biclique problem," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 1054–1067, May 2015.
[32] J. Wang, K. Tang, J. A. Lozano, and X. Yao, "Estimation of the distribution algorithm with a stochastic local search for uncertain capacitated arc routing problems," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 96–109, Feb. 2016.
[33] S. Wang and L. Wang, "An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem," *IEEE Trans. Syst., Man, and Cybern. Syst.*, vol. 46, no. 1, pp. 139–149, Jan. 2016.
[34] V. Chvátal, *Linear Programming*. New York, NY, USA: Freeman, 1983.
[35] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Belmont, MA, USA: Athena Sci., 1997.

Zong-Gan Chen (Student Member, IEEE) received the B.S. degree from Sun Yat-sen University, Guangzhou, China, in 2016. He is currently pursuing the Ph.D. degree in computer science and technology with the South China University of Technology, Guangzhou.

His current research interests include estimation of distribution algorithm, ant colony optimization, differential evolution, and their applications in real-world optimization problems.

Ying Lin (Member, IEEE) received the Ph.D. degree in computer applied technology from Sun Yat-sen University, Guangzhou, China, in 2012.

She is currently an Associate Professor with the Department of Psychology, Sun Yat-sen University and also a Research Fellow with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, South China University of Technology, Guangzhou. Her current research interests include computational intelligence and its applications in network analysis, cognitive diagnosis, and cloud computing.

Yue-Jiao Gong (Senior Member, IEEE) received the B.S. and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2010 and 2014, respectively.

She is currently a Full Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. Her research interests include evolutionary computation, swarm intelligence, and their applications to intelligent transportation and smart city scheduling. She has published over 80 papers, including more than 30 IEEE TRANSACTIONS papers, in the above areas.

Zhi-Hui Zhan (Senior Member, IEEE) received the bachelor's and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

From 2013 to 2015, he was a Lecturer and an Associate Professor with the Department of Computer Science, Sun Yat-sen University. Since 2016, he has been a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, where he is also the Changjiang Scholar Young Professor and the Pearl River Scholar Young Professor. His current research interests include evolutionary computation, swarm intelligence, and their applications in real-world problems and in environments of cloud computing and big data.

Prof. Zhan's doctoral dissertation was awarded the China Computer Federation Outstanding Ph.D. Dissertation and the IEEE Computational Intelligence Society Outstanding Ph.D. Dissertation. He was a recipient of the Outstanding Youth Science Foundation from the National Natural Science Foundations of China in 2018 and the Wu Wen Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017. He is listed as one of the Most Cited Chinese Researchers in Computer Science. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *Neurocomputing*, and the *International Journal of Swarm Intelligence Research*.

Jun Zhang (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Visiting Scholar with Victoria University, Melbourne, VIC, Australia. His current research interests include computational intelligence, cloud computing, high-performance computing, operations research, and power electronic circuits.

Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, 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 TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, and the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.