https://doi.org/10.1007/s11590-016-1072-y

An exact algorithm to extend lifetime through roles allocation in sensor networks with connectivity constraints

Technical Report n 63994, Department of Mathematics, University of Salerno, 29/10/2015.

Francesco Carrabs[a], Raffaele Cerulli[a], Ciriaco D'Ambrosio[a], Andrea Raiconi[a]

[a] *Department of Mathematics, University of Salerno, Via Giovanni Paolo II 132, 84084 Fisciano, Italy (e-mail: {fcarrabs,raffaele,cdambrosio,araiconi}@ unisa.it).*

Abstract

The Maximum Lifetime Problem with Role Allocation and Connectivity Constraints consists of defining an optimal scheduling of the activities in a wireless sensor network in order to ensure that, in each instant of time, the activated sensors can monitor all the points of interest (targets) and route the collected information to a given processing facility. Each sensor is assigned a role, depending on whether it is actually used to monitor the targets, to forward information or kept idle, leading to different battery consumption ratios. We propose a column generation algorithm that embeds a highly efficient genetic metaheuristic for the subproblem. Moreover, to optimally solve the subproblem, we introduce a new formulation with an inferior number of integer variables with respect to a previous one proposed in the literature. Finally, we propose a stopping criterion to interrupt the optimal resolution of the subproblem as soon as a favorable solution is found. The results of our computational tests show that our algorithm consistently outperforms previous approaches in the literature, and also improves the best results known to date on some benchmark instances.

Keywords: Wireless Sensor Network, Roles Allocation, Column Generation, Genetic Algorithm, Connectivity Constraints

1. Introduction

Recent technology advances in several fields, including miniaturization and wireless communications, have enabled the use of Wireless Sensor Networks to monitor and process information in a wide scope of different real world applications. One may consider, for instance, new technology concepts such as the *Internet of Things* [2], and its applications, which include the research project known as *Array of Things*[1].

In comparison, battery technologies have seen smaller improvements. Optimizing energy consumption remains therefore an issue of great relevance, and has been object of many research efforts. Indeed, a problem of interest is related to prolonging for as much as possible the *network lifetime*, that is, the amount of time over which the network can keep under observation some target locations (generally simply referred to as *targets*). This issue is particularly relevant for networks composed of a large number of low-cost sensors, where the fundamental idea is to optimize the energy expense of the individual devices by exploiting their redundancy. In more detail, the problem is faced by appropriately scheduling the sensor activities, in such a way that at any given time only some sensors, that can jointly monitor all targets, are activated, while all the others are switched off or kept in a low-power idle mode. Such a set is usually called *cover*, and by extension, it is said to cover the set of the targets. It was shown in [5] that allowing non-disjoint covers can greatly prolong the network lifetime, and that the resulting problem, known as the *Maximum Lifetime Problem* (MLP), is NP-Complete. Exact, heuristic and approximation algorithms have been proposed in the literature for its resolution ([4], [5], [9], [14]).

Many variants of MLP have been proposed and studied, as well. Among others, resolution approaches have been proposed for the cases in which only a subset of targets need to be monitored by each cover ([9], [17], [22]), or that consider networks composed of heterogeneous sensing devices ([3], [7]), or sensing

[1] https://arrayofthings.github.io

devices with adjustable ranges ([6], [12], [13], [15], [21]).

Other MLP variants take into account the need to transmit the sensed information to a central processing node, which is defined in the literature using different names, such as *base station*, *sink* or *gateway* ([1], [8], [11], [18], [20]).

An interesting variant belonging to this latter area, called *Maximum Lifetime Problem with Role Allocation and Connectivity Constraints* (CMLP-MR), was recently faced in [10]. Consider an underlying undirected graph where two sensors (or a sensor and the base station) are connected by an edge if and only if they are close enough to communicate. In CMLP-MR the sensors composing each cover have to ensure the coverage of all targets and, together with the base station, they have to induce a subtree in this graph. Such a tree, that we name *routing tree*, represents how the information is routed to the base station. Furthermore, since some sensors are used only to transmit information, it is realistic to assume that they consume their battery at a lower rate, during the cover activation period, with respect to the ones with sensing duties. Such sensors are defined *relay nodes*, while the ones used to monitor the targets are called *source nodes*. Therefore in CMLP-MR, besides individuating feasible covers and assigning activation times to them, appropriate roles must also be assigned to the sensors within each cover. In [10] the authors propose an exact resolution approach for the problem, which is based on the column generation method (CG). Starting from an initial subset of feasible covers, this algorithm alternates the resolution of a LP formulation (the *master problem*) which finds the optimal solution for the current subset, and of an auxiliary problem (the *pricing subproblem*, or simply *subproblem*) which identifies new favorable covers that may improve the objective function value, until it finally certifies the optimality of the latest solution found by the master problem. In order to solve the subproblem, [10] presents three different approaches, namely an ILP formulation, a branch-and-cut algorithm based on Benders' decomposition and a method based on constraint programming.

In this work, we present a new CG algorithm for CMLP-MR that presents several new features with respect to the ones proposed in literature. We propose

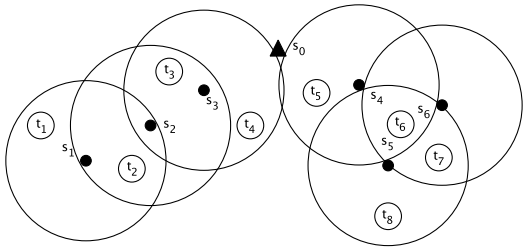
an efficient genetic algorithm (GA) to solve the subproblem, which is able to return multiple favorable covers at each iteration. Whenever the GA fails, we solve the subproblem by using an ILP model, which contains an inferior number of integer variables with respect to the one reported in [10]. Furthermore, we prematurely interrupt the ILP resolution as soon as it is able to find a favorable cover, avoiding to solve it to optimality whenever possible. As will be shown in the computational tests, thanks to these features our algorithm is generally able to solve the instances proposed by [10] in noticeably less computational time, also providing optimal solutions for instances that were unsolved to date.

The rest of the paper is organized as follows. Section 2 formally defines CMLP-MR. Section 3 describes the CG algorithm structure, and presents the two alternative methods to solve the pricing subproblem. Section 4 presents the computational test results. Our conclusions and insights on future work are contained in Section 5.

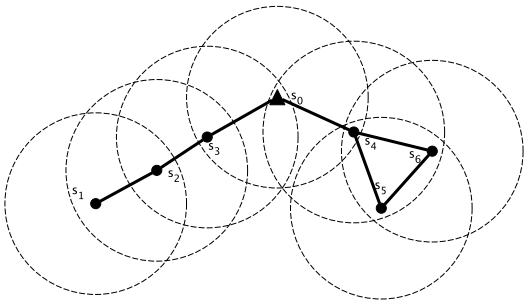
2. Problem Definition

Let $S = \{s_0, s_1, \ldots, s_m\}$ denote the set composed of the base station s_0 and the sensor nodes, and let $T = \{t_1, \ldots, t_n\}$ be the set of the targets. For each sensor s_i and target t_k, let δ_{ki} be a parameter whose value is 1 if t_k is located within the sensing range of the sensor, and 0 otherwise. For each subset S' of $S \setminus \{s_0\}$ and each target t_k, let parameter $\Delta_{kS'}$ be equal to 1 if $\delta_{ki} = 1$ for at least a sensor $s_i \in S'$, and 0 otherwise. Furthermore, consider an undirected *connectivity graph* $G = (S, E)$, such that $(s_i, s_j) \in E$ if and only if they are close enough to exchange data.

Consider for instance the network represented in Figure 1(a), with $S = \{s_0, \ldots, s_6\}$ and $T = \{t_1, \ldots, t_8\}$. The circles centered at the sensors represent their sensing ranges; hence, for instance, s_2 can monitor t_2 and t_3 ($\delta_{22} = \delta_{32} = 1$). Figure 1(b) shows the transmission ranges of the elements of S, represented by the circles with dashed borders, and the resulting connectivity graph. It can be seen that, for instance, the base station (represented by a triangle) is



(a)



(b)

Figure 1: Example wireless sensor network (a); transmission ranges and connectivity graph (b).

connected to s_3 and s_4, and the subgraph induced by $\{s_4, s_5, s_6\}$ is complete, since these three sensors are included into each other's transmission range.

Let $\mathcal{R} = \{0, 1, 2\}$ be the roles that can be assumed by each sensor, where 0 corresponds to the switched off (or idle) mode, 1 to the relay role and 2 to the source role. An allocation of roles to sensors defines a partition $P_p = \{S_p^0, S_p^1, S_p^2\}$ of the set $S \setminus \{s_0\}$, where S_p^r is the subset of sensors assigned to role r. In CMLP-MR, a feasible cover corresponds to such a partition P_p, for which the following two conditions hold:

- $\Delta_{kS_p^2} = 1$ for each target t_k, hence the source nodes cover the whole set of targets;
- the subgraph of G induced by the set of nodes $\{s_0\} \cup S_p^1 \cup S_p^2$ is connected, and therefore it contains a routing tree.

By extension, a partition representing a feasible cover is defined to be feasible as well. From now on, we will use the terms cover or partition interchangeably, and refer to feasible ones where unspecified.

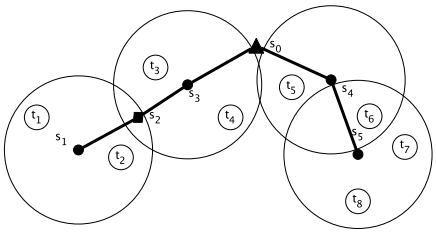


Figure 2: Partition $\{\{s_6\}, \{s_2\}, \{s_1, s_3, s_4, s_5\}\}$ for the network in Figure 1

A feasible partition and its related routing tree for the network in Figure 1 are illustrated in Figure 2. The chosen set of source nodes is $\{s_1, s_3, s_4, s_5\}$, that can cover all the targets. Sensor s_2 is needed as a relay (represented by a square in the figure), since the subgraph induced by $\{s_0, s_1, s_3, s_4, s_5\}$ would not be connected otherwise. Since it does not have covering purposes, its sensing range is not shown in the figure. Sensor s_6 is not needed, therefore it is kept idle and is not shown in the figure. The resulting partition is $\{\{s_6\}, \{s_2\}, \{s_1, s_3, s_4, s_5\}\}$.

Each role $r \in \mathcal{R}$ is assigned an energy consumption rate $l^r \geq 0$, representing how quickly a sensor in such a role consumes its battery. Since role 0 is by definition used to preserve energy, and since a source node may also transmit information, it is assumed that $l^0 \leq l^1 \leq l^2$. Furthermore, for each sensor s_i, let $b_i \geq 0$ represent its battery charge. In a CMLP-MR solution, a sensor can be allocated to different roles in different partitions. The energy consumption needed to switch roles is assumed to be negligible.

For a given feasible partition P_p, each sensor s_i and each role r, let a_{ip}^r be a binary parameter which is equal to 1 if $s_i \in S_p^r$ and 0 otherwise. Assuming to know in advance the whole set of feasible partitions $\{P_1, \ldots, P_M\}$, CMLP-MR can be represented using the following LP formulation:

$$[\mathbf{P}] \max \sum_{p=1}^{M} w_p \tag{1}$$

$$\sum_{p=1}^{M} \sum_{r \in \mathcal{R}} l^r a_{ip}^r w_p \leq b_i \qquad \forall s_i \in S \setminus \{s_0\} \tag{2}$$

$$w_p \geq 0 \qquad \forall p = 1, \ldots, M \tag{3}$$

Continuous variables w_p represent the activation times assigned to the partitions in the optimal solution. Constraints (2) ensure that the sum of the energy consumptions of each sensor s_i in the partitions is not greater than its maximum battery capacity.

The model can not be used in practice on large scenarios, due to the exponentially large number of feasible partitions (and thus of variables in the model). For this reason, we developed the CG approach described in Section 3.

3. Column Generation approach

Our exact CG approach to solve CMLP-MR, from now on called ECG-MR, allows us to find the optimal solution for [**P**] without explicitly enumerating the variables corresponding to each feasible partition. We start by solving the master problem, corresponding to a version of [**P**] which only uses a subset of feasible partitions. We then use the dual prices π_i corresponding to each sensor s_i to build the feasible partition corresponding to the nonbasic variable with minimum reduced cost, using an appropriately designed pricing subproblem. For a given partition P_p, the related reduced cost is evaluated as $\sum_{s_i \in S \setminus \{s_0\}} \sum_{r \in \mathcal{R}} l^r a_{ip}^r \pi_i - c_p$, where $c_p = 1$ is the coefficient of w_p in the objective function of [**P**]. If the minimal reduced cost is non-negative, the solution found by the master problem is optimal for [**P**]. Otherwise, the new variable (partition) could increase the objective function value, and is defined to be *attractive*; it is therefore added to the master problem, and the procedure iterates.

The subproblem is NP-Hard, since it is a particular set cover problem. Hence, in addition to an ILP formulation to solve it exactly, we propose a GA to obtain high-quality heuristic solutions whenever possible. Another feature of our GA is its ability to find several attractive partitions at once. In more detail, after solving the master problem, each iteration of ECG-MR first attempts to solve the subproblem using our GA. If the GA is able to find attractive partitions, they are added to the master problem, and the current iteration ends. If the GA fails, the ILP formulation is used instead. Given the complexity of solving such formulation, we prematurely interrupt its resolution as soon as it identifies an attractive partition that satisfies a predefined objective value threshold. It follows that the subproblem needs to be solved to optimality at least once, in the final iteration which is used to certify the optimality of the current solution.

We now describe our two methods to solve the subproblem in Sections 3.1 and 3.2. The generation of the subset of feasible partitions which is used to initialize the CG is described in 3.3.

3.1. GA to solve the subproblem

The use of evolutionary algorithms to produce feasible covers has been successfully exploited in the literature regarding lifetime optimization problems ([7], [9], [21]). As will be shown in Section 4, the GA proposed in this work is able to significantly speed-up the convergence of ECG-MR.

In our GA algorithm, each chromosome represents a partition, and each sensor is weighted according to its dual price. GA starts from a randomly built population of chromosomes, and it attempts to build new attractive partitions that could be added to the master problem. In this section we first define the chromosome representation of each individual, we then give a general overview of the algorithm, and finally we describe each genetic operator.

3.1.1. Chromosome Representation and Fitness Function

Each solution of the pricing subproblem is represented by a chromosome C, which is encoded using a vector of real values with length $|S|$. A chromosome that represents a feasible partition is defined by extension to be feasible.

Each position $C[i]$, for $i = 0, ..., m$, is called *gene* and is associated to $s_i \in S$. For $i \geq 1$, the gene $C[i]$ represents the role of the related sensor $s_i \in S \setminus \{s_0\}$ in the partition. In more detail, if C represents the partition P_p and $s_i \in S_p^r$, then the value reported in position $C[i]$ is the consumption rate l^r.

The $C[0]$ gene is associated to the base station. In this special case, its value is l^2 if, given the partition P_p corresponding to C, the connectivity subgraph induced by $\{s_0\} \cup S_p^1 \cup S_p^2$ is connected, and l^0 otherwise. Clearly, $C[0] = l^2$ in every feasible chromosome. In Figure 3, the example partition discussed in Section 2 and its chromosome representation are illustrated. The sensing role is assigned to the sensors $\{s_1, s_3, s_4, s_5\}$ that together cover all the targets. As a consequence, their consumption ratio is equal to l^2. The sensor s_6 is switched off and its consumption ratio is equal to l^0. Finally, the sensor s_2 is used to satisfy the connectivity constraints and its consumption ratio is set to l^1, and as introduced, the value for s_0 is set to l^2.

Using its operators, our GA ensures that every chromosome introduced in the population is feasible. The elements of the population are ranked according to a fitness function, which is computed by performing the dot product of the $C[i]$ values, for $i = 1, ..., m$, and the dual prices vector. It follows that by computing the fitness function value of a feasible chromosome, the reduced cost of the corresponding partition is obtained as well. Therefore, any feasible chromosome with a fitness function value lower than 1 corresponds to an attractive partition.

3.1.2. GA Structure

The structure of our GA is resumed by the pseudocode presented in Algorithm 1. The input is represented by the Wireless Sensor Network $WSN = (S, T)$, the connectivity graph $G = (S, E)$, the dual prices vector $\overline{dp}$ and the vector of consumption rates values $\bar{l} = [l^0, l^1, l^2]$. GA first initializes the pop-

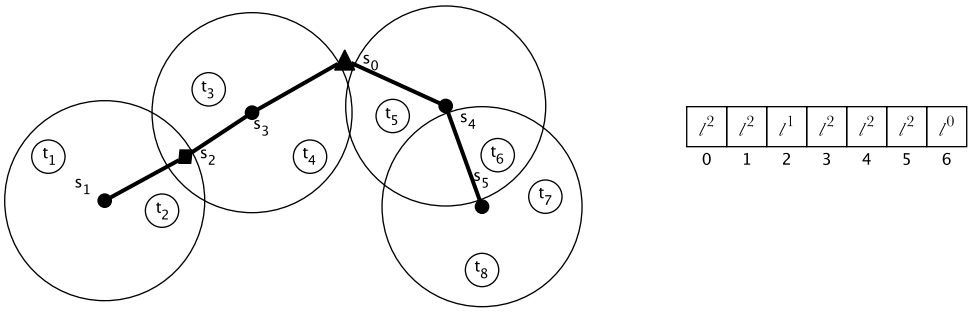


Figure 3: A partition and its chromosome representation

ulation *Pop* with randomly built individuals (line 1), storing the best fitness function value encountered in the process (line 2). Section 3.1.5 describes how the population initialization process works. Moreover, in this phase, each edge belonging to E is also assigned a cost, and a shortest path among each couple of elements of S is computed. The shortest paths are found by using the Floyd-Warshall algorithm (see [19]). These paths will be used while attempting to guarantee connectivity for the newly generated chromosomes, as discussed in Section 3.1.4.

The evolutionary process is carried out within the while loop reported in lines 4-21. Each iteration builds a new feasible chromosome, and the loop is iterated until one of two thresholds ($MaxIt$ and $MaxDup$) have been reached. The first one defines a maximum number of consecutive iterations without improvements with respect to the incumbent best fitness value $bestF$, while the latter sets a maximum number of consecutive created individuals that have a duplicate in the current population. Given two randomly selected members of *Pop*, defined *parents* (line 5), a new, possibly unfeasible one C (the *child*) is built using a crossover operator (line 6). Then, a mutation operator is applied to C (line 7) to increase its diversity. If the resulting chromosome is unfeasible, two additional operators, *cover feasibility* and *connect feasibility*, are applied to guarantee the coverage of all the targets and to ensure connectivity (lines 8-9). Finally, an operator which may allow to switch some sensors to role 0 in the special case

Algorithm 1: GA algorithm

Input: $WSN = (S, T)$, $G = (S, E)$, $\overline{dp}$, $\bar{l}$;

Output: $Chromos$ (a set of attractive partitions);

```
1  Pop ← initPop();
2  bestF ← bestFitness(Pop);
3  it ← 0, dup ← 0;
4  while it ≤ MaxIt and dup ≤ MaxDup do
5      (C_p1, C_p2) ← Tournament(Pop);
6      C ← Crossover(C_p1, C_p2);
7      C ← Mutation(C);
8      C ← CoverFeasibility(C);
9      C ← ConnectFeasibility(C);
10     if (l^1 == l^2) then
11         C ← redundancyRemoval(C);
12     if C ∉ Pop then
13         dup ← 0;
14         Pop ← insert(C);
15         if fitness(C) ≥ bestF then
16             it ← it + 1;
17         else
18             bestF ← fitness(C);
19             it ← 0;
20     else
21         dup ← dup + 1;
22 Chromos ← {C ∈ Pop | fitness(C) < 1};
23 return Chromos;
```

$l^1 = l^2$ is applied (lines 10-11). These operators are described in Sections 3.1.3-3.1.4.

If the new chromosome C is already inside the population, it is rejected, otherwise it will replace one of the $|Pop/2|$ worst chromosomes in Pop (lines 12-14). The $bestF$ value and two counters used to check whether $MaxIt$ or $MaxDup$ have been reached (it and dup) are updated according to the new solution found (lines 12-21).

As soon as a stopping condition is reached, the GA ends, returning all the attractive chromosomes contained in Pop (lines 22-23).

3.1.3. Crossover and Mutation

The crossover operator represents the first main step for the generation of a new chromosome. A binary tournament is used to select the parents; each parent is chosen by first selecting two random elements of Pop, and then taking among them the one with the better fitness function value. Given the parent chromosomes C_{p1} and C_{p2}, the child C resulting from the crossover operator is such that $C[i] = l^2$ if and only if $C_{p1}[i] = l^2$ and $C_{p2}[i] = l^2$ for $i \in \{1, \ldots, m\}$. All other positions of C are set to l^0. This choice was made since the relay nodes of the parents may not be useful in the case in which they connected source nodes missing in the child. Hence, we chose to delegate the task to efficiently connect the source nodes and the base station in C to the *connect feasibility* operator, described in Section 3.1.4.

In order to guarantee a diversification with respect to the parents, the mutation operator alters one of the genes corresponding to source nodes individuated by the crossover, switching its value from l^2 to l^0. If there are no such genes in C after the crossover, the mutation does not perform any operation, and C will be built from scratch by the two operators discussed in Section 3.1.4.

3.1.4. Coverage, Connectivity and Redundancy Operators

The set of source nodes resulting by the crossover and mutation operators may not guarantee the feasibility of the corresponding partition, in terms of neither coverage, nor connectivity. In order to make C feasible, we introduce two operators that fix each issue individually.

The *cover feasibility* operator checks the set of targets T' covered by the current source nodes in C; if $|T'| < |T|$, it randomly selects a sensor that would cover some new targets, and updates its gene in C from l^0 to l^2. The selection and activation of source nodes is iterated, until all targets are covered. In the last step of the cover feasibility operator, the presence of redundant source nodes is checked. To this end, by checking the targets which are covered by more than one source node, a list of genes that would not compromise coverage if they were switched from l^2 to l^0 is built. Then, one of them is randomly chosen and switched; the list is recomputed and the procedure is iterated until no source node can be removed.

The *connect feasibility* operator is then applied to activate sensors in the relay role, if needed. Given the partition P_p corresponding to the C chromosome resulting after the application of the cover feasibility operator, it is first checked whether the subgraph induced in G by $\{\{s_0\} \cup S_p^2\}$ is connected. In this case, $C[0]$ is switched to l^2, and no further operation is needed. Otherwise, a procedure based on the CHINS-Q heuristic for the Steiner Tree problem (see [16]) is applied. The aim of the procedure is to build a connected subgraph of G that includes all the nodes belonging to the set of *required nodes* $N \subset S$, containing the individuated source nodes and the base station s_0, attempting to favor the choice of additional sensors with low dual prices values. To this end, the cost assigned to each edge in G corresponds to the sum of the dual prices of its endpoints, assuming the base station s_0 to have a dual price equal to 0.

The procedure starts by building a new chromosome C', such that the gene $C'[i]$ of a single element $s_i \in N$ is set to l^2 and all other genes are set to l^0. Then, the algorithm finds the shortest path in G between s_i and an element $s_j \in N \setminus \{s_i\}$. The gene $C'[j]$ is set to l^2, and all the genes corresponding to the internal nodes of the path are set to l^1 in the chromosome. The described steps are iterated, finding in each iteration the shortest path between a node whose gene is equal to either l^2 or l^1 in C' and a required node whose gene is equal to l^0. Once the genes of all the elements of N are set to l^2 in C', a feasible chromosome has been obtained, and a routing tree has been individuated in

the process. The procedure is repeated $|N|$ times, varying in each iteration the element of N used as starting point. Finally, C is set to be equal to the chromosome with the best fitness function found, and is returned.

A particular situation occurs in the case in which the consumption rates of source and relay sensors are equal. In this case, that was taken into account in the computational tests performed in [10], the problem reduces to the Connected Maximum Lifetime Problem ([8], [11], [20]) in which the sensors roles are either source or idle. Indeed, if $l^1 = l^2$, all sensors added by the connect feasibility operator may be considered to be source nodes at no additional expense, and therefore coverage redundancy may have been added to the related partition. The *redunancy removal* operator, that is applied in this special case, works similarly to the procedure executed after the cover feasibility operator to remove redundancy. However, in order to guarantee that connectivity is not violated, only nodes that have degree 1 in the routing tree, individuated by the connect feasibility operator, are taken into account (eventually excluding s_0).

3.1.5. GA initialization

The initial population Pop of GA is composed of randomly built feasible chromosomes. Each chromosome is initialized by setting all its genes to l^0, and then updated using the operators discussed in Section 3.1.4. If a generated chromosome is equal to one that has been already added to Pop, to avoid having duplicate chromosomes it gets discarded. The initialization is iterated until either Pop contains a predefined $PopSize$ number of chromosomes, or a maximum number ($InitMaxDup$) of duplicated chromosomes have been generated. Since each chromosome built after the initialization replaces an older one, the cardinality of Pop does not change during each execution of GA.

3.2. ILP formulation to solve the subproblem

We propose a network flow formulation to build the nonbasic solution with minimum reduced cost. Our formulation is based on a directed graph $G^d = (S, E^d)$, such that for each couple of sensors s_i and s_j belonging to $S \setminus \{s_0\}$,

both arcs (s_i, s_j) and (s_j, s_i) belong to E^d if the related edge (s_i, s_j) belongs to E. Furthermore, for each $s_i \in S \setminus \{s_0\}$, $(s_0, s_i) \in E^d$ if the related edge belongs to E. Our proposed formulation is the following:

$$\textbf{[SP]} \min \sum_{s_i \in S \setminus \{s_0\}} \sum_{r \in \mathcal{R}} \pi_i l^r y_i^r \tag{4}$$

$$\sum_{r \in \mathcal{R}} y_i^r = 1 \qquad \forall s_i \in S \setminus \{s_0\} \tag{5}$$

$$\sum_{s_i \in S \setminus \{s_0\}} \delta_{ki} y_i^2 \geq 1 \qquad \forall t_k \in T \tag{6}$$

$$\sum_{(s_0, s_i) \in E^d} f_{0i} = \sum_{s_i \in S} \sum_{r=1}^{2} y_i^r \tag{7}$$

$$\sum_{(s_j, s_i) \in E^d} f_{ji} - \sum_{(s_i, s_j) \in E^d} f_{ij} = \sum_{r=1}^{2} y_i^r \qquad \forall s_i \in S \setminus \{s_0\} \tag{8}$$

$$\sum_{r=1}^{2} y_i^r \leq \sum_{(s_j, s_i) \in E^d} f_{ji} \leq (|S| - 1) \sum_{r=1}^{2} y_i^r \qquad \forall s_i \in S \setminus \{s_0\} \tag{9}$$

$$f_{ij} \in \mathbb{Z}^+ \cup \{0\} \qquad \forall (s_i, s_j) \in E^d \tag{10}$$

$$y_i^r \in \{0, 1\} \qquad \forall s_i \in S \setminus \{s_0\}, r \in \mathcal{R} \tag{11}$$

For each $(s_i, s_j) \in E^d$, the f_{ij} variable represents the number of flow units transmitted on the arc. For each $s_i \in S \setminus \{s_0\}$ and $r \in \mathcal{R}$, the binary variable y_i^r is equal to 1 if s_i is allocated to role r in the new partition, and 0 otherwise.

Objective function (4) minimizes the reduced cost of the partition. Constraints (5) impose that each sensor is allocated to exactly a role, while Constraints (6) make sure that each target can be monitored by a sensor which is chosen to be a source. Constraints (7)-(8) are the flow conservation constraints. The base station produces a number of flow units which is equal to the number of sensors allocated to the source or relay role, while each of these sensors retain exactly one flow unit. Constraints (9) ensure that if positive flow is transmitted along the arc $(s_j, s_i) \in E^d$, then s_i cannot be allocated to role 0. By effect of

Constraints (7)-(9), the arcs with positive flow define a subtree composed of the base station and source or relay nodes. Along with the other constraints, they guarantee that it is a valid routing tree, and thus that a feasible partition is individuated.

In [10], the authors also proposed a network flow formulation for the pricing subproblem (called Model M2), which is however based on a directed graph whose set of nodes is $S \cup T$, with three types of arcs: (i) target-to-sensor; (ii) sensor-to-sensor; (iii) sensor-to-base station. In M2, a flow unit is produced by each target, and transmitted to its chosen source node using a type (i) arc. The flow constraints ensure that each flow unit is transmitted to the base station through a path of source or relay nodes, using arcs of type (ii)-(iii). It is easy to note that **[SP]** has fewer flow variables than M2, since the cardinality of E^d is equal to the number of arcs belonging to type (ii) or (iii). Furthermore, M2 uses a set of binary variables for roles allocation which is equal to ours. A detailed comparison among the two models is provided in Section 4.

3.3. Initializing ECG-MR

Our ECG-MR procedure needs an initial set of feasible partitions to solve the master problem in the first iteration. To obtain this set, we use the initialization procedure described in Section 3.1.5, using random values for the $\overline{dp}$ dual prices vector. All the partitions corresponding to the obtained chromosomes are added to the master problem.

4. Computational Results

In this section we evaluate the effectiveness and performance of the ECG-MR algorithm by comparing it with the CG+BBC and CG+CP algorithms proposed by [10] on the instances considered in that work. The two algorithms correspond to CG procedures using a branch-and-cut approach and a constraint programming approach to solve the subproblem, respectively. A third approach (CG+ILP) using the ILP formulation M2 was shown in [10] to be consistently

outperformed by CG+BBC and CG+CP, therefore we discarded it from the comparison with ECG-MR.

Our algorithm was coded in C++ on an OSX platform, running on an Intel Core2 Duo 3.4 GHz processor with 8 GB of RAM. The mathematical formulations embedded in the CG framework were solved using the Concert library of IBM ILOG CPLEX 12.6. The algorithm was executed in single thread mode, hence we did not take advantage of the parallelism offered by CPLEX. We fixed a maximum running time equal to 1 hour for the resolution of each instance. When a certified optimal solution is not found within this threshold, the best solution found is returned and used for the comparisons. After a tuning phase, we determined the values for the parameters used by ECG-MR in all computational tests. In more detail, the solution value threshold for the premature interruption of **[SP]** was set to 0.9, while the $PopSize$, $MaxIt$, $MaxDup$ and $InitMaxDup$ parameters of GA were set to 100, 1000, 100 and 100, respectively.

In [10], the tests were performed on a machine using an Intel Xeon Quad-Core W3550 3.06 GHz processor with 6 GB of RAM. As in our tests, a 1 hour time limit was considered, and the best results found within this time interval were reported. Furthermore, they allowed their procedures to run for additional 10 hours, in order to find as many certified optima as possible. In this way, they were able to report optimal solutions for all instances except 2. As will be discussed in this section, ECG-MR was instead able to report the whole set of optimal solutions, including the missing 2, within the 1 hour time limit.

The instances proposed in [10] have a number of sensors $|S \setminus \{s_0\}|$ varying in the set $\{100, 200, 300, 400, 500\}$ and a number of targets $|T|$ equal to either 15 or 30. Sensors and targets are disposed in a two-dimensional area with size 500×500. The communication range (R_c) of the sensors is equal to 125, while their sensing range (R_s) is equal to either 100 or 125. The consumption rates vector $\bar{l}$ is either $[l^0 = 0, l^1 = 0.8, l^2 = 1]$ or $[l^0 = 0, l^1 = 1, l^2 = 1]$. Each sensor is assumed to have an equal battery lifetime, normalized to 1 time unit. For each combination of parameters, 4 different instances were generated, for a total of 160 instances.

One of the main contributions of our work is the introduction of an alternative ILP formulation for the subproblem ([**SP**]). In order to verify the effectiveness of this formulation, before introducing the comparison between ECG-MR, CG+BBC and CG+CP, we compare [**SP**] with M2, the ILP subproblem formulation proposed in [10]. We implemented both formulations within a CG approach that is executed on the same machine, using the same master problem. For both these CG algorithms, the subproblem is always solved to optimality using the related ILP formulation. We call these two algorithms CG-SP and CG-M2, respectively. For each tested instance, both compared procedures were initialized with an equal set of randomly generated partitions. The results of this comparison, performed on the instances with up to 200 sensors, are reported in Table 1.

Each line in the table represents a scenario composed of 4 instances, and reports average values. The detailed results can be downloaded from the site of authors. The first two columns of the table report the number of sensors and targets in the scenarios. The next four columns show the network lifetime (in time units) and the computational time (in seconds) for the two algorithms. The last two columns report how many times the two algorithms found an optimal solution. We start the comparison on the scenarios with $R_s = 100$ and $l^1 = 1$.

On these scenarios, it can be noted that when CG-M2 reaches the time limit, the lifetimes computed are often far from the optimal values. In particular, for the scenarios with 200 sensors, CG-SP finds 3 out of 4 optimal solutions when $|T|$=15, and all of them when $|T|$=30, with average lifetimes equal to 10.16 and 8.75, respectively. On the other hand, CG-M2 finds no optimal solutions for these scenarios, and the returned lifetimes are equal to 6.61 when $|T|$=15 and 6.21 when $|T|$=30. The overall number of instances solved to optimality is remarkably different as well. Indeed, CG-SP finds the optimal solution for 14 out of 16 instances, while CG+M2 optimally solves 7 instances. Regarding the algorithms performance, CG-SP is always faster than CG-M2, and the time gap is often significant.

The results on the scenarios $l^1 = 0.8$ and $R_s = 100$ are slightly better for

		CG-M2		CG-SP		#Opt	
$\lvert S \setminus \{s_0\}\rvert$	$\lvert T\rvert$	LT	Time	LT	Time	CG-M2	CG-SP
$R_s = 100$, $l^1 = 1$							
100	15	4.00	42.63	4.00	1.79	4	4
100	30	3.97	943.88	3.99	904.13	3	3
200	15	6.61	3593.42	10.16	1050.73	0	3
200	30	6.21	3592.44	8.75	147.71	0	4
$R_s = 100$, $l^1 = 0.8$							
100	15	4.29	131.23	4.29	10.86	4	4
100	30	3.99	916.81	4.00	44.43	3	4
200	15	9.21	2859.29	10.58	1271.69	1	3
200	30	7.83	3482.92	8.85	654.72	1	4
$R_s = 125$, $l^1 = 1$							
100	15	4.63	1803.61	4.75	663.07	2	4
100	30	4.49	1808.76	4.73	1175.51	2	3
200	15	7.12	3591.53	12.39	1838.54	0	2
200	30	6.55	3592.86	10.93	951.49	0	3
$R_s = 125$, $l^1 = 0.8$							
100	15	5.09	1808.88	5.11	1142.11	2	3
100	30	4.87	1813.78	5.03	1804.80	2	2
200	15	9.53	3058.47	13.03	2714.96	1	1
200	30	7.56	3591.55	11.68	1837.54	0	2

Table 1: Comparison between CG-M2 and CG-SP

both the algorithms. Indeed, CG-SP optimally solves 15 out of 16 instances, while CG+M2 solves 9 of them. On the scenarios with 200 sensors, we can observe that the lifetimes gap is again significant (over 10%) between the two algorithms. Finally, a slight overall increase of the computational times for CG-SP can be noted. However, CG-SP remains always noticeably faster than CG-M2.

From these results, we can conclude that on these smaller instances the

consumption ratio value for the relay nodes does not seem to heavily impact the effectiveness of the two approaches.

By increasing the sensing range to $R_s = 125$, instead, we can observe higher increases in terms of computational time for the algorithms, and a larger decrease with respect to the number of optimally solved instances. Such a result was expected, since as the sensing range increases, the number of feasible partitions is likely to increase as well. In more detail, in the scenarios with $R_s = 125$ and $l^1 = 1$, the gap in terms of effectiveness increases with respect to the instances with $R_s = 100$ and the same l^1 value. Indeed, on the scenarios with 200 sensors, the solutions found by CG-SP are over 40% greater than the ones found by CG-M2. The computational time of CG-SP is often half the computational time of CG-M2. The overall number of optima found by CG-SP is 12, while CG-M2 finds 4 of them. Finally, in the scenarios with $R_s = 125$ and $l^1 = 0.8$, we observe for CG-SP an overall increase of the computational times and a reduction in the number of found optima, that is equal to 8. CG-M2 reports computational times that are similar to the previous case, and finds 5 optimal solutions. The instances with $l^1 = 0.8$ and $R_s = 125$ resulted to be overall the hardest to solve in these tests.

Summarizing, the results reported in Table 1 highlight that it is very important to use efficient methods to solve the subproblem, in particular when it has to be solved to optimality. In our comparison, formulation M2 resulted to be more computationally expensive to solve, slowing down the whole CG approach.

We now compare ECG-MR with the two best-performing approaches reported in [10], namely CG+BBC and CG+CP. Again, we report average values, while the detailed results can be found online on the authors' website.

Table 2 reports the results of the comparison for the instances with $R_s = 100$. As in the previous table, the columns $|S \setminus \{s_0\}|$ and $|T|$ report the number of sensors and targets, while the LT and *Time* columns report average network lifetimes in time units and average computational times in seconds, respectively. Since ECG-MR solves all instances within the time limit, its LT column also coincides with the optimal solution values. Finally, the *%Dev* columns report

$\lvert S \setminus \{s_0\}\rvert$	$\lvert T\rvert$	ECG-MR		CG+BBC			CG+CP		
		LT	Time	LT	Time	%Dev	LT	Time	%Dev
$R_s = 100$, $l^1 = 1$									
100	15	4.00	0.57	4.00	31.40	0.00	4.00	2.58	0.00
100	30	4.00	0.67	4.00	190.58	0.00	4.00	3.58	0.00
200	15	10.25	2.55	10.25	51.65	0.00	10.25	31.60	0.00
200	30	8.75	1.67	8.75	54.68	0.00	8.75	27.68	0.00
300	15	15.00	3.61	15.00	550.58	0.00	15.00	172.65	0.00
300	30	13.25	4.11	13.25	111.40	0.00	13.25	169.28	0.00
400	15	18.25	6.15	18.25	169.85	0.00	18.25	619.35	0.00
400	30	18.00	18.94	18.00	144.48	0.00	18.00	915.15	0.00
500	15	29.00	21.22	27.58	1158.90	4.91	29.00	2319.48	0.00
500	30	26.25	18.13	25.34	1800.50	3.47	26.25	2122.03	0.00
$R_s = 100$, $l^1 = 0.8$									
100	15	4.29	0.43	4.29	23.28	0.00	4.29	3.70	0.00
100	30	4.00	0.59	4.00	48.90	0.00	4.00	6.50	0.00
200	15	10.90	2.25	10.24	911.88	6.06	10.90	52.73	0.00
200	30	8.85	1.50	8.85	347.35	0.00	8.85	44.25	0.00
300	15	16.06	4.22	15.65	981.88	2.58	16.06	735.13	0.00
300	30	13.30	3.11	13.30	105.78	0.00	13.30	161.68	0.00
400	15	18.90	7.37	18.82	944.15	0.42	18.90	461.50	0.00
400	30	18.40	12.63	18.40	94.98	0.00	18.40	590.95	0.00
500	15	29.00	12.97	27.58	1158.75	4.91	29.00	1676.45	0.00
500	30	26.25	16.79	26.25	670.95	0.00	26.25	1752.05	0.00

Table 2: Computational results for ECG-MR, CG+BBC and CG+CP on the scenarios with $R_s = 100$.

how much CG+BBC and CG+CP deviate from the optimal solutions, in terms of percentage gap.

All the scenarios with $l^1 = 1$ and $R_s = 100$ are optimally solved by CG+CP as well, while CG+BBC did not find the optimal solutions for the two scenarios with 500 sensors. The gaps from optimal solutions in these cases are always lower than 5%. What mostly distinguishes ECG-MR from the other two algorithms on these instances are the performances. All the scenarios are, indeed, optimally solved by ECG-MR in less than 22 seconds, while CG+CP requires

$\|S \setminus \{s_0\}\|$	$\|T\|$	ECG-MR		CG+BBC			CG+CP		
		LT	Time	LT	Time	%Dev	LT	Time	%Dev
$R_s = 125$, $l^1 = 1$									
100	15	4.75	0.50	4.75	96.53	0.00	4.75	3.83	0.00
100	30	4.75	0.73	4.59	952.18	3.32	4.75	33.65	0.00
200	15	13.00	2.46	12.76	1519.78	1.87	13.00	77.30	0.00
200	30	11.75	2.76	11.18	1804.08	4.85	11.68	922.53	0.60
300	15	16.75	5.10	15.41	2749.33	8.01	16.75	189.18	0.00
300	30	16.00	5.33	14.95	1826.63	6.55	16.00	237.65	0.00
400	15	24.88	10.83	22.69	2704.18	8.77	24.88	878.65	0.00
400	30	22.38	30.63	21.74	1833.40	2.86	22.38	906.43	0.00
500	15	37.75	42.19	30.81	3600.00	18.38	35.14	2839.95	6.92
500	30	35.50	70.27	31.68	3600.00	10.75	33.35	3023.25	6.05
$R_s = 125$, $l^1 = 0.8$									
100	15	5.28	0.52	5.12	1354.38	2.98	5.28	4.45	0.00
100	30	5.24	0.56	5.17	1809.83	1.34	5.24	6.50	0.00
200	15	14.35	2.16	13.73	984.08	4.34	14.35	77.03	0.00
200	30	12.60	1.88	12.23	1833.28	2.92	12.60	70.75	0.00
300	15	18.46	4.50	17.42	2735.93	5.65	18.46	179.53	0.00
300	30	16.65	4.23	15.72	2289.53	5.57	16.65	179.55	0.00
400	15	26.25	8.84	24.71	2472.95	5.87	26.25	688.45	0.00
400	30	23.55	15.01	21.76	3600.00	7.62	23.55	709.00	0.00
500	15	38.45	36.07	33.29	3182.55	13.41	37.18	2647.23	3.32
500	30	36.15	56.58	31.79	3600.00	12.06	34.93	3004.83	3.37

Table 3: Computational results of ECG-MR, CG+BBC and CG+CP on the scenarios with $R_s = 125$.

up to about 2320 seconds, and CG+BBC, as discussed, reaches the time limit in some cases. By decreasing the consumption rate l^1 to 0.8, we can observe that ECG-MR and CG+CP require less computational time to optimally solve all the scenarios. However, there is again an impressive performance gap among the two algorithms, since they require about 17 and 1750 seconds in the worst case, respectively. On the contrary, CG+BBC has worse performances on these scenarios, since it does not solve to optimality 4 of them.

Finally, in Table 3 the results of the three algorithms on the scenarios with

$R_s = 125$ are reported. As already observed discussing the results in Table 1, increasing the sensing ranges makes the instances harder to solve. Indeed, the computational times of all the algorithms increase for these scenarios, although ECG-MR is still able to solve to optimality all of them. Let us first consider the case $l^s = 1$. On these scenarios, the computational time for ECG-MR grows up to about 70 seconds in the worst case. As mentioned earlier in this section, 2 instances (which belong to these scenarios) were not solved to optimality in [10], even considering additional 10 hours of computational time; these two instances were solved in less than 3 minutes each by ECG-MR.

On the other hand, CG+BBC optimally solves only the first scenario, with a solution gap in the other cases that grows up to 18.38%. CG+CP also gets less effective, with 3 scenarios that are not solved to optimality, and a solution gap that grows up to about 7%.

The scenarios corresponding to $l^1 = 0.8$ appear to be easier to solve for ECG-MR and CG+CP. In particular, CG+CP does not solve to optimality only the 2 scenarios corresponding to 500 sensors, with smaller solution gaps than in the previous case, up to 3.37 %. The performance gap with ECG-MR continues to be remarkable, since the latter algorithm solves each scenario within 57 seconds in the worst case. No scenario of this group is solved to optimality by CG+BBC.

The computational results show ECG-MR to be highly effective with respect to the previous algorithms proposed in the literature. In addition to the reformulated ILP for the pricing subproblem, we mostly ascribe these results to the application of an appropriately designed GA to solve its heuristically, and to the the use of a callback function to prematurely interrupt the ILP resolution. Indeed, our GA was able to quickly return multiple useful columns at once, which in our opinion results in a more effective strategy than spending a considerable amount of time to find the optimal solution at each subproblem invocation. Furthermore, whenever the GA fails, the premature subproblem interruption allows us to introduce a new partition that may help the GA to regain effectiveness in the next iteration, again requiring a computational time that can be significantly inferior to solving the subproblem to optimality.

5. Conclusions

In this work, we addressed the Maximum Lifetime Problem with Role Allocation and Connectivity Constraints, a problem defined in sensor networks in which the aim is to appropriately schedule activation times and assign roles (source, relay or idle) to sensors, in order to prolong the amount of time over which the network can monitor a given set of target locations and route the information to a central base station. We developed an exact approach based on the column generation framework, which includes a genetic metaheuristic for the pricing subproblem resolution. The proposed ECG-MR approach was shown experimentally to overcome previous algorithms proposed in the literature, also providing exact solutions for benchmark instances that were unsolved to date.

Regarding our future research efforts, we intend to study the effect of partial coverage constraints, that is, the case in which a percentage of targets may be left uncovered at any given time. These types of constraints have proven to remarkably complicate lifetime problem variants in the literature, hence we believe that they represent an interesting line of research.

References

[1] A. Alfieri, A. Bianco, P. Brandimarte, C. F. Chiasserini, Maximizing system lifetime in wireless sensor networks, European Journal of Operational Research 181 (1) (2007) 390–402.

[2] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Computer Networks 54 (15) (2010) 2787–2805.

[3] W. Awada, M. Cardei, Energy-efficient data gathering in heterogeneous wireless sensor networks, in: Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2006, pp. 53–60.

[4] P. Berman, G. Calinescu, C. Shah, A. Zelikovsky, Power efficient monitoring

management in sensor networks, in: Proceedings of the Wireless Communications and Networking Conference, vol. 4, 2004, pp. 2329 – 2334.

[5] M. Cardei, M. T. Thai, Y. Li, W. Wu, Energy-efficient target coverage in wireless sensor networks, in: Proceedings of the 24th conference of the IEEE Communications Society, vol. 3, 2005, pp. 1976–1984.

[6] M. Cardei, J. Wu, M. Lu, Improving network lifetime using sensors with adjustable sensing ranges, International Journal of Sensor Networks 1 (1-2) (2006) 41–49.

[7] F. Carrabs, R. Cerulli, C. D'Ambrosio, M. Gentili, A. Raiconi, Maximizing lifetime in wireless sensor networks with multiple sensor families, Computers & Operations Research 60 (2015) 121–137.

[8] F. Carrabs, R. Cerulli, C. D'Ambrosio, A. Raiconi, Exact and heuristic approaches for the maximum lifetime problem in sensor networks with coverage and connectivity constraints, Tech. rep., Department of Mathematics, University of Salerno. (2015).

[9] F. Carrabs, R. Cerulli, C. D'Ambrosio, A. Raiconi, A hybrid exact approach for maximizing lifetime in sensor networks with complete and partial coverage constraints, Journal of Network and Computer Applications 58 (2015) 12–22.

[10] F. Castaño, E. Bourreau, N. Velasco, A. Rossi, M. Sevaux, Exact approaches for lifetime maximization in connectivity constrained wireless multi-role sensor networks, European Journal of Operational Research 241 (1) (2015) 28–38.

[11] F. Castaño, A. Rossi, M. Sevaux, N. Velasco, A column generation approach to extend lifetime in wireless sensor networks with coverage and connectivity constraints, Computers & Operations Research 52 (B) (2014) 220–230.

[12] R. Cerulli, R. De Donato, A. Raiconi, Exact and heuristic methods to maximize network lifetime in wireless sensor networks with adjustable sensing ranges, European Journal of Operational Research 220 (1) (2012) 58–66.

[13] R. Cerulli, M. Gentili, A. Raiconi, Maximizing lifetime and handling reliability in wireless sensor networks, Networks 64 (4) (2014) 321–338.

[14] K. Deschinkel, A column generation based heuristic for maximum lifetime coverage in wireless sensor networks, in: SENSORCOMM 11, 5th Int. Conf. on Sensor Technologies and Applications, vol. 4, 2011, pp. 209 – 214.

[15] A. Dhawan, C. T. Vu, A. Zelikovsky, Y. Li, S. K. Prasad, Maximum lifetime of sensor networks with adjustable sensing range, in: Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2006, pp. 285 – 289.

[16] C. Duin, S. Voss, Steiner tree heuristics - a survey, in: H. Dyckhoff, U. Derigs, M. Salomon, H. C. Tijms (eds.), Operations Research Proceedings 1993. Papers of the 22nd Annual Meeting of DGOR in Cooperation with NSOR, Springer-Verlag, 1994, pp. 485–496.

[17] M. Gentili, A. Raiconi, $\alpha-$coverage to extend network lifetime on wireless sensor networks, Optimization Letters 7 (1) (2013) 157–172.

[18] Y. Gu, Y. Ji, B. Zhao, Maximize lifetime of heterogeneous wireless sensor networks with joint coverage and connectivity requirement, in: EmbeddedCom-09, 8th International Conference on Embedded Computing, 2009, pp. 226–231.

[19] C. H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[20] A. Raiconi, M. Gentili, Exact and metaheuristic approaches to extend lifetime and maintain connectivity in wireless sensors networks, in: J. Pahl,

T. Reiners, S. Voss (eds.), Network Optimization, vol. 6701 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2011, pp. 607–619.

[21] A. Rossi, A. Singh, M. Sevaux, An exact approach for maximizing the lifetime of sensor networks with adjustable sensing ranges, Computers & Operations Research 39 (12) (2012) 3166–3176.

[22] C. Wang, M. T. Thai, Y. Li, F. Wang, W. Wu, Minimum coverage breach and maximum network lifetime in wireless sensor networks, in: Proceedings of the IEEE Global Telecommunications Conference, 2007, pp. 1118–1123.