A Multi-Objective Evolutionary Algorithm for the Deployment and Power Assignment Problem in Wireless Sensor Networks

Andreas Konstantinidis[a,*], Kun Yang[a], Qingfu Zhang[a], Demetrios Zeinalipour-Yazti[b]

[a] *School of Computer Science and Electronic Engineering, University of Essex, Colchester, CO4 3SQ, UK*
[b] *Department of Computer Science, University of Cyprus, 1678 Nicosia, Cyprus*

Abstract

A Wireless Sensor Network (WSN) design often requires the decision of optimal locations (deployment) and transmit power levels (power assignment) of the sensors to be deployed in an area of interest. Few attempts have been made on optimizing both decision variables for maximizing the network coverage and lifetime objectives, even though, most of the latter studies consider the two objectives individually. This paper defines the multiobjective Deployment and Power Assignment Problem (DPAP). Using the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D), the DPAP is decomposed into a set of scalar subproblems that are classified based on their objective preference and tackled in parallel by using neighborhood information and problem-specific evolutionary operators, in a single run. The proposed operators adapt to the requirements and objective preferences of each subproblem dynamically during the evolution, resulting in significant improvements on the overall performance of MOEA/D. Simulation results have shown the superiority of the problem-specific MOEA/D against the NSGA-II in several network instances, providing a diverse set of high quality network designs to facilitate the decision maker's choice.

Key words: deployment, power assignment, sensor networks, multiobjective optimization, evolutionary algorithms

1. Introduction and background

The design of Wireless Sensor Networks (WSNs) [1] is a highly complicated task with substantial impact on the quality, cost and efficiency of real life sensor applications. Sensors are small electronic devices with limited energy, memory and transmit power capabilities, which in some sensor network applications are also limited in number because of their high cost [2]. A typical goal of these network designs is to form a long lived WSN, such that the sensors, using their sensing capabilities and wireless transceivers, effectively cover a region of interest and forward important information to a common collection point, usually referred to as a data sink.

For most WSNs, a major design step is to selectively decide the locations of the sensors to maximize the covered area of a targeted region. This particular problem has different appellations in the literature, e.g. placement, coverage

*Corresponding author: Tel.:+44 (0)1206 87-3803. Fax: +44 (0)1206 87-2900
Email addresses: akonst@essex.ac.uk (Andreas Konstantinidis), kunyang@essex.ac.uk (Kun Yang), qzhang@essex.ac.uk (Qingfu Zhang), dzeina@cs.ucy.ac.cy (Demetrios Zeinalipour-Yazti)

or deployment problem in WSNs, where Younis et al. [3] provide a good overview of various strategies. Many practitioners, such as Meguerdichian et al. [4], have pointed out that the coverage objective is a measure of the quality of service (QoS) that is provided by a particular network design. Several researchers [5, 6] have proven the NP-hardness of various deployment problems. The main focus is often to determine an optimal sensor placement to cover a grid area (sometimes under uncertainty [6]) and minimize the cost or prolong the network lifetime [7]. None of the above studies, however, have considered an energy-aware transmit power level assignment for maximizing the network lifetime while tackling a deployment problem. In sensor network applications, where the number of available sensors is limited and the desired area is large, the sensors usually operate under high transmit power levels, which is the dominant parameter of their total energy consumption. This often results on a premature exhaustion of their initial power supply. Thus, another major step in WSN design is to assign energy efficient transmit power levels to the sensors to maximize the network lifetime under certain energy constraints [8]. This problem is commonly known as the power/range assignment problem in WSNs [9] and it is proven NP-hard by [10], with the objective of minimizing the total communication ranges of the network. The same problem, while maintaining the connectivity [11, 12], is proven NP Complete by Cheng et al. in [13], where Santi et al. [14] solved it by means of a probabilistic approach.

Few researchers, however, have tackled the Deployment and Power Assignment Problems (*DPAP*s) in WSNs, simultaneously. For example, Cheng et al. [15] and Liu et al. [2] have studied various formats of the DPAP. Their main focus was to determine both the locations of the sensors and the distance between the sensors (transmission range) for maximizing the network lifetime or minimizing the deployment cost under certain coverage requirements, or maximize the covered area given a fixed number of sensors and a desired lifetime. In both cases, the authors have analytically tackled the problem for linear networks and they have proposed several heuristic methods for planar networks. The goal of the DPAP, tackled by Chen et al. [16], was to find the minimum number of sensors, the optimal sensor placement and the transmission structure for maximizing a utilization efficiency objective, which is defined as the network lifetime per unit deployment cost. The authors proposed an approximation approach where the placement of the sensors ensures a fully covered area and a simultaneous depletion of their initial power supply. All the latter approaches, however, optimize the lifetime and coverage objectives individually and sequentially, or by constraining one and optimizing the other. This often results in ignoring and losing "better" solutions since coverage and lifetime are conflicting objectives and a decision maker needs an optimal trade-off. Thus, there is not a single solution that can optimize both objectives simultaneously, but a set of equally important solutions.

Because so many different aspects are involved, the respective DPAPs are a proper object for Multi-Objective Optimization (MOO). Considering a maximization Multiobjective Optimization Problem (MOP) with k objectives, a solution X^* is considered non-dominated or Pareto optimal with respect to another solution Y, if $\forall i \in \{1, ..., k\}, X_i \geq Y_i \wedge \exists i \in \{1, ..., k\} : X_i > Y_i$, this is denoted as $X \succ Y$. The set of all Pareto optimal or non-dominated solutions in the search space, also called Pareto Set (PS), is often mapped to a Pareto Front (PF) in the objective space [17].

In [18], we introduced the multi-objective DPAP in WSNs, which is typical for applications that invoke a limited number of expensive sensors, where their operation is significantly affected by their position and communication

[19]. In these cases, the random massive deployment [20] and dynamic power assignment [9] is not the only choice; and the application affords the "luxury" of using a centralized or even an off-line algorithm to compute the locations and transmit power levels of the sensors, prior deployment. Thus, we have considered it critical and challenging to investigate the following problem: for a given surveillance sensing field, determine the locations and the transmit power levels of a limited number of sensors to be deployed, to maximize the network coverage and lifetime, at the same time. By obtaining the PS and PF of the DPAP prior deployment, a critical decision making dilemma is mainly tackled. Namely, "*Having a limited number of available sensors, what portion of the area should be covered and for how long?*". In MOO, a decision maker can analyze both the PS and PF, and then choose the most appropriate Pareto optimal network topology for each scenario, instead of making the decision and then designing the topology. For example, possible choices of the PS in DPAP could be, to employ: (1) a network topology for covering the whole area for a very short time, (2) a long-lived network topology covering a small portion of the area, or alternatively, (3) a network topology that balances the network lifetime and coverage.

In the design and analysis of communication systems and networks, researchers have successfully introduced techniques inspired by other disciplines, such as analogies with physics [21] or natural biology [22] (e.g. Evolutionary Algorithms (EAs) inspired by the biological evolution) for tackling difficult problems, such as DPAP. The successful adaptation of EAs in sensor networks led to the development of several EA-based application specific approaches for WSN design, often dealing with a single objective function [23] or multiple objectives combined to a single objective function [24]. MOO is a relatively new area in WSNs and it is difficult to apply an existing linear/single objective method to effectively tackle the multiobjective DPAP, giving a set of non-dominated solutions. Thus, the DPAP may serve as a real world benchmark for multiobjective methods. The literature hosts several interesting approaches for tackling MOPs, with Multi-Objective Evolutionary Algorithms (MOEAs) posing all the desired characteristics for obtaining a set of non-dominated solutions, in a single run. In the following, we discuss several MOPs in WSNs which are tackled by general purpose MOEAs that utilize standard EA operators [25].

Jourdan and Weck [26] tackled a layout optimization problem in WSNs with the general purpose Multi-objective Genetic Algorithm (MOGA), utilizing a single point crossover and a random mutation for offspring reproduction. In 2005, Rajagopalan and co-workers have formulated a sensor placement problem [27], with main focus on optimizing the sensor locations for maximizing the probability to target detection and minimizing both the energy dissipation and the number of sensors, simultaneously. The authors tackled the MOP with their own Evolutionary Multi-objective Crowding Algorithm (EMOCA), utilizing the general purpose tournament selection, single-point crossover and random mutation operators. In 2007, Oh et al. [28] adopted the Non-Dominated Sorting Genetic Algorithm-II [29] (NSGA-II) to tackle a WSN deployment problem, generating new solutions with the conventional single-point crossover and random mutation operators. More recently (in 2008), Kim et al. [30] have also used NSGA-II to tackle a surveillance sensor placement problem. The authors have utilized the single point crossover for offspring reproduction, adding a restriction, i.e. the common elements in the two parents are not exchanged and a node-exchange mutation. NSGA-II is also used by Jia et al., in 2009, for tackling two multiobjective optimization scheduling prob-

lems [31] and [32]. The authors have utilized the basic tournament selection, two-point crossover and random/swap mutation operators, respectively. Even though the latter problems are considerably different from DPAP, the main difference between their studies and ours is in the way that they treat the MOPs and apply the MOEAs. That is, all the aforementioned studies treat a WSN MOP as a "black box" [33], i.e. without using problem-specific knowledge, which may have undesirable effects, such as forcing the evolutionary process into unnecessary searches and destructive mating, negatively affecting their overall performance. This can be considered as a main drawback of the generic MOEAs when dealing with real life problems (such as DPAP).

Therefore, the incorporation of problem-specific knowledge in MOEAs to direct the search into promising areas of the search space can be proven beneficial [34]. However, designing problem-specific operators for a MOP, as a whole, is difficult. The Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [35] alleviates this difficulty by decomposing the MOP into many scalar subproblems that are optimized in parallel, by using neighborhood information and scalar techniques. The difficulty in adding knowledge on a decompositional MOEA is that the subproblems have different objective preferences, require different treatment and have to be solved simultaneously, in a single run. Therefore, the problem-specific evolutionary operators should adapt to the requirements and objective preference of each subproblem dynamically, during the evolution. In this paper, we propose a MOEA/D-based approach, which strategically monitors problem-specific evolutionary operators and provides different treatment to each subproblem of the DPAP based on its WSN requirements.

The remainder of the paper is organized as follows. Section 2 introduces the multiobjective DPAP, including a DPAP analysis that classifies the Pareto optimal solutions in the objective space based on their objective preferences. Section 3 presents the proposed MOEA/D, i.e. details the problem-specific evolutionary operators. Sections 4 and 5 introduce the experimental setup and the performance metrics, respectively. The simulation results in Section 6 verify the necessity of the proposed operators for improving the performance of MOEA/D with respect to the conventional MOEA/D and the NSGA-II. Section 7 concludes this paper and suggests some topics for future research.

2. The Deployment and Power Assignment Problem (DPAP)

2.1. System model and assumptions

Consider a 2-D static WSN formed by: a rectangular sensing area A, N homogeneous sensors and a static sink H with unlimited energy, placed at the center of A. The sensors are responsible for monitoring and periodically report an event of interest to H. Hence, each sensor i must be able to communicate directly or via multiple hops through nearby sensors with H. We assume a perfect medium access control, such as SMAC [36], which ensures that there are no collisions at any sensor during data communication and we adopt the simple but relevant path loss communication model as in [2]. In this model, the transmit power level that should be assigned to a sensor i to reach a sensor j is $P_i = \beta \times d_{ij}^{\alpha}$, where $\alpha \in [2, 6]$ is the path loss exponent and $\beta = 1$ is the transmission quality parameter. The energy loss due to channel transmission is d_{ij}^{α}, d_{ij} is the Euclidean distance between sensors i and j, $R_c^i = d_{ij}$ is sensor i's communication range, s.t. $R_c^i \leq R_{\text{max}}$, where R_{max} is a fixed maximum communication distance, which

is constrained by the maximum power that sensors can transmit, i.e. $P_{\max}$. The calculated power assignments are considered static for the whole network lifetime. The residual energy of sensor i, at time t, is calculated as follows:

$$E_i(t) = E_i(t-1) - [(r_i(t)+1) \times P_i \times amp] \tag{1}$$

where $(r_i(t)+1)$ is the total traffic load that sensor i forwards towards H at t, $r_i(t)$ is the traffic load that i receives and relays, "$+1$" is the data packet generated by i to forward its own data information and amp is the power amplifier's energy consumption. In DPAP, we assume that the area size A is large and N is relatively small. Consequently, the sensors should spread, to adequately cover A, and communicate through long transmission distances. Thus, the transmit power is a major factor on the sensors' total energy consumption and the energy consumed by the transceiver electronics as well as for reception and generation of data are considered negligible and ignored [2, 16].

For sensing purposes and simplicity, we assume that A is composed of rectangular grids of identical dimensions centered at (x', y') and we adopt a sensing model based on the definite range law approximation [6]. Namely, a grid at (x', y') is covered, denoted by $g(x', y') = 1$, if it falls within a sensor's sensing disk πR_s^2 of radius R_s, otherwise $g(x', y') = 0$. We consider unit-size grids, which are several times smaller than πR_s^2 for a more accurate placement [37].

2.2. Problem formulation

The DPAP can be formulated as a MOP,

Given:

- A: a 2-D plane, where $A = \{(x, y)|0 \leq x \leq 1, 0 \leq y \leq 1\}$.
- N: number of sensors to be deployed in A.
- E: initial power supply, the same for all sensors.
- R_s: sensing range, the same for all sensors.
- $P_{\max}$: maximum transmission power level, the same for all sensors.

Decision variables of a network design X:

- (x_j, y_j) : the location of sensor j.
- P_j : the transmission power level of sensor j.

Objectives: Maximize coverage $Cv(X)$ and lifetime $L(X)$:

The network coverage $Cv(X)$ is defined as the percentage of the covered grids over the total grids of A and is evaluated as follows:

$$Cv(X) = [\sum_{x'=0}^{x} \sum_{y'=0}^{y} g(x', y')]/(x \times y) \tag{2}$$

where, $x \times y$ is the total grids of A and

$$g(x', y') = \begin{cases} 1 & \text{if } \exists j \in \{1, ..., N\}, d_{(x_j, y_j),(x', y')} \leq R_s \\ 0 & \text{otherwise} \end{cases}$$

is the monitoring status of the grid centered at (x', y').

The network lifetime is defined as the duration from the deployment of the network to the cycle t at which a sensor j depletes its energy supply, E [2, 16]. The lifetime objective $L(X)$ is evaluated as in Algorithm 1.

Algorithm 1 Lifetime evaluation

Step 0: Set $t := 1$; $E_j(0) := E, \forall j \in \{1, ..., N\}$;

Step 1: **For** all sensors j at each time interval t **do**

- ***Step 1.1:*** Update $E_j(t)$ according to Equation (1);
- ***Step 1.2:*** Assign each incoming link of sensor j a weight equal to $E_j(t)$;
- ***Step 1.3:*** Calculate the shortest path from j to H;

Step 2: **If** $\exists\ j \in \{1, ..., N\}$ such that $E_j(t) = 0$ **then** stop and set:

$$L(X) := t; \tag{3}$$

Else $t = t + 1$, go to step 1;

The same algorithm can be easily modified to consider different energy models in Step 1.1 and routing algorithms (e.g. geographical-based [12] routing algorithms) in Step 1.3.

2.3. DPAP Analysis

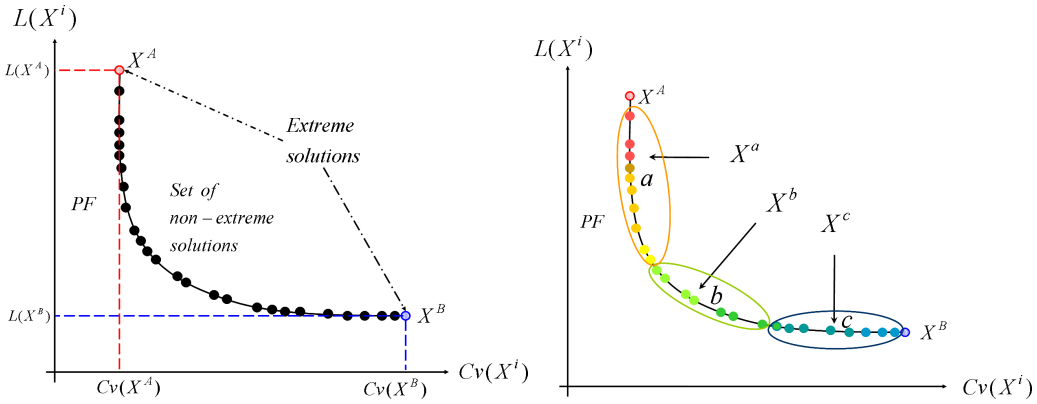
In the multiobjective DPAP defined here, there does not exist a solution that can optimize all objectives at the same time. Therefore, we will be interested in achieving a set of Pareto optimal solutions, or an approximation to it. The Pareto optimal solutions, however, which are close in the objective space, should have many similarities with each other in the search space, recalling the so-called Proximate Optimality Principle (POP) [38]. The POP, an underlying assumption in most heuristics, assumes that good solutions have similar structure.

This subsection aims at providing some insights about the properties and features of some particular solutions that might be part of the PF. For example, the *extreme solutions* X^A and X^B (Figure 1(a)), which optimize one objective each, are identified and good solutions[1] are designed analytically. Moreover, ad hoc design guidelines are provided for the remaining subset $PF - \{X^A, X^B\}$, named the *non-extreme set of solutions*, based on some network concepts and their positions (e.g. areas a,b and c in Figure 1(b)) in the objective space (i.e. objective preference).

The extreme solution X^A provides the maximum lifetime and minimum coverage among all the solutions in PF,

$$L(X^A) = \frac{E}{d_{min}^{\alpha} \times amp}, \quad Cv(X^A) = A'/(x \times y),$$

[1]Note that good solutions do not imply optimal solutions.



(a) The PF of DPAP (b) Various Pareto optimal solutions of the PF

Figure 1: Classifying the optimal network designs in DPAP

where $A' \approx (2 \times (R_s + d_{min}))^2$, d_{min} is a controllable parameter that indicates the minimum distance allowed between a sensor and H. Hence, a dense deployment of all sensors around H with minimum transmission distances, $R_c^i = d_{min}$, and direct communication with H (hence $r_i(t) = 0$) is desirable.

The extreme solution X^B provides the maximum coverage and minimum lifetime among all the solutions in PF. $Cv(X^B)$ highly depends on N, which in DPAP is assumed to be small. Thus, let $N \leq \frac{(x \times y)}{(2R_s)^2}$. In that case, X^B is designed by regularly deploying N sensors with a fixed distance $2R_s$ between each other and H, avoiding any sensing range overlaps:

$$Cv(X^B) = \frac{N \times \pi R_s^2}{(x \times y)}, \quad L(X^B) = \frac{E}{k \times (N/4) \times (2R_s)^\alpha \times amp},$$

where $(k \times (N/4) \times (2R_s)^\alpha \times amp)$ is the energy consumption of each sensor i that is directly connected to H at each t, and $N/4 \times k$ is a fixed minimum number of packets of size k (i.e. the traffic load) that should be carried by each sensor i, assuming a regular, symmetrical deployment.

The goal of DPAP, however, is to provide the interested users with a diverse set of network design choices, giving the trade-off between the extreme solutions X^A and X^B. However, the procedure of designing the non-extreme topologies is complicated, since there is not a scalar method which can design all of them, in a single run. In the following, we introduce some general concepts for searching and/or designing good solutions in different areas of the objective space (e.g. a,b, and c in Figure 1(b)):

- Solution X^a: favors a high network lifetime. Hence, the focus is to provide dense network designs by placing the sensors close to H, with low transmit power levels. This, however, leads to high sensing range overlaps and poor coverage.

- Solution X^c: favors a high network coverage. Therefore, the focus is to provide spread network designs, by placing the sensors with high transmit power levels and low sensing range overlaps between each other [39] and the area boundaries. This, however, leads to a high energy consumption, which results to a poor lifetime.

Furthermore, it is expected that the interrelation between the solutions X^a and X^c and the aforementioned network concepts "fades" as they get closer to the center of the PF, respectively. Thereinafter, a combination of those concepts could provide a balance on the DPAP's objectives as follows:

- Solution X^b: the sensors are connected in such a way that their transmit power level decreases/increases, and the sensing range overlaps increase/decrease as they get closer to H, according to a slight preference on the lifetime or coverage objective, respectively.

2.4. DPAP solution representation and ordering

In this paper, a candidate solution X consists of N items. Its j-th item has two parts, (x_j, y_j) and P_j, which represent the location and the transmit power level of sensor j, respectively.

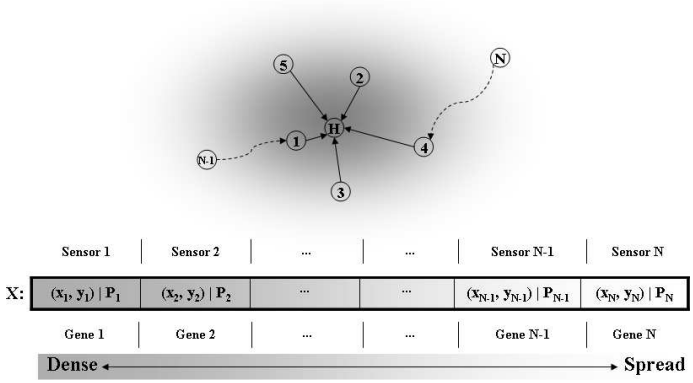


Figure 2: The dense-to-spread solution representation.

In our approach, the items of a solution X are ordered as follows: the sensor locations in X are sorted based on their distance to H, where 1 is the closest and N is the farthest sensor location with respect to H, respectively. This results in having the locations of the sensors that are densely deployed around H at the beginning of each solution and the locations of the sensors that are spread away at the end. The dense to spread ordering (denoted as dtsOr) facilitates the crossover operator that will be introduced shortly. Solution X is illustrated in Figure 2. Thereinafter, each sensor j is assigned a transmit power level P_j proportional to $R_c^j \leq R_{\text{max}}$, such that it reaches its closest neighbor sensor, e.g. k, where $k < j$. The whole process is outlined in Algorithm 2.

- In Step 1, the sensors are ordered based on their distance from H, which facilitates the proposed evolutionary operators that will be introduced shortly.

- Step 2 assigns a minimum transmission power to each sensor such that it reaches its closest neighbor. This is due to the concept that multiple short hops are more beneficial than a long hop in applications where N is small and the sensors communicate through long transmission ranges [15]. The reason is that,

$$(r_j(t)+1) \times d_{ju}^{\alpha} > ((r_j(t)+1) \times d_{jk}^{\alpha} + (r_k(t)+1) \times d_{ku}^{\alpha}),$$

Algorithm 2 The dense-to-spread representation process for each solution Y

Input: A solution Y;

Output: A solution X;

Step 1: Calculate the dense-to-spread ordering of Y to get X;

Step 2: **for** each (x_j, y_j) in X **do**

$$P_j = \begin{cases} (d_{j,H})^{\alpha} & \text{if } (x_H, y_H) \text{ is } j\text{'s closest location and } d_{j,H} \leq R\text{max–(a)} \\ (d_{j,k})^{\alpha} & \text{if } (x_k, y_k) \text{ is } j\text{'s closest location, } k < j,\ P_k \neq 0 \text{ and } d_{jk} \leq R\text{max–(b)} \\ 0 & otherwise \end{cases}$$

since $d_{ju} \geq d_{jk} + d_{ku}$.

- If Step 2 (a) and (b) are not satisfied then, $P_j = 0$, which means that sensor j is disconnected.

3. The proposed MOEA/D for the DPAP

This section details each operator of the proposed MOEA/D designed for the DPAP. Note that, the underlying idea behind the problem-specific EA operators might shed some light on the design of MOEA/Ds for other MOPs.

3.1. Decomposition

Initially, MOEA/D needs to decompose a MOP into a set of subproblems. Any decompositional technique can serve for this purpose [35]. In this paper, the Weighted Sum approach is used, as follows. The multiobjective DPAP is decomposed into m scalar optimization subproblems considering two objectives. The i-th scalar optimization subproblem can be defined as:

$$max \quad g^i(X, \lambda^i) = \lambda^i L(X) + (1 - \lambda^i) Cv(X),$$

where λ^i is the weight coefficient of subproblem $i = 1, \ldots, m$. For the remainder of this paper, we consider a uniform spread of the weights λ^i, which remain fixed for each i for the whole evolution and are determined as follows:

$$\lambda^i = 1 - (i/m)$$

for $i = 1, \ldots, m$ and $\lambda^1 = 1$. Hence, the λ^i coefficient is mainly utilized for decomposing a MOP into a set of scalar subproblems by adding different weights to the objectives. In this paper, we have also given a problem-specific meaning to this parameter. Considering the λ^i weight coefficient of a subproblem i, we can predict the objective preference of a particular design and therefore, its position in the objective space, e.g. Figure 1(b). Thereinafter, appropriate scalar strategies can be employed to optimize it accordingly. Note that, this beneficial procedure cannot be utilized by any non-decompositional MOEA framework.

3.2. MOEA/D general framework

A general MOEA/D approach usually proceeds as in Algorithm 3.

Algorithm 3 The MOEA/D general framework

Input: • network parameters (A, N, E, R_s, $P_{\max}$);
- m : population size and number of subproblems;
- T: neighborhood size;
- uniform spread of weight vectors $(\lambda^1, 1-\lambda^1), ..., (\lambda^m, 1-\lambda^m)$;
- the maximum number of generations, gen_{max};

Output: the external population, EP.

Step 0-Setup: Set $EP := \emptyset$; $gen := 0$; $IP_{gen} := \emptyset$;

Step 1-Initialization: Uniformly randomly generate an initial internal population $IP_0 = \{X^1, \cdots, X^m\}$;

Step 2: For $i = 1, \ldots m$ **do**

Step 2.1-Genetic Operators: Generate a new solution Y using the genetic operators.

Step 2.2-Repair heuristic: Apply a problem-specific repair heuristic on Y to produce Z.

Step 2.3-Update Populations: Use Z to update IP_{gen}, EP and the T closest neighbor solutions of Z.

Step 3-Stopping criterion: If stopping criterion is satisfied, i.e. $gen = gen_{max}$, **then** stop and output EP, **otherwise** $gen = gen + 1$, go to Step 2.

The following remarks are related to the MOEA/D framework :

- The internal population IP_{gen} of size m keeps the best solution found so far for each subproblem.
- Solution Y is generated by using a selection operator (which will be detailed in Subsection 3.4.1) to choose two parent solutions from the IP_{gen}, e.g. Pr_1, Pr_2, a crossover operator (which will be detailed in Subsection 3.4.2) to produce a new solution from Pr_1, Pr_2 and a mutation operator (which will be detailed in Subsection 3.4.3) to modify the new solution Y. Solution Z is produced by using a repair method on Y.
- The T closest neighbor solutions of Z are the solutions of the T closest subproblems of i in terms of their weights $\{\lambda^1, \cdots, \lambda^m\}$. This is commonly known as the T neighborhood of subproblem i.
- The external population EP stores all the non-dominated solutions found so far during the search.

In the following the MOEA/D-based, DPAP-specific operators are presented.

3.3. Initialization

In Step 1 of MOEA/D, we adopt a random method to generate m solutions for the initial internal population (i.e IP_0). Namely, a solution Y is initiated by uniformly randomly generating N sensor locations $(x_j, y_j) \in A$. Solution Y is then ordered to X using Algorithm 2. Each X is then added in IP_0.

3.4. Genetic Operators

In the i-th pass of the loop in Step 2 of the MOEA/D, the genetic operators generate a new solution in Step 2.1.

3.4.1. Selection operator

The first genetic operator in Step 2.1 is the selection. Selection is responsible for emulating the survival of the fittest concept and to choose promising solutions from the current population, known as parents, to be included for offspring reproduction in the next generation. In this paper, we propose a M-tournament selection operator (denoted as M-tourS) that combines the mating restriction considered during selection in [35] and a standard tournament selection [25]. M-tourS, which proceeds as in Algorithm 4, mainly relies on one of the core ideas of MOEA/D. That is, two neighbor solutions in the weight space (i.e. with respect to the Euclidean distance of their weights $\{\lambda^1, ..., \lambda^m\}$) should be similar to each other in the decision space (please refer to [35] for more details).

Algorithm 4 The M-tournament selection operator (M-tourS) for each subproblem i

Input : A population of solutions, IP_{gen};

Output: Two parent chromosomes, Pr_1, Pr_2;

Step 1: Select the solutions $X \in IP_{gen}$ of the M closest subproblems of i to compete in the tournament;

Step 2: Evaluate each solution X of the tournament in terms of $g^i(X, \lambda^i)$;

Step 3: Find the best two solutions of the tournament, set them as Pr_1, Pr_2 and stop;

Hereupon, the main differences of the proposed operator compared to the standard tournament selection operator [25] (denoted as tourS) and the selection operator suggested by [35] (denoted as T-randomS) are the following:

- In Step 1, the solutions selected to compete in a subproblem i's tournament are the solutions of the M closest subproblems of i in IP_{gen}, in terms of the Euclidean distance of their weights $\{\lambda^1, \cdots, \lambda^m\}$, which are called X^i's neighbors, instead of:
  - randomly selecting M solutions from IP_{gen} to compete in the tournament as in [25], having a higher probability of providing poor offspring for a particular subproblem i.
  - randomly selecting solutions from the T closest subproblems of i, as in [35], without competing in a tournament.
- In Steps 2 and 3, X^i's neighbors, e.g. X^j and X^k, are competing in i's tournament in terms of $g^i(X, \lambda^i)$, ignoring their own λ^j and λ^k, their Pareto domination and/or ranking. In this way, more selection pressure is provided towards the optimal point of each particular i for better exploitation.

  Remark 1: the optimization of a network design X^i, should mainly acquire good topological information (i.e. efficient sensor locations and transmit power levels) from a neighbor network design X^j; instead of a network design X^m which is far away in the weight space (even if X^m is a non-dominated solution). This is due to the highly non-linear multi-hop nature of WSNs. A tiny change in the topology may lead to a big change on the objective values, because of the connectivity and the exponential relationship between the sensors transmission distance and energy consumption. According to Subsection 2.3, the subproblems of area a prefer dense network

designs comprised of sensors located close to H with low transmit power levels. In contrast, subproblems in area c prefer spread network designs comprised of sensors spread along the sensing field with high transmit power levels. Thus, it should be preferable to increase the selection pressure, initiate tournaments composed of neighbor solutions and select the best for mating, decreasing the probability of generating poor offspring.

Remark 2: The persistent selection of the best solutions in the neighborhood for parenthood might also have some undesirable effects such as premature convergence, i.e. force the evolutionary search to get trapped in local optima and have a negative impact on the diversity of the population. These cases are usually alleviated by the mutation operator (which will be detailed in Subsection 3.4.3).

The two selected parent solutions Pr_1 and Pr_2 are then forwarded for recombination to the crossover operator.

3.4.2. Crossover operator

In Step 2.1 (Algorithm 3), the crossover combines the two parents Pr_1 and Pr_2 to generate a new solution-the offspring denoted as O, with a probability rate r_c. In this paper, we propose an adaptive crossover operator (denoted as aX) that probabilistically controls two crossover strategies, each favoring different areas of the objective space.

Initially, the window crossover is designed in which the control parameters (behaviors) change dynamically from subproblem to subproblem based on instant requirements. To do so, it determines a "window" of size:

$$w^i := N + N \times (1 - \lambda_i), \tag{4}$$

to select promising genetic material from each parent and direct the search into promising areas of the search space for each particular i. The window crossover strategy proceeds as in Algorithm 5,

Algorithm 5 Window crossover for a subproblem i

Input: Two solutions Pr_1 and Pr_2;

Output: A solution O;

Step 0: Set $O = \emptyset$; $U = Pr_1 \cup Pr_2$;

Step 1: Order solution U by using Algorithm 2;

Step 2: Uniformly randomly generate an integer j from $\{1, 2, \ldots [w^i]\}$, where w^i is defined as in Equation 4;

Step 3 **If** there exists a (x_j, y_j) in $U = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{2N}, y_{2N})\}$ **then**

Step 3.1: Delete (x_j, y_j) from U and add it in O;

Step 3.2: **If** the size of O is not N **then** goto Step 2;

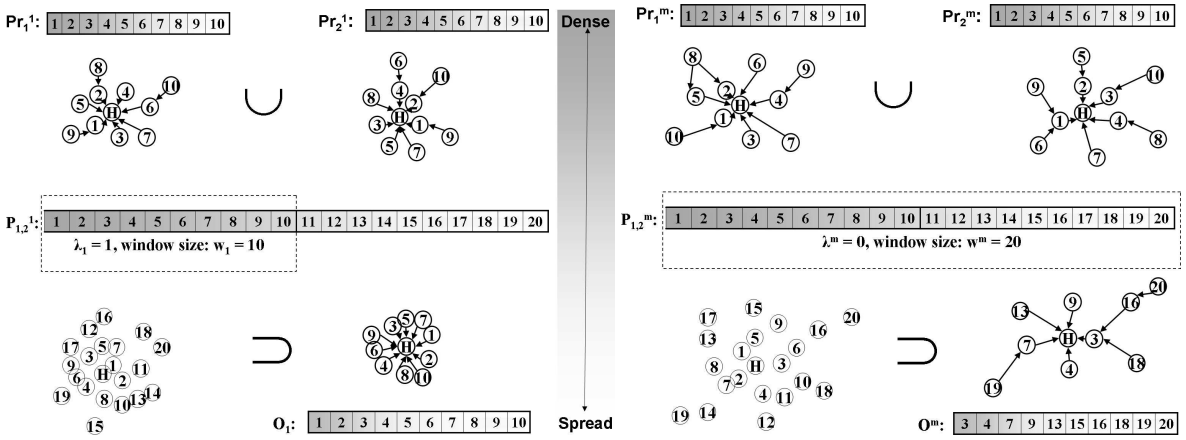
otherwise stop and output O;

- The merged solution U is of size $2N$.
- When λ^i is large and $L(X)$ favors $Cv(X)$, the window is small such that the sensor locations that will be added in O are as close to H as possible with low transmit power levels to provide higher network lifetime.

- When λ^i decreases and $Cv(X)$ starts favoring $L(X)$, w^i gradually increases to give the chance to the sensor locations which are spread in A to be added in O and therefore to provide better network coverage.

- Note that, the window always start at position 1 of solution X to always include the sensor locations of the "dense" part of the network (i.e. close to H, see Figure 2) and therefore to maintain the connectivity as the sensor locations spread in the topology.

Figure 3(a) shows a crossover operation for the extreme subproblem 1 with $\lambda^1 = 1$ and a minimum $w^1 = N$. The sensor locations which are closer to H from both Pr_1 and Pr_2 are added in offspring O, giving a new dense network design. Figure 3(b) shows a crossover operation for the other extreme subproblem m with $\lambda^m = 1$ and maximum $w^m = 2 \times N$. Sensor locations are randomly selected from both Pr_1 and Pr_2 and added in offspring O giving a new spread network design. Thus, the window strategy should be capable of providing offspring solutions in all areas of the objective space in Figure 1(b).



(a) Window crossover for subproblem 1

(b) Window crossover for subproblem m

Figure 3: Examples on the problem-specific window crossover operator

Remark 1: The window crossover on its own, however, may have some undesirable effects for low weights (e.g λ^m) and particularly for area c of the objective space, generating poor offspring. More specifically, when $\lambda^i \rightarrow 0$ then $w^i \rightarrow 2 \times N$, which basically drives the crossover operation into a uniform random selection of sensor locations from the merged parent set U. In that case, there is a high probability of selecting locations which are too close to each other, resulting in high sensing range overlaps and low network coverage. This is not beneficial for the particular subproblems and consequently for offspring reproduction of network designs that require high coverage quality.

To overcome this undesirable effect, a clustering crossover is designed, which aims at obtaining network topologies of high coverage. The clustering crossover proceeds as in Algorithm 6.

Algorithm 6 Clustering crossover strategy for a subproblem i

Input: Two solutions Pr_1 and Pr_2;

Output: A solution O;

Step 0: Set $O = Pr_1 \cup Pr_2$; $d' = d_c$;

Step 1: Order infeasible solution $O = \{(x_1, y_1), \ldots, (x_{2N}, y_{2N})\}$ by using Algorithm 2;

Step 2: For $j = 1$ to $2N$

While $(x_j, y_j) \in O$ and $\exists (x_k, y_k) \in O | d_{jk} \leq d'$ **do**;

Step 2.1: Uniformly randomly delete either location (x_j, y_j) or (x_k, y_k) from O;

Step 2.2: **If** the size of O is equal to N **then** stop and output O;

End while

Step 3: Set $d' = d' + d_c$ and goto Step 2;

- Initially, solution O is of size $2N$. This solution is infeasible since N is the maximum number of locations allowed in each solution. Each sensor j at location $(x_j, y_j) \in O$ represents a cluster, having d' as the minimum Euclidean distance measure between each cluster, which is initially set as the distance d_c between the centers of two adjacent diagonal grids in area A.

- In Step 2, two clusters centered at locations (x_j, y_j) and (x_k, y_k) are merged if $d_{jk} \leq d'$. In that case, either location (x_j, y_j) or (x_k, y_k) is deleted from O. This continues until N locations remain in O.

- When Step 3 is reached, solution O is still infeasible and there are no more locations with $d_{jk} \leq d'$. Thus, increase the Euclidean distance measure $d' = d' + d_c$ to further spread the locations in the solution.

Remark 2: This approach benefits the coverage objective and particularly the solutions of the subproblems with low weights in area c, having less probability to create a poor offspring than the window crossover.

Remark 3: In contrast, it might provide a poor lifetime objective, since the sensors should be assigned high transmit power levels to support the spread-like deployment directed by the clustering-based crossover.

Algorithm 7 Adaptive crossover operator for each subproblem i

Input: Two solutions Pr_1 and Pr_2;

Output: A solution O;

Step 1:

$$Set\ \delta = \begin{cases} 1 & if\ \lambda^i \geq 0.5 \\ \lambda^i + 0.1 & if\ 0.3 < \lambda^i < 0.5 \\ 0 & otherwise \end{cases} \tag{5}$$

Step 2: Uniformly randomly generate a number $rand$ from [0,1].

Step 3:

$$Apply \begin{cases} \text{Algorithm 5-window } to\ generate \text{ O} & if\ rand < \delta \\ \text{Algorithm 6-clustering } to\ generate \text{ O} & otherwise \end{cases} \tag{6}$$

The combination of the window and the clustering crossovers can be a promising example of a probabilistic mixture or an adaptive crossover operator for DPAP, as outlined in Algorithm 7. In this kind of crossovers, different mechanisms are adopted with a probability δ for producing a new solution, where $\delta = \lambda^i$ means that two crossover strategies are almost equally applied in each generation. In this paper, we suggest a δ probability such that the window crossover is applied with highest probability in areas a and b and the clustering in area c (considering Remarks 1-3).

3.4.3. Mutation operator

The last operator in Step 2.1 of MOEA/D (Algorithm 3 in Subsection 3.2) is the mutation, which is responsible for maintaining the diversity of the population by modifying the locations of a solution O with a r_m probability. However, the choice of the new location should be carefully determined, since an improper choice may damage all the preceding actions of the problem-specific selection and crossover operators. For example, if the mutation operator modifies a sensor location (x_j, y_j) without considering the subproblem's objective preference might result in:

- a disconnected sensor, which is undesirable for subproblems that favor a high coverage.
- sensors with high P_j, which is undesirable for subproblems that favor a high lifetime.
- partition of the network, since the deletion of a sensor in multi-hop communication may disconnect other parts of the network and might create a bottleneck that negatively influences the network lifetime and/or uncover any previously covered region.

Thus, it is considered reasonable to allow the mutation operator to randomly modify the locations of a solution with an r_m probability, but restricting the modification to close to the current value or at least to bias the probability distribution in its favor. This may maintain the diversity of the population without destructive behavior or unnecessary searches. Thus, we propose an adaptive mutation operator that is composed of two problem-specific mutation strategies, namely the local and global mutations that favor different areas of the objective space, respectively. The adaptive mutation operator (aM) proceeds as in Algorithm 8.

- In Step 2, if λ^i favors the lifetime objective (i.e. area a and the beginning of area b) then a location (x_j, y_j) is modified "locally", i.e.

$$\begin{aligned} &\text{Uniformly randomly generate } x'_j \in [x_j - d_c, x_j + d_c] \\ &\text{and } y'_j \in [y_j - d_c, y_j + d_c], \end{aligned} \tag{7}$$

  to provide a minimum shift from its current position, where d_c is the distance between the centers of two adjacent diagonal grids, seeking to:

  – either, slightly increase $Cv(X)$ in the sake of increasing P_j, when the shift is backward with respect to H,

  – or, benefit the lifetime objective by decreasing the sensors P_j.

Algorithm 8 Adaptive mutation operator for each subproblem i

Input: A solution O.

Output: A mutated solution Y.

Step 0: Set r_m;

Step 1: Order solution O by using Algorithm 2;

If $\lambda^i > 0.5$ **then**

Step 2: **For** $j = 1$ to N **do**

Step 2.1: Generate a uniform random number $rand \in [0, 1]$;

Step 2.2: **If** $rand \leq r_m$ **then**

Calculate (x'_j, y'_j) using Equation (7). Replace $(x_j, y_j) \in O$ with (x'_j, y'_j);

Else

Step 3: **For** $j = 1$ to N **do**

Step 3.1: Generate a uniform random number $rand \in [0, 1]$;

Step 3.2: **If** $rand \leq r_m$ **then**

Calculate A' and (x'_j, y'_j) using Equation (8). Replace $(x_j, y_j) \in O$ with (x'_j, y'_j);

End if

Step 4: Output $Y = O$;

- In Step 3, if λ^i favors the coverage objective (i.e. the end of area b and area c) then a location (x_j, y_j) is modified "globally", i.e. a new location (x'_j, y'_j) is generated in a sub-area $A' \subseteq A$ which is defined as follows:

$$\begin{aligned} &x_{min} = (x_H - |x_H - x_j|) - R\text{max}, \quad y_{min} = (y_H - |y_H - y_j|) - R\text{max}; \\ &x_{max} = (x_H + |x_H - x_j|) + R\text{max}, \quad y_{max} = (y_H + |y_H - y_j|) + R\text{max}; \\ &x' = x_{max} - x_{min}, \quad y' = x_{max} - x_{min}; \\ &A' \subseteq A \text{ is a 2-D area with length } x' \text{ and width } y'. \\ &\text{Uniformly randomly generate } (x'_j, y'_j) \in A'; \end{aligned} \tag{8}$$

  where x' and y' are the width and height of A', respectively. Note that when $\lambda \to 0$ then it should be that $A' \to A$.

The modified offspring is then forwarded to the repair operator.

3.5. Repair Operator

In Step 2.2 of MOEA/D (Algorithm 3), a local heuristic checks a solution Y if:

Case #1: there is a location $(x_j, y_j) \in Y$ at the same location as H (i.e. (x_H, y_H));

Case #2: a location $(x_j, y_j) \in Y$ is the same as another location $(x_k, y_k) \in Y$;

In both cases, the local heuristic repairs the solution Y by uniformly randomly generating a new location $(x'_j, y'_j) \in A$, such that (x'_j, y'_j) does not fall in either Case #1 or Case #2. The repair heuristic increases the sensors' individual

utilization. Since, in both Cases #1 or #2 the sensors cannot benefit either the lifetime objective by acting, for example, as relays to increase the load balancing and/or increase the multiple short hops towards H, or the coverage objective by covering any uncovered regions in the topology. Solution Z is then used to update the populations of MOEA/D.

3.6. Update of populations

In Step 2.3, the populations (defined in Subsection 3.2) of MOEA/D are updated for each solution Z^i as follows:

1 The (IP_{gen}) update phase. If $g^i(Z^i|\lambda^i) > g^i(X^i|\lambda^i)$ then $IP_{gen} \cup \{Z^i\}$ and $IP_{gen}/\{X^i\}$, otherwise X^i remains in IP_{gen}.

2 The neighborhood (defined in Subsection 3.2) update phase. The new solution Z^i is compared with its T closest $X^j \in IP_{gen}$ neighbor solutions. If $g^j(Z^i|\lambda^j) > g^j(X^j|\lambda^j)$ then, $IP_{gen} \cup \{Z^i\}$ and $IP_{gen}/\{X^j\}$, otherwise, X^j remains in IP_{gen}, where $j = 1, \ldots, T$.

3 The (EP) update phase. $EP = EP \cup \{Z^i\}$ if Z^i is not dominated by any solution $X^j \in EP$, and $EP = EP/\{X^j\}$ if $Z^i \succ X^j$, for all $X^j \in EP$.

3.7. Termination criterion

At the end of each generation the termination criterion (the maximum number of generations, gen_{max}) is checked to decide whether the search should stop.

4. Experimental setup

In this paper, we study four network test instances (Table 1), which represent a broad class of the large-scale and spread DPAP WSN topologies. The network test instances are designed following our analysis in Subsection 2.3.

Table 1: Network Instances

Network Instances	Area size, A (m^2)	# of sensors, N	Density (N/A)
NIn1	1×10^6	13	0.13×10^{-4}
NIn2	4×10^6	52	0.13×10^{-4}
NIn3	1×10^6	50	0.5×10^{-4}
NIn4	4×10^6	200	0.5×10^{-4}

In DPAP, there are too many possible parameter settings to try them all. Hence, in our studies we have adopted the widely used Factorial design [40]. Factorial design investigates all possible combinations of the levels of some factors in a complete replication of an experiment. Factors k are the parameters that affect the experiment and levels (e.g. 2, a High and a Low level) are the factors' values. In cases where the experimenter can reasonably assume that certain interactions between the factors are negligible then, information on the main effects may be obtained by running only a fraction of the complete factorial experiment. This is known as the 2-level Fractional Factorial Design, denoted as $2^{k-\rho}$, where ρ are the factors which are not considered as a main effect on the experiment and their value is decided based on the interactions of the remaining $k-\rho$ factors. All algorithm factors and their levels are presented in Table 2.

Table 2: Parameter settings: algorithm factors with low\high levels

Algorithm Factors	Low	High
Crossover rate, r_c (subsection 3.4.2)	0.1	1
Mutation rate, r_m (subsection 3.4.3)	0.1	0.5
Max # of generations, $gen_{\max}$ (subsection 3.7)	100	250
Pop. size & # of subproblems, m (subsection 3.2)	120	200
Tournament size, M (subsection 3.4.1)	5	10
Neighborhood size, T (subsection 3.2)	2	10

In all simulation studies, the following network parameters are set [41], [42]: $R_s/R_{\max} = 100/200$, $E = 5J$, $d_{min} = 100m$, $a = 2$, $amp = 100pJ/bit/m^2$ and square-grids of side length $10m$. Moreover, the network lifetime and the network coverage are evaluated as in Subsection 2.1 and the lifetime objective is normalized by the $L(X^A)$ defined in Subsection 2.3. All algorithms were coded in Java programming language and run on an Intel/circledR Pentium 4 3.2 GHz Windows XP server with 1.5 GB RAM.

5. Performance Metrics

This section briefly describes the performance metrics used for comparing sets of solutions. In MOO, practitioners are usually interested in the quality of the approximation to the Pareto set that an algorithm is able to generate. In addition, a fast and efficient approach is also desirable. A single metric, however, cannot provide adequate results for the strength of an MOEA in all tasks. Therefore, a set of performance metrics are adopted as follows.

The Δ-metric, proposed by Deb et al. [29], measures the extent of spread achieved among the obtained solutions, as follows:

$$\Delta = \frac{d_f + d_l + \sum_{j=i}^{K} |d_j - \overline{d}|}{d_f + d_l + (K-1)\overline{d}},$$

where d_f and d_l are the distances between the extreme solutions X^1 and X^m and the optimal solution X^A and X^B, respectively. A low $\Delta(A)$ implies a uniform spread of the non-dominated network designs in the objective space by algorithm A, giving a variety of network design choices to the WSN decision maker.

A straightforward comparison metric between two sets of non-dominated solutions is the C-metric [29]. The $C(A, B)$ metric, which is usually considered as a MOEA's quality metric, evaluates the ratio of the non-dominated solutions in an algorithm A's Pareto Front dominated by the non-dominated solutions in an algorithm B's PF, divided by the total number of nondominated solutions obtained by algorithm A, i.e. *NDS(A)*. Hence, let EP^A be the external population of an algorithm A and EP^B be the external population of an algorithm B. Then,

$$C(A, B) = \frac{|EP^A - \{\in EP^A | \exists y \in EP^B : y \succ x\}|}{NDS(A)}.$$

The smaller the $C(A, B)$ is, the better A is. Note that $C(A, B) \neq C(B, A)$.

A common metric, usually considered in cases of real-life discrete optimization problems [30],[34], such as DPAP,

is the number of Non-Dominated Solutions (NDS) obtained by an algorithm A, i.e.

$$NDS(A) = |EP^A|.$$

In DPAP, it is very difficult to obtain many different NDSs. Hence, the higher the NDS(A) is, the better A is, in order to provide an adequate number of choices. However, NDS should be considered in combination with other metrics, (e.g. Δ and C metrics), since it is usually desirable to have a high number of NDS, when the set of solutions is of high quality and spread in the objective space. In contrast, and usually in cases of continuous optimization [35], when the number of NDS is too high, the decision making procedure becomes more complicated and more time consuming.

Besides, an efficient algorithm should obtain high quality solutions within an acceptable CPU time. Thus, the combination of NDS with the C and Δ metrics and the CPU time should be an adequate set of metrics to judge the effectiveness and efficiency of the concerned algorithms. In the following experimental study, statistical tests are carried out to check the significant difference between the average results obtained by each algorithm for each performance metric with a 95% confidence. A one-way ANOVA test is carried out when a group of algorithms is compared and the two-sample t-test when two algorithms are compared. Each test returns an h on the null hypothesis that the average results are not significantly different against the alternative that the average results are significantly different. The $h = +$ indicates a rejection on the null hypothesis and $h = -$ indicates a failure to reject the null hypothesis. Note that, each algorithm is executed around 20 times in each study.

6. Experimental results and discussion

The goals of this section are: 1) to study the effect of the proposed problem-specific evolutionary operators on the MOEA/D, with respect to several widely used operators, under various parameter settings. 2) To test the strength of the problem-specific MOEA/D against the NSGA-II in several network instances.

6.1. The effect of the evolutionary operators

Table 3: Evolutionary components combinations

Algorithm	Representation/Ordering	Selection	Crossover	Mutation
Alg1	rOr	tourS	1X	rM
Alg2	xyOr	tourS	1X	rM
Alg3	rOr	tourS	2X	rM
Alg4	xyOr	tourS	2X	rM
Alg5	dtsOr	tourS	aX	rM
Alg6	xyOr	M-tourS	2X	rM
Alg7	dtsOr	T-randomS	aX	rM
Alg8	dtsOr	M-tourS	aX	rM
Alg9	dtsOr	M-tourS	aX	aM

In this subsection, we study the effect of the proposed evolutionary operators (i.e. dtsOr, M-tourS, aX and aM) and evaluate their impact on MOEA/D. To do so, the following standard operators were used for comparison purposes:

Ordering: Two techniques were designed and compared with the proposed dense-to-spread ordering (dtsOr) defined in Subsection 2.4:

(I) Random ordering (rOr): the solution remains in a random structure during the search.

(II) x-y axis ordering (xyOr): each solution is ordered in an increasing order of the locations x-coordinates. In cases where the x-coordinates are the same then the y-coordinates are considered.

Selection: The standard tournament selection [25] (denoted as tourS) and the T-random selection (denoted as T-randomS) proposed by [35] were compared with the proposed M-tournament selection operator (M-tourS) defined in Subsubsection 3.4.1:

(I) Standard tournament selection (tourS): initiate a tournament by uniformly randomly selecting M solutions from the population. The two best solutions (in terms of Pareto dominance [17]), are selected for parenthood [25].

(II) The T-random selection (T-randomS) proposed by [35]: for each subproblem i, two solutions are randomly selected from its neighborhood of size T (the neighborhood is defined in Subsection 3.2).

Crossover Operators: Two standard crossover operators [25] were compared with the proposed adaptive crossover operator (aX) defined in Subsubsection 3.4.2:

(I) One-point crossover (1X): suppose two parent solutions (e.g. Pr_1, Pr_2) of size N. A crossover point is randomly selected from 1 to N-1. The pieces of the parents are exchanged to produce two offspring, e.g. O_1, O_2.

(II) Two-point crossover operator (2X): two crossover points are randomly selected from numbers 1 to N-1. The pieces of the parents are exchanged to produce two offspring, e.g. O_1, O_2. The two-point crossover was originally proposed for MOEA/D in [35].

Note that, the $1X$ and $2X$ usually produce two offspring in each recombination. In this paper, one offspring O is uniformly randomly chosen from $\{O_1, O_2\}$ to keep the number of function evaluations the same, for fairness.

Mutation Operators: A standard (random) mutation operator was compared with the proposed adaptive Mutation operator (aM), defined in Subsubsection 3.4.3:

(I) Random Mutation (rM): a location (x_i, y_i) is modified by uniformly randomly generating a new location $(x_i', y_i') \in A$. A standard (random) mutation is originally proposed for MOEA/D in [35].

This subsection involves nine representative MOEA/D versions, as summarized in Table 3. Each algorithm is composed of different evolutionary operators. The algorithms are studied in three tests in NIn1. In each test, a 2^{6-3} fractional factorial design of the parameter settings (Table 2) is adopted.

Test 1 - The effect of the adaptive crossover (aX) with the dense to spread ordering (dtsOr)

Test 1 studies the effect of the adaptive crossover (aX) defined in Subsubsection 3.4.2, with the dense to spread ordering (dtsOr) defined in Subsection 2.4. Thus, the crossover rate r_c, $gen_{\max}$ and m (where $gen_{\max} \times m$ is the total number of function evaluations performed by each algorithm in each run) are considered in the basic design as the main effects of test 1 (Table 4). Then, Algorithms 1 to 5 of Table 3 are compared on NIn1.

Table 4: Algorithm parameter settings of test 1, ParSetting1-8

Algorithms:	Alg1-5					
	Basic Design					
Settings	r_c	$gen_{\max}$	m	r_m	M	T
ParSetting1	0.1	100	120	0.5	10	10
ParSetting2	1	100	120	0.1	5	10
ParSetting3	0.1	250	120	0.1	10	2
ParSetting4	1	250	120	0.5	5	2
ParSetting5	0.1	100	200	0.5	5	2
ParSetting6	1	100	200	0.1	10	2
ParSetting7	0.1	250	200	0.1	5	10
ParSetting8	1	250	200	0.5	10	10

Table 5: The statistical results of test 1 for ParSetting1-8

Metric	Alg1	Alg2	Alg3	Alg4	Alg5	ANOVA
Δ:	0.9383	1.0131	**0.9069**	0.9208	0.9081	+
CPU time:	0.9100	1.4366	0.8787	1.4241	**0.7645**	-
#NDS:	9.5000	8.7500	9.2500	7.7500	**12.3750**	-

C-metric:	Alg(1,2)	Alg(2,1)	Alg(3,4)	Alg(4,3)	Alg(2,4)	Alg(4,2)	Alg(4,5)	Alg(5,4)
Average:	0.6749	*0.0235*	0.6158	*0.0000*	0.4750	*0.2556*	**0.2899**	0.4996
t-test:	+		+		+		+	

From the results of test 1, summarized in Table 5, the following conclusions are drawn:

- Alg3 obtains the best $\Delta = 0.9069$ performance with Alg5 being slightly worse with $\Delta = 0.9081$. There is a significant difference between the results in terms of Δ-metric. Alg5 is the fastest method, with respect to Alg1-4, since it requires 0.7645hrs in average, to obtain the highest number of $NDS = 12.3750$. Both the CPU time and NDS are not significantly different from the results obtained by the other MOEA/Ds.

- the comparison in terms of the C-metric shows that the xyOr encoding favors the 1X crossover in Alg2, which outperforms Alg1. In contrast, the rOr is more effective for the 2X crossover, since Alg4 outperforms Alg3. Thereinafter, the comparison between Alg2 and Alg4 shows the superiority of 2X crossover with rOr, which also outperforms Alg5. In all cases, the quality difference between the MOEA/Ds is significant.

The reason that Alg5 (composed of the proposed aX and dtsOr encoding) performs poorly in test 1 is mentioned in Subsubsection 3.4.1, Remark 1. That is, the generic tournament selection operator $tourS$, used in test 1, does not

provide adequate selection pressure (i.e. does not select the best parents in the population). This has a negative impact on the proposed aX with dtsOr, resulting in poor offspring reproduction. Therefore, in the next test, the proposed M-tourS operator is adopted to improve Alg5's performance.

Test 2 - The effect of the M-tournament selection operator (M-tourS):

Table 6: Algorithm parameter settings of test 2, ParSetting9-16

Algorithms:	Alg4-8					
	Basic Design					
Settings	M	genmax	m	r_m	r_c	T
ParSetting9	5	100	120	0.5	1	10
ParSetting10	10	100	120	0.1	0.1	10
ParSetting11	5	250	120	0.1	1	2
ParSetting12	10	250	120	0.5	0.1	2
ParSetting13	5	100	200	0.5	0.1	2
ParSetting14	10	100	200	0.1	1	2
ParSetting15	5	250	200	0.1	0.1	10
ParSetting16	10	250	200	0.5	1	10

Table 7: The statistical results of test 2 for ParSetting9-16

Metric	Alg4	Alg5	Alg6	Alg7	Alg8	ANOVA		
Δ:	0.9507	0.9209	0.9762	**0.9088**	0.9309	-		
CPUtime:	0.7244	0.4161	0.7700	**0.3705**	0.7542	-		
NDS:	9.1250	11.2500	12.8750	**15.8750**	13.5000	-		
C-metric:	Alg(4,6)	Alg(6,4)	Alg(5,7)	Alg(7,5)	Alg(4,7)	Alg(7,4)	Alg(4,8)	Alg(8,4)
Average:	*0.2125*	0.6274	0.6734	*0.2135*	*0.2798*	0.3543	0.3820	**0.3645**
t-test:	+		+		-		-	

In this test, we study the effect of the proposed selection operator (i.e. M-tourS), defined in Subsubsection 3.4.1. The experimental design of test 2 is presented in Table 6, in which M replaces r_c since M is a selection operator parameter. Alg7 extends Alg5 by replacing the standard selection $tourS$ with the selection operator proposed by [35], i.e. T-randomS. Alg8 extends Alg5 by replacing the $tourS$ with the proposed selection operator (M-tourS) to add network knowledge in this particular operator of MOEA/D and increase the selection pressure. Alg6 extends Alg4 (which is currently the algorithm with the highest quality on the PF according to test 1) by also replacing $tourS$ with the proposed M-tourS. Table 7 summarizes the results of test 2. The following conclusions are drawn:

- by increasing the selection pressure of Alg4, i.e. Alg6, the algorithm becomes slightly slower, obtaining lower diversity and significantly lower quality solutions in the PF. The solutions obtained by Alg4 dominate 62% of those obtained by Alg6. A slight increase is shown in the number of NDS.
- In contrast, Alg5 is outperformed by its extended version Alg7, giving a significant difference in quality. The performance of Alg7 increases with respect to Alg4 as well, in terms of Δ, number of NDS and CPU time, at

the cost of a lower quality of solutions in the PF (i.e. 35% of its solutions are dominated by Alg4). When the selection pressure is increased in Alg8, the MOEA/D becomes slower and slightly worse than its predecessor Alg5, in terms of diversity. However, it provides a higher number of NDS and the highest quality solutions obtained so far, with respect to Alg1-7.

Hence, the increase in the selection pressure provided by M-tourS, improves the performance of MOEA/D in terms of the NDS and the quality of solutions in the PF, at the cost of a slightly lower diversity (see section 3.4.1, Remark 2). To avoid the loss of diversity, the proposed adaptive mutation operator is further adopted in Test 3.

Test 3 - The effect of the adaptive mutation operator (aM):

Table 8: Algorithm parameter settings of test 3, ParSetting17-24

Algorithms:	Alg7-9					
	Basic Design					
Settings	r_m	genmax	m	M	r_c	T
ParSetting17	0.1	100	120	10	1	10
ParSetting18	0.5	100	120	5	0.1	10
ParSetting19	0.1	250	120	5	1	2
ParSetting20	0.5	250	120	10	0.1	2
ParSetting21	0.1	100	200	10	0.1	2
ParSetting22	0.5	100	200	5	1	2
ParSetting23	0.1	250	200	5	0.1	10
ParSetting24	0.5	250	200	10	1	10

Table 9: The statistical results of test 3 for ParSetting17-24

Metric	Alg7	Alg8	Alg9	ANOVA
Δ:	0.9135	0.9362	**0.9010**	-
CPU time:	**0.3179**	0.4975	0.4940	-
NDS:	**15.8750**	14.0000	14.6250	-

C metric:	Alg(7,9)	Alg(9,7)	Alg(8,9)	Alg(9,8)
Average:	0.5289	**0.2736**	0.7061	**0.2187**
t-test:	+		+	

Test 3 studies the effect of the proposed adaptive mutation operator (aM), defined in Subsubsection 3.4.3. The experimental design of test 3 is presented in Table 4, where the mutation rate r_m replaces M. In this test, Alg9 extends Alg8 by introducing the proposed aM to add network knowledge in this particular evolutionary component of MOEA/D and increase the diversity of the PF. The statistical results of test 3, summarized in Table 9, show the effectiveness of the proposed mutation operator, as follows:

- Alg9 performs better in terms of diversity and quality of solutions in the PF, with respect to both its predecessors Alg7 and Alg8. Besides, Alg9 is faster than Alg8 with a higher average number of NDS. However, only the difference in quality (i.e. C-metric) is significant.

In conclusion, the proposed evolutionary operators compose an efficient problem-specific MOEA/D, providing a large, diverse set of high quality network designs within an acceptable CPU time. The superiority of the proposed MOEA/D (i.e. Alg9) with respect to the two conventional MOEA/Ds (i.e. Alg1 and Alg3) is illustrated in Figure 4. Note that in all cases the lines between the points are just for better visualization and do not necessarily imply the presence of Pareto optimal solutions.

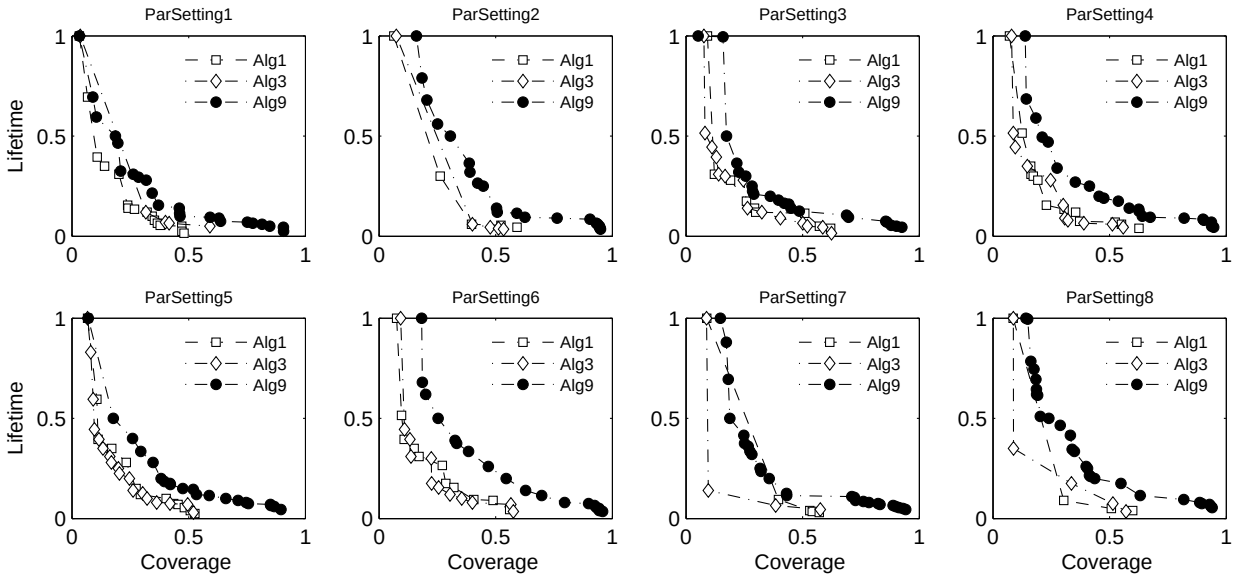


Figure 4: The DPAP-specific MOEA/D (i.e. Alg9) vs. general purpose MOEA/Ds (i.e. Alg1 and Alg3), NIn1

6.2. Further discussion on the adaptive crossover operator

In this subsection, we verify the effectiveness of the proposed adaptive crossover operator. This is empirically shown by comparing the MOEA/D with the two crossover strategies individually and probabilistically (i.e. adaptive crossover operator) for the parameter settings of Table 4 in NIn1. The results in Figure 5 clearly show the preference of each problem-specific crossover strategy on different subproblems. The window crossover is more flexible and generates non-dominated solutions across, almost, the whole range of the PF. However, the drawback mentioned in Subsubsection 3.4.2 (Remark 1) is clearly demonstrated in all cases. That is, the window crossover produces poor offspring when the λ parameter is low and consequently, when the subproblems desire a high coverage quality. In other words, the window crossover lacks obtaining high quality solution(s) in area c and to approximate solution X^B.

On the other hand, the clustering crossover is dedicated to providing solutions in the aforementioned areas of the PF, giving non-dominated solutions of higher coverage quality in almost all test instances, approximating the optimal solution X^B (Subsubsection 3.4.2, Remark 2). However, it lacks obtaining high quality solutions for the rest of the PF. This is due to the high transmission distances and consequently the high transmit power levels assigned to the sensors through the clustering crossover (Subsubsection 3.4.2, Remark 3). Thereinafter, the adaptability of the

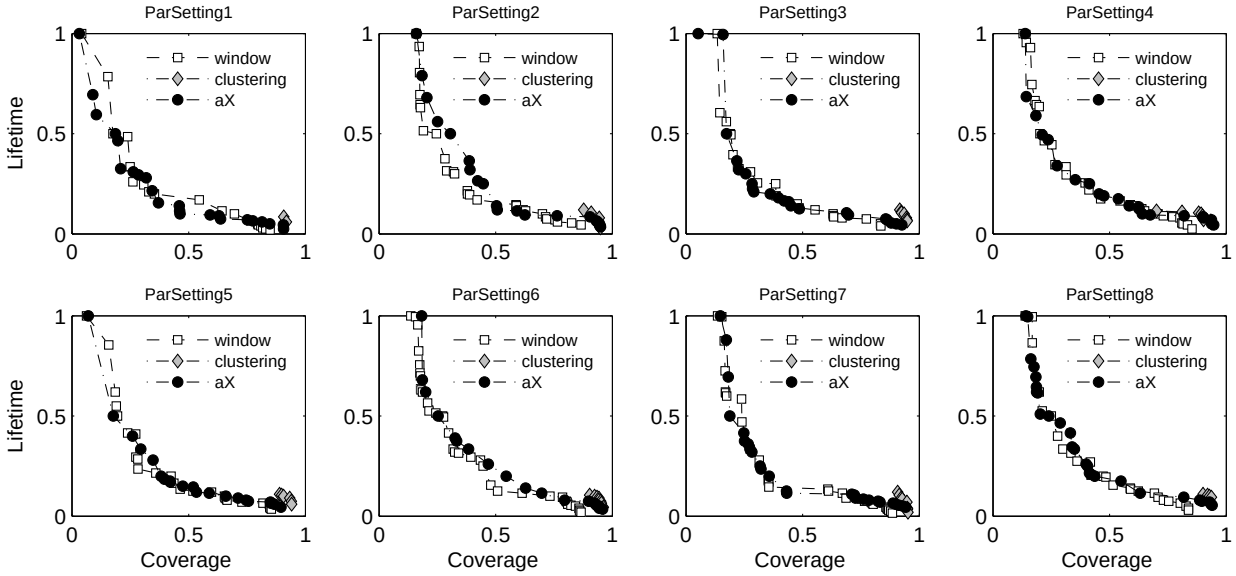


Figure 5: MOEA/D with the window, the clustering and the adaptive (aX) crossovers in ParSetting1-8, NIn1

proposed crossover operator is demonstrated. The aX takes advantage of both the window and the clustering crossover strategies and provides a diverse set of high quality solutions across the whole range of the objective space.

6.3. Comparison of MOEAs

In this subsection, we study the efficiency and effectiveness of the proposed problem-specific MOEA/D on DPAP. To do so, we have compared the proposed method with a state of the art in MOEAs based on Pareto dominance. Namely, the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [29]. NSGA-II maintains a population IP_{gen} of size m at each generation gen, for $gen_{\max}$ generations. NSGA-II adopts the evolutionary operators (i.e. selection, crossover and mutation) for offspring reproduction as MOEA/D. The key characteristic of NSGA-II is that it uses a fast non-dominated sorting and a crowded distance estimation for comparing the quality of different solutions during selection and to update the IP_{gen} and the EP. We refer interested readers to [29] for details. In this paper, NSGA-II adopts the following non-decompositional operators that have shown promising performance in Subsection 6.1: the x-y axis ordering (xyOr) (Ordering-II), the standard tournament selection (tourS) (Selection-I), the two-point crossover (2X) (Crossover-II) and the random mutation (rM) (Mutation-I). For comparing the two MOEAs we have adopted both visual and statistical comparison, through the performance metrics introduced in Section 5, in all network test instances of Table 1. The parameter settings were fixed, following the experience we gained from the latter experimental study: $gen_{\max} = 250$, $m = 120$, $r_c = 0.9$, $r_m = 0.5$ and $M = 10$. For MOEA/D, the number of subproblems is m and $T = 2$. Note that it is difficult to select optimal parameter values and a set of experiments cannot yield an insight that can be claimed in generality.

Figure 6 and Table 10 show the superiority of the proposed MOEA/D against the NSGA-II. MOEA/D performs

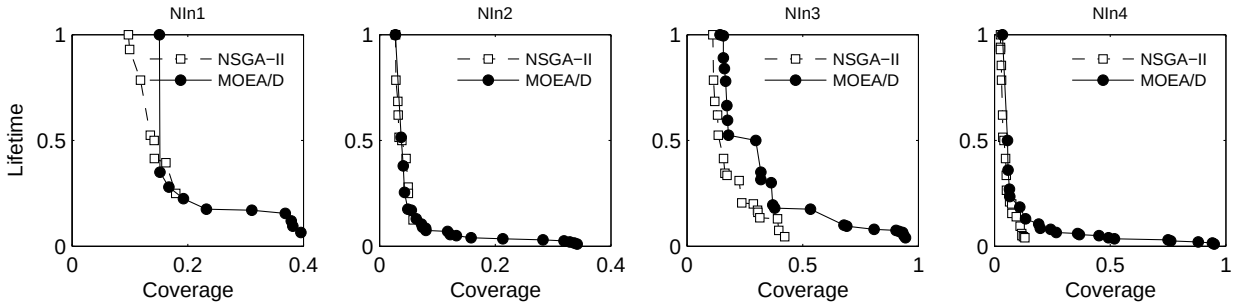


Figure 6: MOEA/D vs. NSGA-II on NIn1-4, NIn1

Table 10: MOEA/D (MD) vs. NSGA-II (NG), NIn1-4

Metric:	Δ(MD)	Δ(NG)	CPU(MD)	CPU(NG)	NDS(MD)	NDS(NG)	C(MD,NG)	C(NG,MD)
NIn1:	0.9842	**0.7417**	**0.2588**	2.7466	**10**	8	**0.1000**	0.7500
NIn2:	0.9780	**0.6535**	**2.2655**	2.7451	**21**	10	**0.1905**	0.2000
NIn3:	**0.6515**	0.6852	1.3784	**1.3300**	**23**	17	**0.0000**	1.0000
NIn4:	**0.7597**	0.8235	37.5989	**9.0934**	21	21	**0.0000**	0.8571
t-test:	-		-		-		+	

better than NSGA-II in terms of quality and number of NDS in all network instances and in terms of diversity in dense network topologies. In network topologies with low density, NSGA-II provides a more uniform spread of solutions. The PF obtained by MOEA/D, however, is much wider than the one obtained by NSGA-II in all cases, giving solutions in almost the whole area of the objective space. In contrast, NSGA-II lacks obtaining non-dominated network designs in area c and obtains few designs in area b.The difference between the two algorithms in terms of quality is significant. Note that, the CPU time required by the two approaches is not significantly different. Table 11 summarizes the lifetime and coverage of the extreme network designs X^A and X^B, which are analytically measured according to Subsection 2.3 for each network instance, and their approximation by the solutions X^1 and X^m obtained by each MOEA. The results show that MOEA/D approximates the extreme network designs more efficiently than NSGA-II. Another conclusion that can be empirically drawn is that, MOEA/D is not sensitive on the WSN's area size or density giving similar results in each case. That is, MOEA/D obtains a similar approximation towards the extreme solutions X^A and X^B in terms of lifetime and coverage quality, for the same $10000m^2$ and $40000m^2$ area sizes with different densities and for the same 0.0013 and 0.005 densities in different area sizes.

7. Conclusions and Future Research

In this paper, the DPAP in WSNs is formulated as a MOP and is decomposed into a set of scalar subproblems. The subproblems are classified based on their objective preferences and tackled by MOEA/D using problem-specific knowledge, simultaneously. A solution representation dedicated to DPAP and several DPAP-specific, MOEA/D-based evolutionary operators are proposed. Namely, the M-tournament selection, the adaptive crossover and the adaptive mutation operators, which are highly interrelated with each other and adapt to the needs and objective preferences of

Table 11: Analytical extreme solutions X^A and X^B and their approximation by the solutions X^1 and X^m obtained by MOEA/D and NSGA-II

NIn	Method	$L(X^A \backslash X^1)$	$Cv(X^A \backslash X^1)$	$L(X^B \backslash X^m)$	$Cv(X^B \backslash X^m)$
1	Analytical	1	0.16	0.00003	0.408
	MOEA/D	1	0.1511	0.065	0.3956
	NSGA-II	1	0.0974	0.25	0.1793
2	Analytical	1	0.04	7.69×10^{-5}	0.3926
	MOEA/D	1	0.027475	0.01	0.341525
	NSGA-II	1	0.0272	0.125	0.05
3	Analytical	1	0.16	8×10^{-5}	1
	MOEA/D	1	0.1431	0.04	0.944
	NSGA-II	1	0.1118	0.045	0.4221
4	Analytical	1	0.04	2×10^{-5}	1
	MOEA/D	1	0.0342	0.01	0.949575
	NSGA-II	1	0.0262	0.04	0.131

each subproblem dynamically, during the evolution. Simulation results have shown the effectiveness of the proposed EA operators on improving the performance of MOEA/D. The problem-specific MOEA/D has finally demonstrated its superiority against NSGA-II on several network test instances. MOEA/D obtains a diverse set of high quality WSN designs, without any prior knowledge on the objective preferences to facilitate the decision maker's choice.

There is a number of avenues for further research. For example, it will be interesting to test the performance of the proposed operators against more sophisticated EA operators, such as [43]. Moreover, the DPAPs in WSNs include many features (e.g. small, massively dense topologies) and issues (e.g. connectivity), which are also important as those in the proposed DPAP. Thus, various multiobjective DPAPs can be defined and tackled by problem-specific MOEA/Ds, similarly to this work. In principle, MOEA/D can easily adopt local search techniques. Hence, designing low-level problem-specific local heuristics, for further improving the performance of MOEA/D is also a future study.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Comms Magazine (2002) 102–114.

[2] X. Liu, P. Mohaparta, On the deployment of wireless data back-haul networks, IEEE Trans. on Wireless Comm. 6 (4) (2007) 1426–1435.

[3] M. Younis, K. Akkaya, Strategies and techniques for node placement in wireless sensor networks: A survey, Elsevier Ad Hoc Networks 6 (4) (2007) 621–655.

[4] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. B. Srivastava, Coverage problems in wireless ad-hoc sensor networks, in: IEEE Infocom, Vol. 3, 2001, pp. 1380–1387.

[5] K. Chakrabarty, S. S. Iyengar, H. Qi, E. Cho, Grid coverage for surveillance and target location in distribuuted sensor networks, IEEE Transactions On Computers 51 (12) (2002) 1448–1453.

[6] Q. Wu, N. S. V. Rao, X. Du, S. Sitharama, V. K. Vaishnavi, On efficient deployment of sensors on planar grid, Elsevier Computer Communications 30 (2007) 2721–2734.

[7] M. Cardei, J. Wu, Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems, CRC Press, 2005, Ch. Coverage in Wireless Sensor Networks, pp. 19–1–19–12.

[8] Q. Xue, A. Ganz, On the lifetime of large scale sensor networks, Elsevier Computer Communications 29 (2006) 502–510.

[9] P. Santi, Topology control in wireless ad hoc and sensor networks, ACM Computing Surveys 37 (2) (2005) 164–194.

[10] A. E. F. Clementi, P. Penna, R. Silvestri, Hardness results for the power range assignmet problem in packet radio networks, in: Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, Springer-Verlag, 1999, pp. 197–208.

[11] A. Konstantinidis, K. Yang, H.-H. Chen, Q. Zhang, Energy aware topology control for wireless sensor networks using memetic algorithms, Elsevier Computer Communications 30 (14-15) (2007) 2753–2764.

[12] T. Melodia, D. Pompili, I. F. Akyildiz, On the interdependence of distributed topology control and geographical routing in ad hoc and sensor networks, IEEE Journal on Selected Areas In Communications 23 (3) (2005) 520–532.

[13] X. Cheng, B. Narahari, R. Simha, M. Cheng, D. Liu, Strong minimum energy topology in wireless sensor networks: Np-completeness and heuristics, IEEE Transactions on Mobile Computing 2 (3) (2003) 248–256.

[14] P. Santi, D. M. Blough, F. Vainstein, A probabilistic analysis for the range assignment problem in ad hoc networks, in: Managing Next Generation Networks and Services, Lecture Notes in Computer Science, Vol. 4773/2007, Springer Berlin / Heidelberg, 2007, pp. 523–526.

[15] P. Cheng, C. N. Chuah, X. Liu, Energy aware node placement in wireless sensor networks, in: IEEE Global Telecommunications Conference, Vol. 5, 2004, pp. 3210–3214.
[16] Y. Chen, C.-N. Chuah, Q. Zhao, Network configuration for optimal utilization efficiency of wireless sensor networks, Elsevier Ad Hoc Networks 6 (2008) 92–107.
[17] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley and Sons, 2002.
[18] A. Konstantinidis, K. Yang, Q. Zhang, An evolutionary algorithm to a Multi-Objective Deployment and Power Assignment Problem in wireless sensor networks, in: IEEE Global Communications Conference, GlobeCom08, Vol. AH16, 2008, pp. 475–481.
[19] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, Y. F. Hu, Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards, Elsevier Computer Communications 30 (2007) 1655–1695.
[20] S. Toumpis, Mother nature knows best: A survey of recent results on wireless networks based on analogies with physics, Computer Networks 52 (2) (2008) 360–383.
[21] S. Toumpis, L. Tassiulas, Optimal deployment of large wireless sensor networks, IEEE Transactions on Information Theory 52 (7) (2006) 2935–2953.
[22] J. Knowles, M. Oates, D. Corne, Advanced multi-objective evolutionary algorithms applied to two problems in telecommunications, BT Technology Journal 18 (4) (2000) 51–65.
[23] C. W. Ahn, R. S. Ramakrishna, A genetic algorithm for shortest path routing problem and the sizing of populations, IEEE Trans. Evolutionary Computation 6 (6) (2002) 566–579.
[24] K. P. Ferentinos, T. A. Tsiligiridis, Adaptive design optimization of wireless sensor networks using genetic algorithms, Elsevier Computer Networks 54 (1) (2007) 1031–1051.
[25] C. Reeves, Handbook of Metaheuristics, Kluwer, 2003, Ch. Genetic algorithms, pp. 65–82.
[26] D. B. Jourdan, O. L. de Weck, Layout optimization for a wireless sensor network using a multi-objective genetic algorithm, in: IEEE Semiannual Vehicular Technology, Vol. 5, 2004, pp. 2466–2470.
[27] R. Rajagopalan, P. K. Varshney, C. K. Mohan, K. G. Mehrotra, Sensor placement for energy efficient target detection in wireless sensor networks: A multi-objective optimization approach, in: Conference on Information Sciences and Systems, Baltimore, Maryland, 2005.
[28] S. C. Oh, C. H. Tan, F. W. Kong, Y. S. Tan, K. H. Ng, G. W. Ng, K. Tai, Multiobjective optimization of sensor network deployment by a genetic algorithm, in: IEEE Congress on Evolutionary Computation, 2007. CEC 2007, 2007, pp. 3917–3921.
[29] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[30] K. Kamyoung, M. T. Alan, N. Xiao, A multiobjective evolutionary algorithm for surveillance sensor placement, Environment and Planning B: Planning and Design 35 (2008) 935–948.
[31] J. Jia, J. Chen, G. Chang, Z. Tan, Energy efficient coverage control in wireless sensor networks based on multi-objective genetic algorithm, Computers and Mathematics with Applications 57 (11–12) (2009) 1756–1766.
[32] J. Jia, J. Chen, G. Chang, Y. Wen, J. Song, Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius, Computers and Mathematics with Applications 57 (11–12) (2009) 1767–1775.
[33] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. G. da Fonseca, Performance assessment of multiobjective optimizers: An analysis and review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132.
[34] N. Weicker, G. Szabo, K. Weicker, P. Widmayer, Evolutionary multiobjective optimization for base station transmitter placement with frequency assignment, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 189–203.
[35] Q. Zhang, H. Li, MOEA/D: A multi-objective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.
[36] W. Ye, J. Heidemann, D. Estrin, An energy-efficient mac protocol for wireless sensor networks, in: INFOCOM, 2002, pp. 1567–1576.
[37] X. Xu, S. Sahni, Approximation algorithms for sensor deployment, IEEE Transactions on Computers 56 (12).
[38] F. Glover, M. Laguna, Tabu Search, Norwell, MA: Kluwer, 1998.
[39] L. Liu, F. Xia, Z. Wang, J. Chen, Y. Sun, Deployment issues in wireless sensor networks, in: Lecture Notes in Computer Science, Mobile Ad-hoc and Sensor Networks, Vol. 3794, Springer Berlin / Heidelberg, 2005, pp. 239–248.
[40] D. C. Montgomery, Design and Analysis of Experiments, John Wiley & Sons, 2001.
[41] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, C. Gill, Integrated coverage and connectivity configuration in wireless sensor networks, in: Proceedings of the 1st international conference on Embedded networked sensor systems, 2003, pp. 28–39.
[42] M. Cardei, M. O. Pervaiz, I. Cardei, Energy-efficient range assignment in heterogeneous wireless sensor networks, in: International Conference on Wireless and Mobile Communications, 2006.
[43] Q. Zhang, J. Sun, E. P. K. Tsang, An evolutionary algorithm with guided mutation for the maximum clique problem, IEEE Trans. Evolutionary Computation 9 (2) (2005) 192–200.