Heuristic and Meta-Heuristic Approaches for Energy-Efficient Coverage-Preserving Protocols in Wireless Sensor Networks

Antonina Tretyakova, Franciszek Seredynski, Frédéric Guinand

Heuristic and Meta-Heuristic Approaches for Energy-Efficient Coverage-Preserving Protocols in Wireless Sensor Networks

Antonina Tretyakova, Franciszek Seredynski, Frédéric Guinand

Heuristic and Meta-Heuristic Approaches for Energy-Efficient Coverage-Preserving Protocols in Wireless Sensor Networks

Antonina Tretyakova
Dep. of Mathematics and Natural Sciences
Cardinal Stefan Wyszynski University in Warsaw
Warsaw, Poland
a.tretyakova@uksw.edu.pl

Franciszek Seredynski
Dep. of Mathematics and Natural Sciences
Cardinal Stefan Wyszynski University in Warsaw
Warsaw, Poland
f.seredynski@uksw.edu.pl

Frédéric Guinand
Dep. of Mathematics and Natural Sciences
Cardinal Stefan Wyszynski University in Warsaw
Warsaw, Poland
frederic.guinand@univ-lehavre.fr

ABSTRACT

Monitoring some sites using a wireless sensor network (WSN) may be hampered by the difficulty of recharging or renewing the batteries of the sensing devices. Mechanisms aiming at improving the energy usage at any moment while fulfilling the application requirements are thus key for maximizing the lifetime of such networks. Among the different methods for achieving such a goal, we focus on energy management methods based on duty-cycling allowing the sensors to switch between two modes: a high-energy mode (active) and a low-energy mode (sleep). In this paper we propose two new scheduling heuristics for addressing the problem of maximizing the lifetime of a WSN under the constraint of coverage of a subset of fixed targets. The first one is a stochastic greedy algorithm and the second one is based on applying Simulated Annealing (SA). Both heuristics use a specific knowledge about the problem. Experimental results show that while both algorithms perform well, greedy algorithm is preferable for small and medium sizes networks, and SA algorithm has competitive advantages for larger networks.

CCS CONCEPTS

• **Theory of computation** → **Simulated annealing**; • **Computing methodologies** → **Planning and scheduling**; • **Computer systems organization** → **Sensor networks**;

KEYWORDS

Maximum Lifetime Coverage Problem; Area Coverage in Wireless Sensor Networks; Meta-Heuristics; Simulated Annealing; Energy-Efficient Coverage Preserving Protocol

ACM Reference Format:
Antonina Tretyakova, Franciszek Seredynski, and Frédéric Guinand. 2017. Heuristic and Meta-Heuristic Approaches for Energy-Efficient Coverage-Preserving Protocols in Wireless Sensor Networks. In *Proceedings of The 20th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Miami, USA, November 2017 (MSWiM'2017),* 8 pages.
https://doi.org/10.1145/3132114.3132119

1 INTRODUCTION

Applications of wireless sensor networks (WSN) are manifold and their number increases every day [4]. Some of them are deployed in well-controlled environments allowing an easy maintenance, recharging or renewing of their battery. However, when deployed in hostile environments where human access is limited or even impossible, sensors may not be easily accessible and thus batteries cannot be recharged or renewed. Such scenarios may occur in nature: remote deserts, dense forests, flanks of volcanoes, as well as in urban environments: after an earthquake or on industrial sites after catastrophic events.

In such situations energy saving mechanisms are key to allow the WSN to work properly. The *lifetime* of a sensor network is then defined as the maximal duration for which application requirements are fulfilled, and most of the time the lifetime is limited by a lack of energy due to batteries depletion. This has motivated many works on energy conservation in WSN during the last two decades and, according to [3], three main methods may be considered: duty-cycling, data driven and mobility. The present work focuses on duty-cycling energy conservation methods. Such a method considers that each sensor is able to switch off its main energy-intensive circuits during some periods of time. Sensors may thus be either in *active state* during one time slot and in *sleep state* during another one. Deciding the scheduling of activity/idleness periods for the sensors is the main topic of the present work. Sensors iteratively perform mainly three tasks, sensing, computing and communicating, thus power management can concern any of these three steps, in the current work we mainly focus on energy saving mechanisms for the sensing task.

The scheduling decision depends mainly on the goal of the application, and many applications relying on WSN are interested in the monitoring of a specific zone. The underlying problem, regarding the application, is to obtain the best coverage for collecting data from the environment. There exist, at least, two different ways for an application to consider the coverage. The coverage of a surface or the coverage of

targets. In this work we are interested in this latter situation where the environment contains some regularly spaced targets. These targets are called points of interest (POI) in the present document.

Given all these elements, our problem consists in finding the best tradeoff between fulfilling the coverage constraint on the POIs and the minimization of the energy usage, which can be formulated as the maximization of the lifetime of the network while fulfilling the application requirements. In this paper we present the results obtained by two new knowledge-based algorithms to address this problem: a greedy heuristic and an algorithm based on Simulated Annealing (SA).

The rest of the paper is organized as follows. In the next section the main related works are outlined. Section 3 presents in detail the problem statement. The two following sections describe our greedy (Section 4) and Simulated Annealing-based algorithms (Section 5). Section 6 presents the results of simulation experiments and discusses the choices of the experimental parameters, followed by some concluding remarks.

2 STATE OF THE ART

The problem addressed in this paper is a variant of the *Maximum Lifetime Coverage Problem* (MLCP). In the present work we focus on a variant involving a set of fixed points or targets, called POIs (Point of Interest). Each POI may be monitored by more than one sensor at a time. The application requirement is that at any moment a minimum fraction of all the POIs have to be monitored. For maximizing the lifetime of the network we are interested in method scheduling the activity of each sensor that may be either in active or in sleep state.

One of the first attempt to solve this kind of problem is due to Slijepcevic and Potkonjak [13] in 2001. They proposed a heuristic composed of two parts: a first part for gathering all the sensors into mutually exclusive sets with a full coverage constraint, and a second part for scheduling the activity of each set of sensors in such a way that, at any moment, one and only one set can be active. Given the good results obtained in terms of network lifetime, this work inspired many other research. Among them, Abrams and his colleagues in [2] proposed to relax the full coverage constraint by allowing the algorithm to cover only a subset of the targets, and present three approximation algorithms for a variation of the set k-cover problem, this relaxation is also considered in our work. One year later, [5] proposed a theoretic model of the problem: the disjoint set covers problem (DSC) and they proved its NP-completness, then, in [6], the authors propose another formulation as the maximum set covers problem (MSC) and also prove its NP-completness, two complexity results that motivate the search for efficient heuristics for providing approximate solutions to this problem.

During the last decade many nature-inspired approaches addressing similar problems have been published [11]. Among them are genetic algorithms [12], multi-objective genetic algorithm [9], evolution strategies [7], memetic algorithm [14],

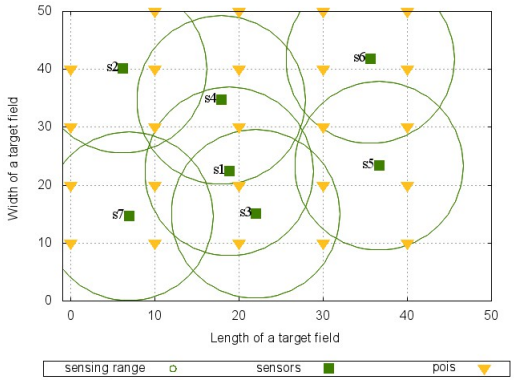

Figure 1: An example of sensor network deployed over the target field

particle-swarm optimization [1], differential evolution [16], etc. Our recent research applying a general purpose nature-inspired metaheuristic to solve this problem [15] shows that they are not enough efficient. As we already mentioned, the quality of solutions and computational complexity are not satisfactory. Coverage problems were considered under different scenarios and types of WSNs, namely Wireless Multimedia Sensor networks in dynamic environment [7], Wireless camera-based Sensor Networks [14], Directional Sensor Networks [8]. These assumptions lead to different problems statement and, therefore, each approach should be modified to enable to solve coverage problem in another type of WSNs.

Our approach differs from these works by the formulation of the lifetime and also by

3 PROBLEM STATEMENT

Let us consider a sensor network $S = \{s_1, \ldots, s_N\}$ consisting of N sensor nodes randomly distributed over a given *target field* F. F is considered to be a two-dimensional rectangular area. F contains a set of points of interest (POIs) regularly distributed on a 2D-grid, as illustrated on Figure 1. A sensor s_j is defined as a point of coordinates (x_j, y_j) in the two-dimensional area, with a sensing range R_s and battery capacity b. All sensors are identical: same sensing range and same initial battery capacity.

It is assumed that each sensor can work in two modes: *active mode* and *sleep mode*. In active mode a sensor s_i is able to monitor all the POIs that are within a disk centered at s_i with a radius equals to R_s and we consider that time is divided in time intervals of same length. We denote by $state_i^j$ the state of sensor s_i during time interval t_j. If the sensor s_i is active during this time interval, $state_i^j = 1$ and $state_i^j = 0$ if s_i is in sleep mode during time interval t_j.

Below we give a number of definitions concerning the problem statement.

Definition 3.1. A sensor $s_i(x_i, y_i)$ covers a POI *p(x, y)* iff the Euclidean distance between them $d(s_i, p) \leq R_s$

We denote by $POIs_{obs}(s_i)$ the set of POIs covered by sensor s_i when in active mode, and by $cov(s_i)$ the number of such POIs:

$$cov(s_i) = |POIs_{obs}(s_i)| \quad (1)$$

During time interval t_j, the set of POIs covered by the set of active sensors is denoted as $POIs_{obs}(t_j)$, i.e.

$$POIs_{obs}(t_j) = \cup_{i=1}^{N} POIs_{obs}(s_i)|_{state_i^j)=1} \quad (2)$$

Definition 3.2. The coverage of F during a given time interval t_j and denoted by $Cov(t_j)$, is the ratio of the number of observed POIs by sensors in active mode during t_j to all POIs:

$$Cov(t_j) = \frac{|POIs|_{obs}(t_j)}{|POIs|} \quad (3)$$

We make the assumption that the energy consumption of any sensor depends on its sensing range such that the energy used by any sensor is the same whatever the considered time interval.

Thus, a potential solution for our problem is a schedule of the states of all the sensors and this schedule can be represented by a matrix with binary elements. Each column j of the matrix corresponds to the states of the sensors during time interval t_j and each row i represents the schedule of the activity of sensor s_i.

Definition 3.3. A schedule of a WSN is a binary $T_{max} \times N$ matrix denoted as Sol, i.e.

$$Sol(S)= \begin{pmatrix} state_1^1 & \cdots & state_1^j & \cdots & state_1^{T_{max}} \\ \vdots & \ddots & \vdots & & \vdots \\ state_i^1 & \cdots & state_i^j & \cdots & state_i^{T_{max}} \\ \vdots & & \vdots & \ddots & \vdots \\ state_N^1 & \cdots & state_N^j & \cdots & state_N^{T_{max}} \end{pmatrix}$$

where $state_j^i \in \{0,1\}$ is a state of sensor s_i during time interval t_j, 0 corresponds to the sleep state and 1 is related to the active state.

Definition 3.4. A schedule $Sol(S)$ is a feasible solution if the following inequality is met:

$$(\forall i)_{i=1,...,N} | \sum_{j=1}^{T_{max}} state_i^j \leq b \quad (4)$$

Definition 3.5. We call *Coverage String* the series of the coverage of F for time intervals from t_1 to $t_{T_{max}}$, i.e.

$$Coverage\ String = \{Cov(t_1), Cov(t_2), ..., Cov(t_{T_{max}})\} \quad (5)$$

For our problem we consider $q \in]0, 1]$ a real number representing a minimum threshold value for the coverage of F. If during a given time interval t_j, $Cov(t_j) \geq q$ the coverage requirement of the application is considered to be fulfilled for this time interval.

In that context, the lifetime of the WSN is the total number of time intervals for which $Cov(t_j) \geq q$

$$Lifetime(q) = \sum_{j=1}^{j=T_{max}} \eta_{t_j} \quad (6)$$

where

$$\eta_{t_j} = \begin{cases} 1 & \text{if } Cov(t_j) \geq q \\ 0 & \text{if } Cov(t_j) < q \end{cases}$$

The parameter T_{max} is a predefined number and should be set greater than the value of the lifetime of the WSN obtained by some method and less than the upper bound $Lifetime^{Up}$, i.e

$$Lifetime(q) < T_{max} \leq Lifetime^{Up} \quad (7)$$

Let us derive an upper bound of the lifetime. We make the assumption that F contains exactly N_{POIs}. The maximum number of POIs that can be monitored by sensors is equal to:

$$Max(N_{POIs}^{cov}) = \sum_{i=1}^{N} cov(s_i) \quad (8)$$

If we make the assumption that during each time interval the battery of an active sensor is decreased by one unit, then the maximum number of POIs that can be monitored by all the sensors until all the batteries are depleted is equal to: $Max(N_{POIs}^{cov}) \times b$

But during each time interval, only $N_{POIs} \times q$ POIs have to be monitored to fulfill the application requirement, so the maximum number of time intervals during which the coverage requirement is fulfilled, denoted by $Lifetime^{Up}$, is equal to:

$$Lifetime^{Up} = \frac{Max(N_{POIs}^{cov}) \times b}{N_{POIs} \times q} \quad (9)$$

And this bound is tight. Indeed, $Lifetime(q)$ could be equal to the value $Lifetime^{Up}$ if the set of all the sensors can be divided into disjoint subsets in such a way that active sensors of each subset cover exactly $N_{POIs} \times q$ POIs without intersections.

As a consequence the upper bound of the parameter T_{max} is $Lifetime^{Up}$ defined by equation 9 .

We consider Maximum Lifetime Coverage Problem (MLCP) as a scheduling problem applied to a WSN solving the area coverage problem in the discrete two-dimensional space. Under our formulation, a method addressing the MLCP problem should minimize the number of sensors monitoring redundant POIs during each time interval in order to minimize energy consumption.

$Lifetime(q)$ as defined in Equation 6 is our evaluation function. It should be maximized over the space of all feasible solutions.

Then, Maximum Lifetime Coverage Problem can be stated as follows:

Given

- a set of numbers $POIs = \{1, 2, ..., N_{POIs}\}$, each element represents an ordinal number of a POI,

- a family of N subsets $\overline{S} = \{S_1, S_2, ..., S_N\}$, where each element $S_i \subseteq POIs$, $i = 1, 2, ..., N$, is related to covered POIs by sensor s_i, and,
- an integer b representing the initial battery capacity.

Objective:

- find a maximal number m of subsets $\{S'_1, S'_2, ..., S'_m\}$, where $S'_j \subseteq \overline{S}$, such that the number of covered elements $|\cup_{S_i \in S'_j} S_i|$ meets the coverage ratio (see equation 10) and each element S_i of the family $\overline{S}$ is included in no more than b subsets $\{S'_{j_1}, S'_{j_2}, ..., S'_{j_z}\}$ (see equation 11), i.e.

$$(\forall j)_{j=1,...,m} | \frac{| \cup_{S_i \in S'_j} S_i |}{|POIs|} \geq q \tag{10}$$

$$(\forall i)(\exists j_1, ..., j_z) | S_i \in S'_{j_1}, ..., S_i \in S'_{j_z}, \tag{11}$$

where $i = 1, ..., N$ and $(\forall k)_{k=1,...,z}$ $1 \leq j_k \leq m$ and $z \leq b$.

An objective of searching a maximal number m of subsets satisfying (10) is equivalent to lifetime maximization and corresponds to scheduling activities of sensor nodes. The last equation (11) corresponds to the battery capacity restriction.

MLCP - specific knowledge. A searching process conducted by both a greedy heuristic and SA-based algorithm (see, below) incorporates the MLCP specific knowledge and is based on the following classification of columns in a schedule. All columns of the schedule solution are divided on three groups called three subsequences:

(1) *Redundant Subsequence (RS)*,
(2) *Excellent Subsequence (ES)*,
(3) *Unsatisfactory Subsequence (US)*.

Each subsequence groups time intervals such that a network of active sensors covers the target area with certain *coverage ratio*.

RS subsequence is introduced in order to reveal time intervals during which we potentially have redundant sensors and we wish to shift elements from *RS* into *ES*. *RS* is defined as a sequence of time intervals $\{t_i\}$, during which the *coverage* is greater than the *coverage ratio* q on at least δ, i.e.

$$cov(t_i) > q + \delta, \tag{12}$$

where δ is a small value representing a predefined declination from *coverage ratio* q. *ES subsequence* consists of time intervals $\{t_i\}$ in the schedule during which the *coverage* of the target field is within δ range from given *coverage ratio* q:

$$|cov(t_i) - q| \leq \delta \tag{13}$$

We use *ES* as a mark of high quality of the schedule solution regarding to lifetime. In order to prolong the WSN's lifetime a number of elements in ES should be increased and their values should be less than elements included in *US*. *US subsequence* is defined as time intervals $\{t_i\}$ in the schedule during which the *coverage* of the target field is less than the *coverage ratio* q on at least δ, i.e.

$$cov(t_i) < q - \delta \tag{14}$$

Let us denote a number of elements in RS, ES and US as N_R, N_E and N_U respectively. An example of RS, ES and US for a schedule built for the network depicted in Figure 1 with q=0.55 and δ=0.05 can be found in Figure 2. In the schedule, ones correspond to active sensors, zeros - switched off sensors. According to coverage, seven time intervals are divided on three types: ES = $\{t_3, t_7\}$, RS =$\{t_5\}$ and US = $\{t_1, t_2, t_4, t_6\}$. In this schedule *Lifetime(0.55)* of the network is equal to 3.

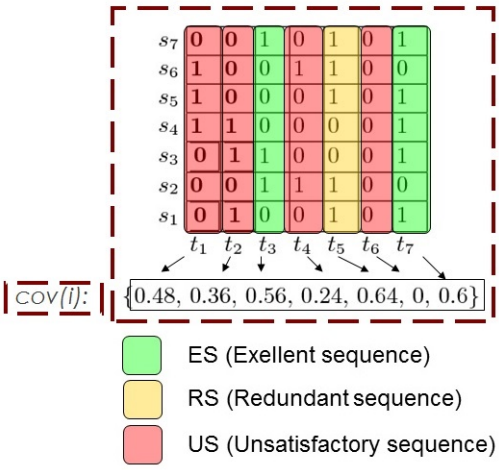

Figure 2: An example of RS, ES and US for a schedule with q=0.55 and δ=0.05.

4 A GREEDY HEURISTIC TO SOLVE MLCP

In this section, we present an iterative knowledge-based stochastic greedy heuristic to solve MLCP. The algorithm is based on constructing a tree of solutions. A root of the tree is a randomly created solution. In each iteration a solution called a predecessor is changed under two steps described below to form one more solution called a successor. The next iteration continues from the node corresponding to the best solution between a predecessor and its successor from the previous iteration.

The pseudocode of the algorithm is presented in algorithm 1. At each iteration a schedule is a subject of two types procedures, pseudo codes of which are sketched in Procedure 1 (Algorithm 2) and Procedure 2 (Algorithm 3).

The aim of procedure 1 is to improve a current solution via joining active subnetworks during time intervals when necessary coverage is not achieved. This purpose can be obtained by multiple shifting several columns from US toward ES or RS. A schedule is changed under the Procedure 1 as follows. Two new columns are generated by applying boolean-valued functions OR and AND to a pair of two values from same row from US columns (line 5 in algorithm 2). The new first column contains the values resulted of OR operator, and the second column contains the results of AND operator applied.

Algorithm 1 Pseudocode - Greedy algorithm for random initial solution.

1: *Input* :
2: WSN: $S = \{s_1, ..., s_N\}$, $s_i = \{(x_i, y_i), R_s, b\}$
3: Target field: $POIs = \{p_1, ...p_{N_{POIs}}\}$, $p_k = (x_k, y_k)$
4: A number of iterations: N_I,
5: Problem requirements: q, δ
6: Solution: T_{max},
7: initialize random $sol_{cur}(N, T_{max})$
8: k = 1
9: **for** $i \leftarrow 1$ **to** N_I **do**
10: compute US,
11: apply Procedure1(US, k) to form successor solution,
12: compute *Lifetime(q)*
13: **if** *Lifetime(q)* of the predecessor > *Lifetime(q)* of the successor **then**
14: k = k + 1
15: **end if**
16: compute RS, US for the successor,
17: apply Procedure2(RS, US) to the successor,
18: compute *Lifetime(q)* for the successor,
19: keep the best from the predecessor and its successor,
20: i = i+1
21: **end for**
22: *Output* :
23: **return** *sol*

Algorithm 2 Pseudocode - Procedure1(US, k)

1: *Input* :
2: *sol*
3: **for** $i \leftarrow 1$ **to** N_U - k + 1 **do**
4: **for** $j \leftarrow 1$ **to** k **do**
5: modify i and $i + j$ columns from US,
6: j = j + 1
7: **end for**
8: i = i +1
9: **end for**
10: **return** *sol*

In another words, in the result the first selected column contains "1" in all cells, if at least one of corresponding cells from two columns has contained "1". The second column contains the rest of two values which was not used in the first one. These steps are repeated k or no more than $\frac{N_U(N_U-1)}{2}$ times, where N_U is a number of elements in US.

As the algorithm proceeds it may happen that a new solution is not improved in the sense of *Lifetime(q)*. In that case a parameter k (a number of use Procedure 1) is increased by 1 (lines 12-14 in algorithm 1). Initially, k is equal to 1.

An example of the solution obtained due to one step of the Procedure 1(US, 1) is outlined in Figure 3. The schedule is constructed for a sensor network depicted in Figure 1 for seven consecutive time intervals. For coverage ratio q equal to 0.55 and coverage declination δ equal to 0.05, US consists of the three elements $\{1, 2, 4\}$. As N_U=3, then the procedure

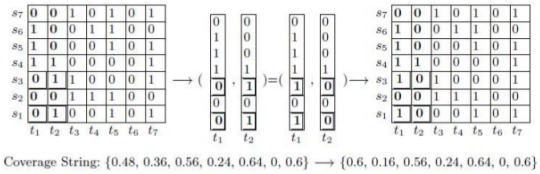

Figure 3: An example of a one iteration in procedure 1, coverage ratio q=0.55 and δ = 0.05.

repeats k or no more than 3 times. For instance, the initial schedule involves two columns corresponding to t_1 and t_2 time intervals. In the new solution two pairs related to sensors s_1 and s_3 in the two chosen columns change their values. Due to the operation (line 5 in algorithm 2) coverage at the first time interval is improved till 0.6. Thus, *Lifetime(0.55)* of the network is also improved by 1. The predecessor and its successor of solutions with its *coverage strings* are presented in the figure (left) and (right), respectively.

The solution obtained by the first modification (Procedure 1) is next changed by the Procedure 2. The aim of this stage is to reduce redundant consumption of energy, i.e. a randomly chosen sensor in active state from RS time interval is switched off and, next, is switched on during US time interval.

Algorithm 3 Pseudocode - Procedure2(RS, US)

1: *Input* :
2: *sol*
3: **for** $i \leftarrow 1$ **to** N_R **do**
4: modify the $i - th$ column in RS in the solution,
5: i = i +1
6: **end for**
7: **return** *sol*

The Procedure 2 (see, Algorithm 3) is executed on a current solution and consists of the following steps:

- from the $i - th$ RS column a cell is randomly selected with probability p_i:

where n_1 is a number of "1" cells in the column. Let us denote a row of the selected cell as j.

- the "0" cell in the first US column in $j - th$ row is changed to "1".

Therefore, the first selected cell is equal to 1. The second selected cell is taken as the first "0" cell from US and from the same row as previous cell was. The selected cells swap their values. If there is no a cell with the value "0" in US, the predecessor solution coincides with its successor. The above mentioned two steps are repeated consequently N_R times for each RS column.

An example of solution change due to one step of Procedure 2 is shown in Figure 4. A schedule constructed for a sensor network depicted in Figure 1 for seven consecutive time intervals. For coverage ratio q equal to 0.55 and coverage declination δ equal to 0.05, initially, coverage string is as

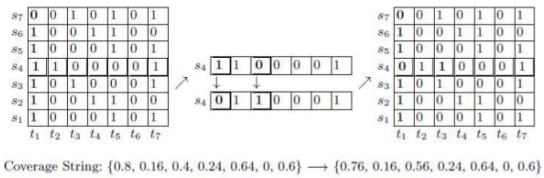

Figure 4: An example of a one iteration in procedure 2, coverage ratio q=0.55 and δ = 0.05.

follows {0.8, 0.16, 0.4, 0.24, 0.64, 0, 0.6}, where RS contains two elements t_1 and t_5. Therefore, Procedure 2 is executed twice via changing the first and fifth columns in the schedule. From the t_1 column with the probability 0.16 equal for all cells of "1"s values the one is taken, for an example the 4-th cell was selected. The chosen "1" cell changes its value into "0", while the first "0" cell of the corresponding row from US (in the picture such cell is taken from t_3 column) changes its value into the opposite one. Coverage of the schedule for the first time interval (from the column t_1) decreases, while the coverage for the t_3 increases. Coverage string of the successor solution becomes the following: {0.76, 0.16, 0.56, 0.24, 0.64, 0, 0.6}. Thus, a number of elements in ES is increased, *Lifetime(0.55)* is improved by 1 as well.

The predecessor schedule and its successor are evaluated by *Lifetime(q)* metric and the best one is saved as a current schedule for the next iteration to be applied. These steps are repeated until stop condition is met. The last saved schedule is a result of the algorithm.

5 SIMULATED ANNEALING ALGORITHM TO SOLVE MLCP

Simulated Annealing is one of the nature inspired metaheuristics based on the physical annealing process observed in glass and metal metallurgy and firstly proposed by Kirkpatrick, Gelatt and Vecchi [10].

The general idea of SA technique to solve MLCP is presented in Algorithm 4.

After setting initial parameters of the algorithm, in particularly an initial temperature T, the algorithm starts from a randomly generated initial solution (see, line 1), which becomes a current solution of the problem. Next, a new solution is randomly generated from a neighbourhood of the current solution (see, line 6). If a new solution is better than the current solution it becomes a new current solution (see, lines 9 - 10). If a new solution is worse than the current one it can be accepted as a new solution with probability $\exp^{-\frac{\Delta}{T}}$, which depends on the difference between a quality of the current and a new solution, and the current temperature T of the system (see, lines 11 - 13). This process of searching a solution under current temperature T is continued a predefined number of iterations (see, line 4 - 14), and after exceeding this number the temperature of the system is decreasing (see, line 15). If stop condition is not fulfilled (see, line 3) the

Algorithm 4 General SA

1: Generate initial solution Sol_{cur}
2: Set temperature T
3: **while** stop condition is not fulfilled **do**
4: **for** $i \leftarrow 1$ **to** L **do**
5: life = compute fitness of Sol_{cur}
6: Generate neighbouring solution $SolN(Sol_{cur})$
7: lifeN = compute fitness of $SolN$
8: Δ = life - lifeN
9: **if** $\Delta \leq 0$ **then**
10: $Sol_{cur} = SolN$
11: **else**
12: $Sol_{cur} = SolN$ with probability $\exp^{-\frac{\Delta}{T}}$
13: **end if**
14: **end for**
15: decrease T
16: **end while**
17: *Output* :
18: Sol_{cur}

algorithm returns to the main loop of searching a solution under a given temperature T.

The main problem related to the application of SA to solve MLCP is a generation of a new solution in the neighbourhood of a current one. Because of specific limitations concerning feasible solutions of MLCP, generation of valid solutions is a difficult problem. To solve the problem we propose an algorithm which uses information about MLCP. Pseudocode of the algorithm is presented below (see, Algorithm 5.

Algorithm 5 Generating neighbouring solution

1: *Input* :
2: Sol
3: k
4: compute RS, US
5: **for** $i \leftarrow 1$ **to** k_{neigh} **do**
6: **for** $j \leftarrow 1$ **to** $RS.size()$ **do**
7: choose a random cell of "1" value from $i-th$ RS column, the row of the cell let us denote as l
8: find the first "0" cell from US and $l-th$ row
9: swap the values of the chosen pair
10: $j = j + 1$
11: **end for**
12: $i = i + 1$
13: **end for**
14: *Output* :
15: Sol

Generating neighbouring solutions. Generating subsequent solutions is based on a swap of a pair of opposite values in one row of a current schedule. A neighbouring solution differes from the current solution by a number of bits changed comparatively with the given solution. Let us call this characteristic defined by a number of changed pairs of bits, as *neighbourhood size* and denote it as k_{neigh} - neighbourhood.

In such a way, in the case of changing two random cells of a schedule, we obtain a solution in 1-neighbourhood from a given solution. A random neighbour is generated as follows: k_{neigh} times two opposite values chosen at random from the same row are swapped (see, Algorithm 5). An additional information about a solution is computed, such as coverage of each time slot according to which the time line is divided on RS, ES and US. The idea of knowledge-wise neighbourhood generating procedure is to switch off an active sensor from redundant subsequence in order to reduce a number of redundantly covered POIs and to switch it on in the first unsatisfactory subsequence with the aim of increasing coverage at the additional time interval. These steps may increase lifetime of the generated solution.

6 EXPERIMENTAL RESULTS

In this section, we present some preliminary results of experimental study of the proposed algorithms. More experiments are yet to come in order to compare the performances of these two methods with other state-of-the-art solutions that have to be adapted to our variant of the MLCP. Performance of each algorithm is evaluated on the test of nine *problem instances*. Each *problem instance* is defined by a number of sensors N, sensor coordinates (x, y), a number of POIs uniformy allocated on the target field, sensing range R_s and battery capacity b.

The coverage requirement consists of two real number: percentage of coverage required q and admissible declination δ from the q-requirements.

We consider WSN consisting of a number N of sensors equal to 100, 200 and 300, respectively. For each value of N, we created three instances, which differ by random allocation of sensors, so nine instances were used in experimental study. Each instance is described as Instance{*indicator of network size*}{*order number of WSN instance*}, where N is equal to *indicator of network size* times 100. To give an example, Instance23 represents the third instance of WSN consisting of 200 sensors. For testing purposes the following instances were created:

- *instance11*, *instance12*, *instance13* for *N=100*,
- *instance21*, *instance22*, *instance23* for *N=200*,
- *instance31*, *instance32*, *instance33* for *N=300*.

The values of sensing range R_s and battery capacity b are equal to 20 m and 10 $t.u.$. As a targer field we consider a uniform distribution of POIs over the square area of dimensions 100×100 m^2 and between two neighbor POIs are separated by a step g equal to 10 m.

The algorithm's parameters should be chosen as the set of the best values for each of the algorithms: greedy heuristic and SA. SA is defined by the following values. Temperature is cooled according to the logarithmic scheme with initial temperature 50, length of the temperature cycle 25, the frozen level 10 and maximal number of iterations 100. The termination condition is as follows: exceeding maximal number of iterations or achieving the frozen temperature level. Greedy heuristic needs to set a number of iterations equal to 150, after which the algorithm stops.

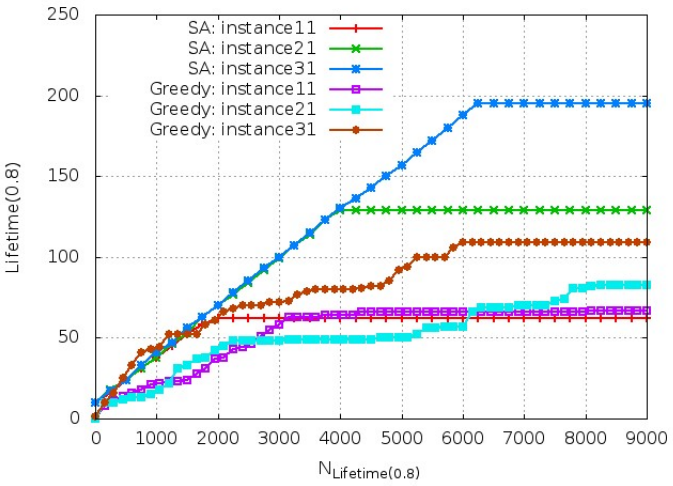

Figure 5: Example of runs of two algorithms: *Greedy* **and SA for instances consisting of 100, 200 and 300 nodes;** q**=0.8,** $R_s = 20$**,** $b = 10$**.**

All experimental results in this section are based on averaging of ten runs for each problem instance.

Experiment 1. An example of a typical run of these two algorithms is presented in Figure 5, which presents the dynamics of $Lifetime(0.8)$ obtained by greedy and SA for three instances consisting of 100, 200 and 300 nodes as a function of a number of computation of the $Lifetime(q)$ function.

One can notice that behaviour of both algorithms is different. While SA with a linear speed achieves its maximal value of $Lifetime(0.8)$ (equal to near 190 for 300 nodes, near 130 for 200 nodes, and near 60 for 100 nodes) the *greedy algorithm* moves to its maximum with different speeds achieving $Lifetime(0.8)$ equal to near 108 for 300 nodes, near 80 for 200 nodes, and near 62 for 100 nodes. SA finds much better solution for 300 nodes (aroud 190 against around 108) with approximatelly the same computational costs (around 6000 iterations). A similar situation is for 200 nodes, but computational costs of *greedy algorithm* is twice higher than for SA (around 4000 iterations against around 8100 iterations), but for 100 nodes *greedy algorithm* is slightly better (60 against 62) with twice larger computational costs (around 2000 against 4150 iterations).

Experiment 2. In order to study more detaily the behaviour of both algorithms, the next experiment was conducted where results were obtained through multiple runs of algorithms. Table 1 contains maximal, average and standard deviation of $Lifetime(q)$ function values obtained by *greedy* and SA.

One can see from the table that for relatively small network with 100 nodes the average and the maximal values of $Lifetime(q)$ are better for *greedy algorithm*. The similar results are for a network with 200 nodes, but the difference

Table 1: Maximal, average with standard deviation values of Lifetime(q) obtained by two algorithms: *greedy* and SA for nine instances; $q = 0.8$, $R_s = 20$, $b = 10$.

	greedy		SA	
instance	Max	Avg $\pm$ σ	Max	Avg $\pm$ σ
$N = 100$				
*instance*11	**77**	**74.0 $\pm$ 6.09**	71	68.0 $\pm$ 1.73
*instance*12	**82**	**79.0 $\pm$ 5.3**	77	74.0 $\pm$ 1.41
*instance*13	**74**	**71.0 $\pm$ 5.39**	71	68.0 $\pm$ 1.41
$N = 200$				
*instance*21	**154**	**152.0 $\pm$ 7.62**	153	147.0 $\pm$ 2.0
*instance*22	**152**	**148.0 $\pm$ 8.49**	149	144.0 $\pm$ 2.0
*instance*23	**156**	**151.0 $\pm$ 6.09**	152	149.0 $\pm$ 1.73
$N = 300$				
*instance*31	230	226.0 $\pm$ 6.0	**232**	**226.0 $\pm$ 2.23**
*instance*32	225	220.0 $\pm$ 7.55	**227**	**223.0 $\pm$ 2.0**
*instance*33	**227**	221.0 $\pm$ 22.21	**227**	**223.0 $\pm$ 2.0**

between results provided by both algorithms is small. For the network with 300 nodes SA provides better results than *greedy algorithm*, but also the difference between results provided by both algorithms is small.

It is worth to notice that standard deviation computed for SA in all cases is better than σ-values computed for the results provided by *greedy algorithm*. This indicates that SA is more stable than *greedy algorithm*. Therefore, we can assume that in the case of larger problem instances SA should provide better solutions than *greedy*.

CONCLUSION

In this paper the problem of lifetime maximization in WSNs stated as MLCP with assumption of not full coverage defined by a coverage ratio requirement q was considered. The problem belongs to a class of NP-hard problems characterized by high computational complexity, what motivates the use of algorithms providing approximate solutions. To solve the problem we proposed and studied two centralized knowledge-based algorithms: stochastic greedy heuristic and simulated annealing algorithm.

Results of experimental study of algorithms show that while both algorithms perform well in a relatively wide spectrum of nodes number, the greedy heuristic is sligtly better for low and medium sizes of a network, and SA algorithm becomes competitive for larger networks.

REFERENCES

[1] Mohammadjavad Abbasi, Muhammad Shafie Abd Latiff, Ali Modirkhazeni, and Mohammad Hossin Anisi. Optimization of wireless sensor network coverage based on evolutionary algorithm. *IJCCN International Journal of Computer Communications and Networks*, 1(1), December 2011.

[2] Z. Abrams, A. Goel, and S. Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In *Third International Symposium on Information Processing in Sensor Networks, 2004. IPSN 2004*, pages 424–432, April 2004.

[3] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks, Elsevier*, 7(3):537–568, 2009.

[4] T. Arampatzis, J. Lygeros, and S. Manesis. A survey of applications of wireless sensors and wireless sensor networks. In *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.*, pages 719–724, June 2005.

[5] M. Cardei and D.-Z. Du. Improving wireless sensor network lifetime through power management organization. *Wireless Networks*, 11:333–340, 2005.

[6] M. Cardei, M. T. Thai, Yingshu Li, and Weili Wu. Energy-efficient target coverage in wireless sensor networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1976–1984. IEEE, March 2005.

[7] Hossein Fayyazi, Mohammad Sabokrou, Mojtaba Hosseini, and Ali Sabokrou. Solving heterogeneous coverage problem in wireless multimedia sensor networks in a dynamic environment using evolutionary strategies. *ICCKE2011, International Conference on Computer and Knowledge Engineering, Ferdowsi University of Mashhad, Mashhad, Iran (2011)*, October 13-14 2011.

[8] Joon-Min Gil and Youn-Hee Han. A target coverage scheduling scheme based on genetic algorithms in directional sensor networks. *Sensors, doi:10.3390/s110201888*, 11:1888–1906, 2011.

[9] Jie Jia, Jian Chen, Guiran Chang, and Zhenhua Tan. Energy efficient coverage control in wireless sensor networks based on multi-objective genetic algorithms. *Computers Mathematics with Applications, Pergamon Press*, pages 1756–1766, June 2008.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *SCIENCE*, 220(4598):671–680, 1983.

[11] P. Musilek, P. Krmer, and T. Barto. Review of nature-inspired methods for wake-up scheduling in wireless sensor networks. *Swarm and Evolutionary Computation*, 25:100 – 118, 2015.

[12] B. Sahoo, V. Ravu, and P. Patel. Observation on using genetic algorithm for extending the lifetime of wireless sensor networks. *IJCA Special Issue on 2nd National Conference- Computing, Communication and Sensor Network (CCSN)*, pages 9–13, 2011.

[13] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. *IEEE International Conference on Communications, Vol 2*, pages 472–476, June 2001.

[14] C. K. Ting and C. C. Liao. A memetic algorithm for extending wireless sensor network lifetime. *Information Sciences, Elsevier Inc.*, pages 4818–4833, August 2010.

[15] A. Tretyakova and F. Seredynski. Application of evolutionary algorithms to maximum lifetime coverage problem in wireless sensor networks. *27-th IEEE International Parallel and Distributed Processing Symposium, NIDISC Workshop, Boston, USA (2013)*, 2013.

[16] Yulong Xu, Jianan Fang, Wu Zhu, and Wenxia Cui. Differential evolution for lifetime maximization of heterogeneous wireless sensor networks. *Hindawi Publishing Corporation Mathematical Problems in Engineering (2013), http://dx.doi.org/10.1155/2013/172783*, 2013.