Computers & Operations Research 39 (2012) 3166–3176

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

An exact approach for maximizing the lifetime of sensor networks with adjustable sensing ranges

André Rossi [a,*], Alok Singh [b], Marc Sevaux [a]

[a] Lab-STICC, Université de Bretagne Sud – F-56321 Lorient, France
[b] Department of Computer and Information Sciences, University of Hyderabad, Hyderabad 500 046, Andhra Pradesh, India

ARTICLE INFO

Available online 13 April 2012

Keywords:
Wireless sensor networks
Network lifetime
Column generation
Genetic algorithm

ABSTRACT

This paper addresses the problem of target coverage for wireless sensor networks, where the sensing range of sensors can vary, thereby saving energy when only close targets need to be monitored. Two versions of this problem are addressed. In the first version, sensing ranges are supposed to be continuously adjustable (up to the maximum sensing range). In the second version, sensing ranges have to be chosen among a set of predefined values common to all sensors. An exact approach based on a column generation algorithm is proposed for solving these problems. The use of a genetic algorithm within the column generation scheme significantly decreases computation time, which results in an efficient exact approach.

1. Introduction

Advances in signal processing and embedded systems are at the origin of the growing popularity of Wireless Sensor Networks (WSN) in a wide range of applications [1]. While they were initially used in remote or hostile environments (for battlefield surveillance, or tsunami monitoring), WSN are also increasingly used for health care [12]. Although these applications rely on very different types of sensors, most of them share the following characteristics: sensors operate on a battery that cannot be recharged and a large number of sensors is deployed for improving fault tolerance and lifetime. This paper is concerned with lifetime maximization, that is achieved by making the best possible use of sensors redundancy. Moreover, the sensors are supposed to have adjustable sensing ranges. Such a feature saves energy in the situation where a sensor needs to cover close targets only, as power requirement is a non-decreasing function of the distance between the sensor and the farthest target it covers.

More formally, suppose n sensors are randomly deployed in order to cover a set of m targets $\{\tau_1,\ldots,\tau_k,\ldots,\tau_m\}$. Each sensor s_i has an initial energy b_i. Sensors can either be active or inactive. An inactive sensor does not cover any target, and its power consumption is negligible. When a sensor is active, its power consumption depends on its sensing range. All the sensors have the same maximum sensing range, denoted by R^{max}: a sensor can cover a target if its distance is less than or equal to R^{max}. Lifetime maximization is reached by gathering sensors into non-disjoint subsets called *covers* (each cover being such that each target can be covered by at least one sensor in that cover), and by scheduling these covers, *i.e.*, by determining the amount of time during which each cover is used. The sensors that are not part of the cover that is currently being used are not active. Moreover, covers are not necessarily disjoint, as this allows for reaching longer lifetimes [15]. The schedule must be such that the total amount of energy consumed by sensor s_i is at most equal to its initial energy b_i. The network lifetime is the sum of these durations: when it is exceeded, the coverage of all targets is no longer possible.

This paper addresses two close versions of the lifetime maximization problem:

- Lifetime Maximization with Ad-hoc Sensing Ranges (LM-ASR): Each sensor can adjust its sensing range so as to cover targets with the minimum amount of necessary power (for all targets which distance to the sensor is less than R^{max}). In this model, continuous variations of the sensing range are allowed, this problem version is addressed in [6],
- Lifetime Maximization with Predefined Sensing Ranges (LM-PSR): Sensors have M_{PSR} nonzero and distinct predefined sensing ranges, so all the targets that are under such a predefined range are covered at a predefined power level. In this model, M_{PSR} predefined sensing ranges are supposed to be given, this problem version is addressed in [3].

A mathematical formulation based on integer or linear programming is proposed in [3,6], but is never used for solving the

* Corresponding author. Tel.: +33 297874564; fax: + 33 297874527.
E-mail addresses: andre.rossi@univ-ubs.fr (A. Rossi), marc.sevaux@univ-ubs.fr (M. Sevaux).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.04.001

problem, as both formulations rely on an exponential number of variables. In [17,18,16,11], heuristic methods are proposed for finding the best possible cover for LM-PSR. More precisely, [11] uses a NSGA-II approach, where a trade off is sought between the coverage rate of the sensors in the cover (breach is allowed), the financial cost of the cover (which is proportional to the number of sensors in the cover) and the total power of the sensors used in the cover. These approaches, however, are concerned with the generation of a single cover, and the problem of maximizing lifetime is not addressed.

The present paper generalizes both problems, and provides exact approaches hybridized with metaheuristics for both of them. These approaches are based on column generation, which has been successfully used to address lifetime maximization problems in the literature [2,8,9,14].

The remainder of this paper is organized as follows. Section 2 describes in detail the terminology and notations used in this paper and illustrates them, wherever appropriate, with suitable examples. Section 3 describes the problem models and their resolution approaches. Computational results along with their analysis are presented in Section 4. Finally, Section 5 outlines some concluding remarks and ideas for future works.

2. Definitions and notations

All the notations introduced in this section can be found in Table 1.

2.1. Power consumption and sensor coverage

Dealing with adjustable sensing ranges requires to extend the notion of covers as follows. A cover S_j is a n-column vector of nonnegative reals where $S_{i,j} > 0$ if and only if sensor s_i is part of the cover. $S_{i,j}$ is the power (*i.e.*, the energy consumption rate) of sensor s_i in the cover. If sensor s_i is not part of S_j, then $S_{i,j} = 0$ as this sensor does not consume energy when S_j is used. The amount of energy consumed by active cover S_j is proportional to t_j, where t_j is the amount of time during which cover S_j is used.

The battery of sensor s_i has initial energy b_i and the power consumption of s_i is denoted by $p_i(t)$, as it can vary over time if the sensing range of s_i does. Consequently, the constraint on the limited amount of energy available for sensor s_i can be stated as

$$\int_0^{+\infty} p_i(t)\,dt \le b_i \quad \forall i \in \{1,\ldots,n\} \tag{1}$$

It is assumed that $p_i(t)$ is a function of the sensing range $r(t)$: $p_i(t) = f(r(t))$. As in [3,6], two power consumption models are considered. The first one is referred to as the linear model, the second one is the quadratic model.

$$f_{lin}(r) = p^{max}\left(\frac{r}{R^{max}}\right), \quad 0 \le r \le R^{max}$$

$$f_{qua}(r) = p^{max}\left(\frac{r}{R^{max}}\right)^2, \quad 0 \le r \le R^{max}$$

In both cases, $p_i(t)$ is equal to p^{max} when the sensing range is equal to R^{max} (its maximum value). Since the models proposed in this paper do not depend on the form of the power consumption function, it is denoted by $f(r)$, and is either $f_{lin}(r)$ or $f_{qua}(r)$.

In both LM-ASR and LM-PSR, $r(t)$ can only assume a finite set of numerical values. This is obvious for LM-PSR, and in LM-ASR, $r(t)$ assumes at most as many different values as there are targets under a sensor range. Thus, it can be deduced that $p_i(t)$ is a step function of time. Indeed, the step values are either set by the distance to neighboring targets in LM-ASR, or are taken in a predefined collection of values in LM-PSR. Eq. (1) can then be written as

$$\sum_{j=1}^{c} S_{i,j} t_j \le b_i, \quad \forall i \in \{1,\ldots,n\} \tag{2}$$

where c is the number of covers, and $S_{i,j}$ is the power consumption of sensor s_i associated with the sensing range selected for that cover.

Every sensor s_i is associated with an ordered set D_i containing the targets that it can cover, sorted by increasing power requirement. In an ideal environment, the targets in D_i are those with distance to sensor s_i less than or equal to R^{max}. We assume such an environment, even though this hypothesis is not necessary for the solution approaches proposed in this paper to be valid. The presence of obstacles in the environment, for example, may cause the sensors power consumption not to be a function of $r(t)$ alone as in f. Such a situation can be handled with no inconvenience using one of the two following approaches. If a more realistic function for computing power requirement is available, then it is used instead of f_{lin} of f_{qua}. Otherwise, the network can go through an initialization phase during which the sensing range increases progressively (in LM-ASR), or takes its predefined values sequentially (in LM-PSR), and the targets discovered during that phase are stored along with the corresponding power they require. However in that case, the problem name should refer to adjustable power levels, rather than adjustable sensing ranges.

Whatever the method for computing or measuring power consumption, the targets in D_i are sorted by increasing power requirement, *i.e.*, $D_i(1)$ is the target that can be covered by sensor

Table 1
Notations.

Notation	Meaning
n	Number of sensors
m	Number of targets
s_i	A sensor, for all $i \in \{1,\ldots,n\}$
τ_k	A target, for all $k \in \{1,\ldots,m\}$
b_i	Initial energy of sensor s_i for all $i \in \{1,\ldots,n\}$
$p_i(t)$	Power consumption of sensor s_i over time
D_i	Targets covered by sensor s_i, sorted by increasing power requirement
$p_{i,r}$	Minimum power consumption of sensor s_i required for covering target $D_i(r)$, for all $r \in \{1,\ldots,\lvert D_i\rvert\}$
R^{max}	Sensors' maximum sensing range
M	Maximum number of different power consumption values that a sensor can have
R	Set of the predefined sensing ranges sorted by increasing order (for LM-PSR only)
$C_{k,i}$	Power consumption minimum rank at which sensor s_i can cover target τ_k
S_j	jth cover (*i.e.*, set of sensors). $S_{i,j}$ is the power of sensor s_i in the cover, $\forall j \in \{1,\ldots,c\}$
t_j	Amount of time during which cover S_j is used
$x_{i,r,j}$	Binary decision variable that is set to one iff sensor s_i is part of cover S_j and is used with power $p_{i,r}$

s_i with minimum power consumption, whereas $D_i(|D_i|)$ is the target that can be covered by sensor s_i with maximum power consumption. $p_{i,r}$ is defined as the minimum power consumption of sensor s_i required for covering targets $D_i(r)$, for all r in $\{1,\ldots,|D_i|\}$. More formally, $p_{i,r-1} \le p_{i,r}$ for all i in $\{1,\ldots,n\}$, and for all r in $\{2,\ldots,|D_i|\}$. The maximum number of different power consumption values that a sensor can have is denoted by M. In LM-ASR, $M = \max_{i\in\{1,\ldots,n\}}|D_i|$. In LM-PSR, $M = M_{PSR}$. In the case of LM-PSR, R is the set of the predefined sensing ranges sorted by increasing order, with $R_{M_{PSR}} = R^{max}$ and $|R| = M_{PSR}$.

For all $k \in \{1,\ldots,m\}$, $C_{k,i}$ is defined as the power consumption minimum rank at which sensor s_i can cover target τ_k, if the distance between s_i and τ_k is less than or equal to R^{max}. If the distance is greater than R^{max}, then $C_{k,i}$ is set to $M+1$. For example, $C_{k,i} = z$ with $z \in \{1,\ldots,M\}$ means that target τ_k can be covered by sensor s_i, provided that its power is at least $p_{i,z}$. If $z = M+1$, then target τ_k is out of range of s_i.

2.1.1. Example

For the sake of illustration, the ordered sets D, C and p are computed for the two problem versions LM-ASR and LM-PSR, with the one-dimensional example shown in Fig. 1. The horizontal axis allows the reader to assess the distances more easily.

There are $n=2$ sensors, $m=3$ targets, the maximum sensing range R^{max} is set to 4 and sensors power requirements are given by $f_{qua}(r)$ with $p^{max} = 1$. It is also assumed that $b_1 = b_2 = 1$. This implies that each sensor can cover all the targets which are at distance less than 4 from it, for 1 s before its battery is depleted. Naturally, the lifetime can be extended further if sensors reduce their sensing range.

The ordered sets D are common to both LM-ASR and LM-PSR, they are defined as

$$D_1 = (1,2), \quad D_2 = (2,3,1)$$

Indeed, it can be seen from Fig. 1 that sensor s_1 can cover targets τ_1 and τ_2, and τ_1 is closer to s_1 than τ_2. Sensor s_2 can cover all the three targets, namely τ_2, τ_3 and τ_1 when sorted by non decreasing distance to s_2. Equivalently, we may have $D_2 = (3,2,1)$.

Sets C and p are different according to the problem version:

- LM-ASR

  The ordered sets defining the power consumption of sensors are defined below:

  $$p_1 = (\tfrac{1}{4},1), \quad p_2 = (\tfrac{1}{16},\tfrac{1}{16},\tfrac{9}{16})$$

  Sensor s_1 is at distance 2 from the closest target it covers, hence $p_{1,1} = (2/R^{max})^2 = 0.25$. The second target in D_1 is at distance R^{max} from s_1, hence $p_{1,2}$ reaches its maximum value (*i.e.*, 1).
  Sensor s_2 is at distance 1 from the closest target it covers, hence $p_{2,1} = (1/R^{max})^2 = \frac{1}{16}$. The second target in D_2 is also at distance 1 from s_2, hence $p_{2,2} = \frac{1}{16}$, and target τ_1 is at distance 3 so $p_{2,3} = \frac{9}{16}$.
  The maximum cardinality of D_i over i in $\{1,\ldots,n\}$ defines the maximum number of different sensing ranges that a sensor can have. Here, s_1 has two sensing ranges, and s_2 has three, so $M=3$.

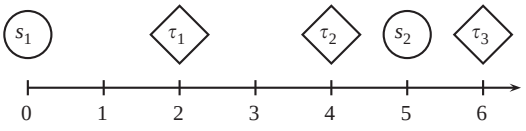

Fig. 1. A one-dimensional example.

  The ordered sets C_k that indicate the rank in power consumption of the sensors that cover target τ_k are

  $$C_1 = (1,3), \quad C_2 = (2,1), \quad C_3 = (4,2)$$

  Target τ_1 is covered by s_1, which can cover it using its first (lower) sensing range. τ_1 is also covered by s_2, but it requires its third sensing range to reach that target.
  Target τ_2 is covered by s_1, which can cover it using its second (highest) sensing range; it is also covered by s_2 but only requires its lower sensing range to reach it.
  Finally, target τ_3 is not covered by s_1, hence $C_{3,1}$ is set to $M+1=4$. Target τ_3 is covered by sensor s_2, and requires its second sensing range (or its first sensing range, as they are equal).

- LM-PSR

  It is supposed that $M_{PSR} = 2$, and the two predefined sensing ranges are $R = \{2,4\}$. By definition of $f_{qua}(r)$, the corresponding power consumption levels are $\frac{1}{4}$ and 1, respectively.
  The ordered sets defining the power consumption of sensors are

  $$p_1 = (\tfrac{1}{4},1), \quad p_2 = (\tfrac{1}{4},\tfrac{1}{4},1)$$

  Sensor s_1 is at distance 2 from the closest target it covers, hence the first predefined sensing range is needed so $p_{1,1} = (\frac{2}{4})^2 = 0.25$. The second target in D_1 is at distance R^{max} from s_1, hence $p_{1,2}$ reaches its maximum value (*i.e.*, 1).
  Sensor s_2 is at distance 1 from the closest target it covers, so $p_{2,1} = \frac{1}{4}$. The second target in D_2 is at distance 2 from s_2, hence $p_{2,2} = \frac{1}{4}$: both targets require the first predefined sensing range to be used. Target τ_1 is at distance 3 so the second predefined sensing range is used, leading to $p_{2,3} = 1$.
  The maximum number of different sensing ranges is equal to the number of predefined sensing ranges so $M=2$.
  The ordered sets C_k that indicate the rank in power consumption of the sensors that cover target τ_k are

  $$C_1 = (1,2), \quad C_2 = (2,1), \quad C_3 = (3,1)$$

  Target τ_1 is covered by s_1, which can cover it using the first predefined sensing range. τ_1 is also covered by s_2, but it requires the second predefined sensing range to reach that target.
  Target τ_2 is covered by s_1, which can cover it using the second sensing range; it is also covered by s_2 which can cover it using the first sensing range to reach it.
  Finally, target τ_3 is not covered by s_1, hence $C_{3,1}$ is set to $M+1=3$. Target τ_3 is covered by sensor s_2, and requires the first predefined sensing range.

2.2. Cover dominance properties

This section introduces two equivalent ways of representing covers. Those two different encodings are necessary as they are both used in the column generation approach. Then, theoretical properties are provided for characterizing non-dominated covers that are useful for maximizing lifetime.

2.2.1. Cover encodings

Real vector encoding: Cover S_j can be defined as a n-column real vector where $S_{i,j}$ is the power consumption of sensor s_i, *i.e.*, for all $i \in \{1,\ldots,n\}$, $S_{i,j}$ belongs to the discrete set $\{0,p_{i,1},p_{i,2},\ldots,p_{i,|D_i|}\}$ for LM-ASR, and $S_{i,j}$ belongs to the discrete set $\{0,p_{i,1},p_{i,2},\ldots,p_{i,M_{PSR}}\}$ for LM-PSR. For both problems, $S_{i,j} = 0$ if and only if s_i is not part of cover S_j.

Binary encoding: Equivalently, S_j can be represented as a set of n ordered lists of $|D_i|$ binary numbers for LM-ASR, and M_{PSR} binary numbers for LM-PSR. More precisely, $x_{i,r,j}$ is set to one if sensor s_i is part of the cover and is used with power $p_{i,r}$, for all $i \in \{1,\ldots,n\}$, it is zero otherwise. In addition, at most one variable $x_{i,r,j}$ should be set to one for all $i \in \{1,\ldots,n\}$ as each sensor has at most one power consumption level (and no level at all if it is not part of the cover).

Changing encoding: The following equation is used for transforming a cover from the binary encoding to the real vector encoding in LM-ASR:

$$S_{i,j} = \sum_{r=1}^{|D_i|} x_{i,r,j} p_{i,r} \quad \forall i \in \{1,\ldots,n\}, \quad \forall j \in \{1,\ldots,c\} \tag{3}$$

For LM-PSR, $|D_i|$ and $p_{i,r}$ must be replaced with M_{PSR} and $f(R_r)$, respectively.

The reciprocal transformation is given in Algorithm 1 for LM-ASR.

Algorithm 1. From the real vector encoding to the binary encoding.

```
for i = 1 to n do
    for r = 1 to |D_i| do
        if S_{i,j} = p_{i,r} then
            x_{i,r,j} ← 1;
        else
            x_{i,r,j} ← 0;
```

Algorithm 1 can be used for LM-PSR, by replacing $|D_i|$ and $p_{i,r}$ with M_{PSR} and $f(R_r)$, respectively.

2.2.2. Non-dominated covers

All the approaches that use column generation for addressing lifetime maximization in wireless sensor networks rely on the notion of cover [2,9,14]. In all these column generation approaches, the main issue is to find a cover that is likely to increase further the lifetime of the current solution. To the best of our knowledge, this article provides the first theoretical study on how to reduce the search space of such attractive covers. As a result, it is shown that the search space can be reduced to non-dominated covers. This provides a significant advantage over performing a search in the much larger set of valid covers.

A cover is said to be *valid* if the power level of its sensors is such that all the targets are covered by at least one sensor. A valid cover S_j is said to be *dominated* if there exists another valid cover $S_{j'}$ containing a subset of sensors of S_j that are used at a lower power level, and at least one sensor is used in $S_{j'}$ at a strictly lower power than in S_j. More formally, S_j is dominated if there exists another valid cover $S_{j'}$ such that $S_{i,j'} \leq S_{i,j}$ for all $i \in \{1,\ldots,n\}$, and there exists at least one integer i_0 in $\{1,\ldots,n\}$ such that $S_{i_0,j'} < S_{i_0,j}$.

Therefore, in any non-dominated cover, decreasing the power level of any sensor compromises coverage. Consequently, a non-dominated cover can be built from any valid cover by decreasing the power of sensors as long as it is possible to do so without compromising targets coverage. This process is referred to as cover refinement and is implemented in Algorithm 2.

Lemma 1. *There exists an optimal solution to LM-ASR (or LM-PSR) in which non-dominated covers only are used a non-zero amount of time.*

Proof. Lemma 1 is proved by contradiction. Suppose that all optimal solutions to LM-ASR (or LM-PSR) are such that there exists a dominated cover S_j used for a nonzero amount of time. Then a non-dominated cover $S_{j'}$ can be built from S_j, and be used instead of S_j in the optimal solution. Doing so preserves target coverage by definition of non-dominated covers. Repeating this process for all dominated covers used a nonzero amount of time leads to an optimal solution to LM-ASR (or LM-PSR) where non-dominated covers only are used a non-zero amount of time. Contradiction.

As a consequence of Lemma 1, the search for an optimal solution to LM-ASR (or LM-PSR) can be restricted to non-dominated covers only. Enforcing this restriction is efficient when designing a solution approach to LM-ASR or LM-PSR, as non-dominated covers are generally a small fraction of valid covers.

Lemma 2. *The number of sensors in a non-dominated cover is at most m.*

Proof. Each target has to be covered by at least one sensor, so at most m sensors are required for covering all the targets. If a valid cover has more than m sensors, then one of them can be removed, so it is dominated.

Lemma 2 can be seen as a necessary (but not sufficient) condition for a cover to be non-dominated. It is used in the column generation algorithm for building candidate non-dominated covers in Section 3.3.

2.2.3. Example

The Example introduced in Section 2.1.1 is used again for illustrating both cover encodings, valid covers, dominated and non-dominated covers, for LM-ASR and LM-PSR.

- LM-ASR
  There exist 5 valid covers, represented under the real vector encoding:

$$S_1^{ASR} = \begin{bmatrix} \frac{1}{4} \\ \frac{1}{16} \end{bmatrix}, S_2^{ASR} = \begin{bmatrix} 0 \\ \frac{9}{16} \end{bmatrix}, S_3^{ASR} = \begin{bmatrix} 1 \\ \frac{1}{16} \end{bmatrix}, S_4^{ASR} = \begin{bmatrix} 1 \\ \frac{9}{16} \end{bmatrix}, S_5^{ASR} = \begin{bmatrix} \frac{1}{4} \\ \frac{9}{16} \end{bmatrix}$$

  Covers S_3^{ASR}, S_4^{ASR} and S_5^{ASR} are dominated because of cover S_1^{ASR}. Hence, S_1^{ASR} and S_2^{ASR} are non-dominated.
  The binary encoding of covers S_1^{ASR} and S_2^{ASR} are X_1^{ASR} and X_2^{ASR}, respectively:

$$X_1^{ASR} = \begin{bmatrix} 1 & 0 & - \\ 1 & 0 & 0 \end{bmatrix}, X_2^{ASR} = \begin{bmatrix} 0 & 0 & - \\ 0 & 0 & 1 \end{bmatrix}$$

  The – sign on the first row indicates that s_1 has only two power consumption levels ($|D_1| = 2$). In S_1^{ASR}, both sensors are part of the cover, and are used with their minimum power. In S_2^{ASR}, only s_2 is part of the cover, and is used at its maximum power.
- LM-PSR
  There are also five valid covers for LM-PSR, they are shown below under the real vector encoding:

$$S_1^{PSR} = \begin{bmatrix} \frac{1}{4} \\ \frac{1}{4} \end{bmatrix}, S_2^{PSR} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, S_3^{PSR} = \begin{bmatrix} 1 \\ \frac{1}{4} \end{bmatrix}, S_4^{PSR} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, S_5^{PSR} = \begin{bmatrix} \frac{1}{4} \\ 1 \end{bmatrix}$$

  Again, S_3^{PSR}, S_4^{PSR}, and S_5^{PSR} are dominated. The binary encoding for S_1^{PSR} and S_2^{PSR} are X_1^{PSR} and X_2^{PSR}, respectively:

$$X_1^{PSR} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, X_2^{PSR} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

  Unlike LM-ASR, all the rows in the binary encoding of a cover have the same size in LM-PSR because each sensor has exactly M_{PSR} power consumption levels.

3. Problem modeling

3.1. A mixed integer linear programming model

LM-ASR can be modeled as the following mathematical formulation:

$$\text{Max} \quad \sum_{j=1}^{c} t_j \tag{4}$$

$$\text{s.t.} \quad \sum_{j=1}^{c} \left(\sum_{r=1}^{|D_i|} x_{i,r,j} p_{i,r} \right) t_j \le b_i, \quad \forall i \in \{1,\ldots,n\} \tag{5}$$

$$\sum_{r=1}^{|D_i|} x_{i,r,j} \le 1 \quad \forall i \in \{1,\ldots,n\}, \forall j \in \{1,\ldots,c\} \tag{6}$$

$$\sum_{i=1}^{n} \sum_{r=C_{k,i}}^{|D_i|} x_{i,r,j} \ge 1 \quad \forall k \in \{1,\ldots,m\}, \ \forall j \in \{1,\ldots,c\} \tag{7}$$

$$\sum_{i=1}^{n} \sum_{r=1}^{|D_i|} x_{i,r,j} \le m \quad \forall j \in \{1,\ldots,c\} \tag{8}$$

$$x_{i,r,j} \in \{0,1\} \quad \forall i \in \{1,\ldots,n\}, \ \forall r \in \{1,\ldots,|D_i|\}, \ \forall j \in \{1,\ldots,c\} \tag{9}$$

$$t_j \ge 0 \quad \forall j \in \{1,\ldots,c\} \tag{10}$$

The objective function is to maximize the network lifetime. Eq. (5), which is non-linear, ensures that energy limitations are respected for each sensor. It is a combination of Eqs. (2) and (3). Eq. (6) states that in each cover, each sensor is used with at most one power level. The target coverage requirement are enforced by Eq. (7), Eq. and (8) restrict the search for covers with at most m sensors. This model can be used for LM-PSR by replacing $|D_i|$ and $p_{i,r}$ with M_{PSR} and $f(R_r)$, respectively.

This mathematical model is linearized by introducing continuous variables $z_{i,r,j}$ for replacing $x_{i,r,j}p_{i,r}t_j$ in Eq. (5). These new variables are linked with $x_{i,r,j}$ and t_j as follows:

$$z_{i,r,j} \le p_{i,r} t_j \quad \forall i \in \{1,\ldots,n\}, \ \forall r \in \{1,\ldots,|D_i|\}, \ \forall j \in \{1,\ldots,c\} \tag{11}$$

$$z_{i,r,j} \le b_i x_{i,r,j} \quad \forall i \in \{1,\ldots,n\}, \ \forall r \in \{1,\ldots,|D_i|\}, \ \forall j \in \{1,\ldots,c\} \tag{12}$$

$$z_{i,r,j} \ge p_{i,r} t_j + b_i (x_{i,r,j} - 1), \quad \forall i \in \{1,\ldots,n\}, \ \forall r \in \{1,\ldots,|D_i|\}, \ \forall j \in \{1,\ldots,c\} \tag{13}$$

$$z_{i,r,j} \ge 0 \quad \forall i \in \{1,\ldots,n\}, \ \forall r \in \{1,\ldots,|D_i|\}, \ \forall j \in \{1,\ldots,c\} \tag{14}$$

As the objective function will push $z_{i,r,j}$ downward, Eqs. (11) and (12) are useless. The linearized model is then

$$\text{Max} \quad \sum_{j=1}^{c} t_j$$

$$\text{s.t.} \quad \sum_{j=1}^{c} \sum_{r=1}^{|D_i|} z_{i,r,j} \le b_i \quad \forall i \in \{1,\ldots,n\}$$

$$z_{i,r,j} \ge p_{i,r} t_j + b_i (x_{i,r,j} - 1) \quad \forall i \in \{1,\ldots,n\}, \ \forall r \in \{1,\ldots,|D_i|\}, \ \forall j \in \{1,\ldots,c\}$$

$$\sum_{r=1}^{|D_i|} x_{i,r,j} \le 1 \quad \forall i \in \{1,\ldots,n\}, \ \forall j \in \{1,\ldots,c\}$$

$$\sum_{i=1}^{n} \sum_{r=C_{k,i}}^{|D_i|} x_{i,r,j} \ge 1 \quad \forall k \in \{1,\ldots,m\}, \ \forall j \in \{1,\ldots,c\}$$

$$\sum_{i=1}^{n} \sum_{r=1}^{|D_i|} x_{i,r,j} \le m \quad \forall j \in \{1,\ldots,c\}$$

Table 2
Using the MILP formulation of LM-ASR on four instances.

Instances	n	m	Best sol.	Opt. sol.
n012m005	12	5	4.0459	4.0459
n025m010	25	10	4.8780	4.8780
n050m015	50	15	11.6013	14.0702
n100m030	100	30	19.8385	38.9922

$$x_{i,r,j} \in \{0,1\} \quad \forall i \in \{1,\ldots,n\}, \ \forall r \in \{1,\ldots,|D_i|\}, \ \forall j \in \{1,\ldots,c\}$$

$$t_j \ge 0 \quad \forall j \in \{1,\ldots,c\}$$

$$z_{i,r,j} \ge 0 \quad \forall i \in \{1,\ldots,n\}, \ \forall r \in \{1,\ldots,|D_i|\}, \ \forall j \in \{1,\ldots,c\}$$

This mixed integer linear problem (MILP) has a very large number of variables and constraints, and it may be used for addressing very small instances only. Indeed, we have solved that formulation for LM-ASR with the commercial solver Xpress-MP [7] on the four instances whose characteristics are shown in Table 2 (the hardware and software used in all numerical experiments is described in Section 4). The last column displays the optimal objective value returned by the column generation algorithm. The solver was allowed to solve the MILP formulation for 1 h on each instance, and none of them could be solved to optimality: even if the optimal solution was found for the first two instances, the solver could not prove them optimal. Moreover, it can be seen that solution quality drastically decreases when problem size increases.

The last two instances of Table 2 are also used in Section 4, and can be solved to optimality in less than 10 s with the column generation based algorithms introduced in Section 3.2. In addition, as solving the LP relaxation of the MILP formulation requires even more time than solving the problem to optimality with the column generation based algorithms, using rounding procedures for generating initial covers for the column generation algorithms is too costly. However, this MILP formulation shows that variables t_j and $x_{i,r,j}$ are connected together with a single constraint, which suggests to perform a Dantzig–Wolfe decomposition, resulting in the proposed column generation algorithm.

3.2. Master problem

LM-ASR and LM-PSR can be formulated as the following linear program, where S_j is a non-dominated cover written under the real vector encoding, and c is the number of non-dominated covers:

$$\text{Max} \quad \sum_{j=1}^{c} t_j$$

$$\text{s.t.} \quad \sum_{j=1}^{c} S_{i,j} t_j \le b_i \quad \forall i \in \{1,\ldots,n\}$$

$$t_j \ge 0 \quad \forall j \in \{1,\ldots,c\}$$

The objective function is the sensor network lifetime, and the constraints enforce that each battery has a limited amount of energy. This linear program cannot be solved in practice because it requires enumerating the exhaustive set of non-dominated covers, which cardinality is exponential in n.

Furthermore, as this linear program has n constraints (apart from nonnegativity), a feasible basis is a collection of at most n variables t_j, therefore at most n non-dominated covers are used for a nonzero amount of time in any basic optimal solution [4].

That is the reason why the proposed approach relies on column generation [13]. This iterative approach consists in

solving the linear program above with a limited number c of columns (this problem is referred to as the master problem), then the auxiliary problem introduced in Section 3.3 is solved for generating an attractive column, *i.e.*, a non-dominated cover to be introduced in the master problem for further maximizing its objective function. To this end, cover S_j is built so as to maximize the reduced cost that its associated variable t_j would have once introduced in the master problem. If its reduced cost is strictly positive, S_j is actually added to the master problem as a new column, and the master problem is solved again. This iterative process stops when it is no longer possible to find a non-dominated cover with a strictly positive reduced cost: this proves that the current solution to the master problem is optimal, and the algorithm stops.

The master problem is initialized with a single cover (*i.e.*, initially $c=1$) that involves all the sensors, used at their maximum power level. If the master problem is infeasible, then there exists at least one target out of the range of any sensor, hence the problem has no solution. This initial cover is obviously dominated, but replacing it with a non-dominated one has a very marginal impact on convergence. This is due to the fact that this cover becomes useless and is replaced with more efficient ones after a few iterations.

3.3. Auxiliary problem ILP formulation

At every iteration, the master problem finds the optimal lifetime value based on a restricted set of c columns. The auxiliary problem is to generate an additional attractive column, referred to as cover S_{c+1}, that may allow to increase lifetime further once it is introduced in the master problem. If such a column is found, it is added to the master problem, c is increased by one and the master problem is resolved. If no such column exists, then the current solution of the master problem is optimal, as introducing even all the columns would not allow to increase the lifetime.

Unlike the master problem, the cover under construction in the auxiliary problem is written under the binary encoding. As a cover (or a column) is said to be attractive if and only if its reduced cost is strictly positive, the auxiliary problem objective function is to maximize the reduced cost value associated with the cover S_{c+1}, which is $1-y^T S_{c+1}$, where y is the vector of dual variables of the master problem solved in the current iteration. The auxiliary problem for LM-ASR consists of Eqs. (15)–(19). The auxiliary problem for LM-PSR is obtained by replacing every occurrence of $|D_i|$ and $p_{i,r}$ with M_{PSR} and $f(R_r)$, respectively:

$$\text{Max} \quad 1-\sum_{i=1}^{n}\left(\sum_{r=1}^{|D_i|} x_{i,r,c+1}\,p_{i,r}\right)y_i \tag{15}$$

$$\text{s.t.} \quad \sum_{r=1}^{|D_i|} x_{i,r,c+1} \le 1 \quad \forall i \in \{1,\ldots,n\} \tag{16}$$

$$\sum_{i=1}^{n}\sum_{r=C_{k,i}}^{|D_i|} x_{i,r,c+1} \ge 1 \quad \forall k \in \{1,\ldots,m\} \tag{17}$$

$$\sum_{i=1}^{n}\sum_{r=1}^{|D_i|} x_{i,r,c+1} \le m \tag{18}$$

$$x_{i,r,c+1} \in \{0,1\} \quad \forall i \in \{1,\ldots,n\}\ \forall r \in \{1,\ldots,|D_i|\} \tag{19}$$

The decision variables are $x_{i,r,c+1}$, whereas y_i, $p_{i,r}$, C_k and D_i are given data. The objective function is the reduced cost of the new cover, where $S_{i,c+1}$ have been substituted using Eq. (3). Constraint (16) ensures that each sensor is associated to a single power level (or no power level at all if it is not part of the cover). Constraint (17) states that the cover must be valid, *i.e.*, all targets must be covered by at least one sensor. Note that if sensor s_i does not cover target τ_k, then $r=M+1$ and so no term involving $x_{i,r,c+1}$ is part of the sum. Constraint (18) is the necessary condition stated in Lemma 2 for the cover to be non-dominated. Finally, constraint (19) enforce binary requirements.

The auxiliary problem returns S_{c+1} if its objective value is strictly positive, otherwise the algorithm stops as a zero objective value implies that there does not exists any attractive cover, hence the master problem current solution is optimal.

3.3.1. Relationship between LM-ASR and LM-PSR

The auxiliary problem can be strengthened in the case of LM-PSR. If a sensor is such that there is no target located at distance $d \in]R_r, R_{r+1}]$ with $r \in \{1,\ldots,M_{PSR}-1\}$, then the corresponding power level (namely $f(R_r)$) is never used by this sensor. Consequently, variable $x_{i,r,c+1}$ can be set to zero. Thus, the following equation is added to the auxiliary problem:

$$x_{i,r,c+1} = 0 \quad \forall (i,r) \in \{1,\ldots,n\} \times \{1,\ldots,M_{PSR}\}, \quad f(R_r) \notin p_i \tag{20}$$

Such a situation never happens with LM-ASR, as power levels are defined for each sensor for exactly fitting its neighboring targets with minimum power requirement. Eq. (20) is then specific to LM-PSR.

Despite constraint (18), the cover returned after solving the auxiliary problem may not always be non-dominated. Algorithm 2 is a post-optimization procedure that aims at refining S_{c+1} (under binary encoding) so as to make it non-dominated.

Algorithm 2. Refining S_{c+1} (under binary encoding) into a non-dominated cover.

```
for i = 1 to n do
    for r = |D_i| to 1 do
        if x_{i,r,c+1} = 1 then
            x_{i,r,c+1} ← 0;
            if r > 1 then
                x_{i,r-1,c+1} ← 1;
            breach ← false;
            k ← 1;
            while breach = false and k ≤ m do
                if Σ_{i'=1}^{n} Σ_{r'=C_{k,i'}}^{|D_{i'}|} x_{i',r',c+1} = 0 then
                    breach ← true;
                k ← k + 1;
            if breach = true then
                x_{i,r,c+1} ← 1;
                if r > 1 then
                    x_{i,r-1,c+1} ← 0;
```

Algorithm 2 attempts to reduce the power of all sensors in S_{c+1} to its closest lower level. If the considered sensor is already using its minimum power, it is tested for removal. Then, the coverage constraints (Eq. (17)) are checked. If one of them does not hold, the Boolean variable *breach* is set to true, and the sensor is set back to its original power level. Otherwise, further power reductions and removals are attempted. When this algorithm terminates, the resulting cover is non-dominated as reducing the power or removing any sensor compromises target coverage.

Lemma 3. *Whatever M_{PSR} and the values for the predefined sensing ranges, the objective function value reached by LM-ASR is always larger than or equal to the objective function value reached by LM-PSR.*

Proof. Let $\mathcal{S}^{PSR}$ be an optimal solution to LM-PSR consisting of c covers denoted by S_j^{PSR}, each of them being used a nonzero amount of time t_j. A feasible solution to LM-ASR consisting in c covers used the same amount of time and denoted by $\{S_1^{ASR},\ldots,S_c^{ASR}\}$ is built from $\mathcal{S}^{PSR}$ by transforming the cover S_j^{PSR} for all j in {1, …, c} as follows.

For all i in $\{1,\ldots,n\}$, $S_{i,j}^{ASR}$ is set to $p_{i,r}$ where r is the maximum integer such that $p_{i,r} \le S_{i,j}^{PSR}$. Thus, sensor s_i covers the same targets in S_j^{PSR} and S_j^{ASR} because by definition of $p_{i,r}$ the targets covered by s_i are the same for any power value in $[p_{i,r},p_{i,r+1}[$. Consequently, $S_j^{ASR} \le S_j^{PSR}$ for all $j \in \{1,\ldots,c\}$ so S_j^{ASR} can be used for t_j units of time, leading to the same lifetime duration as $\mathcal{S}^{PSR}$. □

Lemma 4. *LM-PSR and LM-ASR have the same optimal objective value if*

$$\bigcup_{i \in \{1,\ldots,n\} r \in \{1,\ldots,|D_i|\}} p_{i,r} \subseteq \bigcup_{r \in \{1,\ldots,M_{PSR}\}} f(R_r) \tag{21}$$

where $p_{i,r}$ refer to power levels in LM-ASR, $f(R_r)$ being the power levels in LM-PSR, associated with predefined sensing ranges.

Proof. If Eq. (21) holds, then any non-dominated cover for LM-ASR is also non-dominated for LM-PSR. Then, LM-PSR is identical to LM-ASR: for each sensor, all useless power levels are associated a zero decision variable in the auxiliary problem by Eq. (20). Once zero decision variables are removed, the auxiliary problems for LM-ASR and LM-PSR are identical. □

The example in Fig. 2 shows that Eq. (21) is not a necessary condition for LM-PSR and LM-ASR to have identical optimal objective values. Indeed, all power levels in LM-ASR may not be used in an optimal solution. More specifically, s_1 has to maintain its sensing range to $R^{max}=4$ for covering target τ_2, hence its sensing range cannot be decreased to 2. In that case, LM-PSR with only one range (R^{max}) and LM-ASR reach the same optimal lifetime (1 s), whereas Eq. (21) does not hold as $p_1=(\frac{1}{4},1)$ and $f(R_1)=1$.

3.3.2. Example

The example introduced in Section 2.1.1 is used for illustrating the proposed approach as well as Lemma 3.

- LM-ASR
  The initial cover is S_4^{ASR}. At the first iteration, the master problem optimal objective value is $LT=1$, and the auxiliary problem generates a second cover, which turns out to be S_2^{ASR}. At the second iteration, the master problem optimal objective value is $LT=\frac{16}{9}\approx 1.777$, with $t_1=1$ and $t_2=\frac{7}{9}$. The auxiliary problem generates a third cover, which is S_1^{ASR}. At the third iteration, the master problem optimal objective value is $LT=\frac{16}{3}\approx 5.333$, with $t_1=0$, $t_2=\frac{4}{3}$ and $t_3=4$. Then, the auxiliary problem has a zero optimal objective value, which proves that the master problem current solution is optimal.

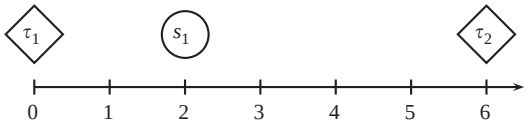

Fig. 2. A one-dimensional example for showing that the converse of Lemma 4 is not true.

- LM-PSR
  The initial cover is S_4^{PSR}. At the first iteration, the master problem optimal objective value is $LT=1$, and the auxiliary problem generates a second cover, which turns out to be S_1^{PSR}. At the second iteration, the master problem optimal objective value is $LT=4$, with $t_1=0$ and $t_2=4$, and the auxiliary problem has a zero optimal objective value, which proves that this solution is optimal.

As shown in Lemma 3 the objective function value of LM-ASR is larger than the one of LM-PSR.

3.4. A genetic algorithm for the auxiliary problem

Solving the auxiliary problem to optimality is sometimes very time-consuming, and results in the production of a single cover. This section describes the main features of the genetic algorithm, referred to as *GA* that we designed for overcoming these drawbacks. More precisely, in the proposed approach, the regular way for generating attractive covers is through *GA*. Upon failure, *GA* is again run afresh one more time, and ILP is used as last resort (*i.e.*, when both runs of *GA* have failed to find any attractive cover), and for proving that the current solution to the master problem is optimal.

Chromosome representation and fitness evaluation: Each chromosome represents a cover and is encoded using the real vector encoding introduced in Section 2.2.1. Three fitness functions are used. The primary fitness function is the objective function of the auxiliary problem that has to be maximized. The secondary fitness function is the power of the cover. The third fitness function is the number of sensors in the cover. The secondary fitness function is needed for distinguishing between two covers having the same value for the primary fitness function. Similarly, the third fitness function is needed for distinguishing between two covers having the same values for primary and secondary fitness functions. A cover is considered to be better than the other, if either it has a higher value according to the primary fitness function or the primary fitness values are equal but it consumes less power than the other or both the primary and secondary fitness values are equal and it has fewer sensors than the other. Three fitness functions are needed because many covers have the same value for the primary and secondary fitness functions. This happens more often in LM-PSR than in LM-ASR.

Selection: Probabilistic binary tournament selection is used to select the two parents for crossover. In probabilistic binary tournament selection, two candidates are picked uniformly at random from the current population and their fitness is compared. With probability π_b, the candidate which has better fitness is selected to be a parent, otherwise the candidate with worse fitness is selected. The second parent is selected in an analogous manner.

Crossover: We have used the two crossover operators. The first crossover operator that is used simply transfers each sensor present in either of the two parents to the child chromosome. However, if a sensor is present in both the parents and have different power levels then for such a sensor the child will have the higher of the two power levels with probability π_h, otherwise it will have the lower of the two power levels. The second crossover that is used is the uniform crossover. The first crossover is used with probability π_f, otherwise the second crossover is used. The two crossovers are needed to maintain a balance between exploitation and exploration.

Mutation: Mutation is applied to the child obtained through crossover. The mutation operator considers each sensor one-by-one, and, if it is present in the child, it deletes it with probability

π_m. If a sensor is not present then mutation operator inserts it at a random power level with probability π_m.

Repair operator The child chromosome obtained after the application of genetic operators may not be feasible. Therefore, a repair operator is designed, which not only converts the child into a feasible cover, but also into a good non-dominated cover. It consist of three stages. The first stage transforms the child into a feasible cover. The second stage tries to improve the value of the auxiliary problem's objective function (hereafter, refers to as objective function in the remainder of this section) by removing some sensors. The third stage tries to further improve the value of the objective function by reducing the range of some sensors.

The first stage considers each target one-by-one and if the target under consideration is not covered by any sensor then it either adds a new sensor to the cover or increases the power level of an existing sensor in a way that covers the target in question and that leads to minimum reduction in the value of the objective function. Coverage information of all targets are updated before considering the next target.

The second stage deletes those sensors from the cover all of whose covered targets are covered by other sensors also. It follows an iterative process. During each iteration it begins by computing the set S_{red} of those sensors in the cover, all of whose targets are redundantly covered. Then from this set it removes the sensor s_i corresponding to the maximum $y_i \times p_{i,r}$ value from the cover, where $p_{i,r}$ is the current power level of sensor s_i. This process is repeated, until the iteration, where S_{red} is found to be empty. Note that deleting sensor with maximum $y_i \times p_{i,r}$ value leads to maximum increase in the value of the objective function.

The third stage ensures that remaining sensors are used only at their respective minimum power levels necessary for covering all targets. It follows an iterative process where during each iteration it decreases the power of a sensor from its current level to its next (strictly) lower level if doing so does not result into a coverage breach. If there are more than one candidate sensors, then the power level of sensor with maximum $y_i \times (p_{i,r} - p_{i,q})$ value is reduced from its current value $p_{i,r}$ to its next (strictly) lower level $p_{i,q}$. Note that for LM-PSR, $(r-q)=1$, whereas for LM-ASR as many power levels can be equal so $(r-q) \geq 1$. This process is repeated until it is not possible to reduce the power level of any sensor.

Replacement policy: Our genetic algorithm uses steady-state population replacement method [5]. In this method genetic algorithm repeatedly selects two parents, performs crossover and mutation to generate a single child that replaces a less fit member of the population. This is different from generational method where the entire parent population is replaced with an equal number of newly created children every generation. In comparison to the generational method, the steady-state population replacement method generally finds better solutions faster. This is due to keeping the best solutions in the population permanently and the immediate availability of the generated child for selection and reproduction. Another advantage of the steady-state population replacement method is the ease with which duplicate copies of the same individuals are avoided in the population. In the generational method, duplicate copies of the highly fit individuals may exist in the population. Within few generations, these highly fit individuals can dominate the whole population. When this happens, the crossover becomes totally ineffective and the mutation becomes the only possible way to improve solution quality. When this happens, improvement, if any, in solution quality is quite slow. Such a situation is known as the premature convergence. In the steady-state method, we can easily prevent this situation by simply comparing each newly generated child with current population members and discarding the child, if it is identical to any current member.

Initial population generation: Each member of the initial population is generated randomly by following a three stage procedure. With probability π_a, the first stage includes as many sensors as possible with $y_i=0$ at their maximum power level. Actually, these sensors do not contribute to the objective function irrespective of their power levels. These sensors are added to the solution one-by-one in some random order and the coverage information of all the target are updated whenever a sensor is added to the solution. The remaining uncovered targets are covered by following an iterative approach. During each iteration an uncovered target is selected randomly and then a sensor that can cover this target is also selected randomly. This sensor can be a new sensor or a sensor already present in the cover but with a power level lower than required for covering the target under consideration. The power level of this sensor is set to the minimum power level necessary for covering the target in question. The coverage information of all the targets are updated. This process is repeated until no target is left uncovered. The second and third stages are exactly similar to the second and third stages of repair operator except during each iteration a candidate is selected randomly. This is done to increase the diversity of initial population.

Each newly generated chromosome is checked for uniqueness against the population members generated so far, and, if it is unique, then it is included in the initial population, otherwise it is discarded.

Other features: If our genetic algorithms found an attractive cover then up to *MAXCOVER* best attractive covers are returned at each iteration which accelerates convergence. If the genetic algorithm fails to find an attractive cover, it is applied afresh one more time before using the ILP. The stopping criterion that is used for our genetic algorithm is the maximum number of consecutive iterations it_{max} without improvement in best solution quality. However, the value of it_{max} varies over the set $OPTION = \{50, 100, 250, 500, 1000, 2000\}$ from one run to the other. If in the previous run genetic algorithm returned less than *MAXCOVER* attractive covers then it_{max} is set to next higher value, if possible, in the *OPTION*, otherwise it_{max} is set to next lower value, if possible, in the *OPTION*. it_{max} is set to 50 in the very first run of the genetic algorithm.

In case of LM-ASR, as many power levels are equal, therefore, whenever a sensor is set to a new power level a check is made to ensure that all targets that can be covered with that much power, are indeed covered.

4. Computational results

4.1. Instances and experimental setup

The instances that we have used in our computational experiments have been generated as follows. n sensors and m targets are generated at random in a 500×500 area. The maximum sensing range of sensors is set to $R^{max} = 150$. The instance name format is `nXXXmYYY`, where `XXX` $\in \{50, 100, 150, 200, 300, 600\}$ is the number of sensors, and `YYY` $\in \{15, 30, 45, 60, 90, 120, 180, 360\}$ is the number of targets. Five instances of the same size have been generated, leading to a grand total of 60 different instances.

In addition to LM-ASR, LM-PSR is addressed for three different numbers of sensing ranges. In PSR_1, sensors are either sensing up to their maximum range R^{max}, or not used at all; this corresponds to the problem where sensing ranges are not adjustable. In PSR_3, the three sensing ranges are $\{50, 100, 150\}$ and in PSR_6, the six considered sensing ranges are $\{25, 50, 75, 100, 125, 150\}$. Thus, the number of sensing ranges considered in this paper is as large as in [3]. The quadratic model f_{qua} defined in Section 2.1 has been used for the sensors' power consumption.

All the approaches presented in this paper are implemented in `C`, a function is used for removing the covers used a zero amount of time when the number of covers exceeds $10(n+m+1)$ in the master problem (this is expected to keep the master

program fast when *GA* returns up to *MAXCOVER* covers per iteration). Algorithm 2 is also run at every iteration for transforming the cover found by the auxiliary problem into a non-dominated cover.

For the genetic algorithm, we have used a population of $\min(n,100)$ individuals, $MAXCOVER=10$, $\pi_a=0.25$, $\pi_f=0.8$, $\pi_h=0.9$, $\pi_m=0.05$. We have used two different values of π_b. The first parent is selected with $\pi_b=0.9$, whereas the second parent is selected with $\pi_b=0.8$. All these parameter values are chosen empirically after large number of trials.

All the computations have been performed on an Intel Xeon Processor system at 2.66 GHz, with 8 GB RAM under Microsoft Windows 7. The LP and ILP solver is GLPK [10].

Table 3
Computation time with and without restricting the search for non-dominated covers.

n150m045	Average CPU time to optimality			
	PSR-1	PSR-3	PSR-6	ASR
Dominated covers	0.43	2.54	7.93	69.18
Non-dominated covers	0.44	2.17	6.89	52.82
Improvement (%)	−2.33	14.57	13.11	23.65

4.2. Results

As a preliminary result, the benefit of using non-dominated covers is assessed on the five instances with $n=150$ sensors and $m=45$ targets, for LM-ASR and LM-PSR. The first row of Table 3 displays the CPU time in seconds of the column generation algorithm for which the cover refinement procedure (see Algorithm 2) is never used, and where Eq. (18) is not present in the auxiliary problem. The second row of the table shows the results where the cover refinement procedure is called after each iteration of the auxiliary problem, and where Eq. (18) is present in the auxiliary problem formulation. In both cases, ILP only is used for addressing the auxiliary problem. The last row displays the CPU time improvement (in percent) when using non-dominated covers.

It can be seen that the benefit of restricting the search for attractive covers to non-dominated ones is more visible for LM-ASR, and for LM-PSR with a high number of predefined sensing ranges. These problems are the most difficult ones, as shown in Tables 4 and 5. Lifetime values are not reported in Table 3 because they are identical.

Table 4 reports the results where the auxiliary problem is addressed with ILP at each iteration, while Table 5 displays the results where *GA* is used for addressing the auxiliary problem, the ILP formulation being solved only when *GA* fails to find attractive covers.

Table 4
Lifetime and computation time when addressing the auxiliary problem with *ILP*.

Instances	Average network lifetime				Average CPU time to optimality			
	PSR-1	PSR-3	PSR-6	ASR	PSR-1	PSR-3	PSR-6	ASR
n050m015	4.000	7.621	9.561	12.262	$0.03^{(5)}$	$0.06^{(5)}$	$0.09^{(5)}$	$0.22^{(5)}$
n050m030	3.600	6.361	8.474	11.291	$0.03^{(5)}$	$0.08^{(5)}$	$0.16^{(5)}$	$0.37^{(5)}$
n100m030	9.000	19.487	24.089	31.665	$1.04^{(5)}$	$0.83^{(5)}$	$2.66^{(5)}$	$4.61^{(5)}$
n100m060	7.000	12.004	14.088	18.937	$0.96^{(5)}$	$0.96^{(5)}$	$2.23^{(5)}$	$178.66^{(5)}$
n150m045	11.800	24.360	32.502	44.934	$0.44^{(5)}$	$2.17^{(5)}$	$6.89^{(5)}$	$52.82^{(5)}$
n150m090	11.600	23.744	29.814	41.993	$1.18^{(5)}$	$11.26^{(5)}$	$126.39^{(5)}$	$95.29^{(2)}$
n200m060	14.600	32.378	46.867	64.455	$1.02^{(5)}$	$6.36^{(5)}$	$47.73^{(5)}$	$235.30^{(4)}$
n200m120	14.400	32.105	39.699	55.003	$2.81^{(5)}$	$20.70^{(5)}$	$67.83^{(5)}$	$270.79^{(2)}$
n300m090	25.000	55.525	74.139	103.320	$5.38^{(5)}$	$82.36^{(4)}$	$285.88^{(3)}$	$984.85^{(1)}$
n300m180	23.400	54.581	67.643	93.930	$14.14^{(5)}$	$1015.86^{(5)}$	$373.81^{(1)}$	$1\ h^{(0)}$
n600m180	51.000	105.683	138.292	58.844	$82.34^{(5)}$	$722.41^{(3)}$	$963.55^{(1)}$	$1\ h^{(0)}$
n600m360	48.600	96.000	90.625	16.176	$221.22^{(5)}$	$1\ h^{(0)}$	$1\ h^{(0)}$	$1\ h^{(0)}$

Table 5
Lifetime and computation time when addressing the auxiliary problem with *ILP+GA*.

Instances	Average network lifetime				Average CPU time to optimality			
	PSR-1	PSR-3	PSR-6	ASR	PSR-1	PSR-3	PSR-6	ASR
n050m015	4.000	7.621	9.561	12.262	$0.01^{(5)}$	$0.02^{(5)}$	$0.03^{(5)}$	$0.08^{(5)}$
n050m030	3.600	6.361	8.474	11.291	$0.02^{(5)}$	$0.03^{(5)}$	$0.04^{(5)}$	$0.12^{(5)}$
n100m030	9.000	19.487	24.089	31.665	$0.03^{(5)}$	$0.47^{(5)}$	$1.41^{(5)}$	$3.53^{(5)}$
n100m060	7.000	12.004	14.088	18.937	$0.04^{(5)}$	$0.15^{(5)}$	$0.28^{(5)}$	$94.69^{(5)}$
n150m045	11.800	24.360	32.502	44.934	$0.05^{(5)}$	$0.29^{(5)}$	$3.48^{(5)}$	$25.43^{(5)}$
n150m090	11.600	23.744	29.814	41.994	$0.10^{(5)}$	$2.69^{(5)}$	$15.24^{(5)}$	$10.28^{(2)}$
n200m060	14.600	32.378	46.867	64.455	$0.08^{(5)}$	$0.29^{(5)}$	$21.03^{(5)}$	$520.87^{(5)}$
n200m120	14.400	32.105	39.699	55.015	$0.13^{(5)}$	$1.98^{(5)}$	$19.27^{(5)}$	$25.44^{(2)}$
n300m090	25.000	57.536	74.143	103.320	$0.20^{(5)}$	$26.73^{(4)}$	$101.59^{(3)}$	$934.02^{(2)}$
n300m180	23.400	54.581	67.643	94.239	$0.32^{(5)}$	$90.26^{(5)}$	$50.59^{(1)}$	$1\ h^{(0)}$
n600m180	51.000	105.901	138.564	184.887	$1.01^{(5)}$	$114.24^{(3)}$	$44.34^{(1)}$	$1\ h^{(0)}$
n600m360	48.600	96.156	122.299	153.522	$2.01^{(5)}$	$673.56^{(3)}$	$1\ h^{(0)}$	$1\ h^{(0)}$

Table 6
Summary comparison of *ILP* alone and *ILP*+*GA* for the auxiliary problem.

Problems	PSR-1	PSR-3	PSR-6	ASR
Number of optimal solutions found with *ILP*	60	52	45	34
Number of optimal solutions found with *ILP*+*GA*	60	55	45	36
CPU time cut with *ILP*+*GA* vs. *ILP* alone (for the instances solved to optimality with *ILP*)	98.78%	88.03%	79.69%	62.24%
Average lifetime with *ILP* on all instances	18.67	37.40	47.98	46.10
Average lifetime with *ILP*+*GA* on all instances	18.67	38.35	48.47	68.07

The first column displays the instance name. Columns 2–5 show the average lifetime for the five instances associated with each row for PSR_1, PSR_3, PSR_6 and ASR. Columns 6–9 display the average CPU time in seconds for returning an optimal solution. Each problem is allocated a maximum amount of 1 h per instance; the number of instances solved to optimality is shown in parenthesis. Consequently, the average CPU time is computed on these instances only.

Finally, Table 6 summarizes the results shown in Tables 4 and 5, by stressing the differences between addressing the auxiliary problem with the ILP formulation, and using the proposed hybrid approach. The table compares both approaches in terms of number of optimal solutions found, displays the CPU time cut provided by the hybrid approach compared to the ILP formulation, the average lifetime on all instances.

4.3. Discussion

Tables 4 and 5 show that for the same number of sensors, doubling the number of targets leads to a smaller lifetime. This is quite clear for LM-ASR that can take advantage of the actual distance between sensors and targets, whereas this is less visible for LM-PSR_1 for which the energy consumption of a sensor depends on R^{max} only. It can also be checked that maximum lifetimes are reached for ASR, which is an experimental verification of Lemma 3. The lifetime decreases when the number of sensing ranges decreases in LM-PSR, which was also a predictable result. The solution space of the auxiliary problem increases with the number of sensing ranges, hence LM-ASR is the most difficult problem and LM-PSR_1 is naturally the easiest one. Column generation algorithms are subject to the well-known tail effect: a very large number of iterations is required at the end of the search for proving optimality. Additionally, the last iterations (*i.e.*, the last generated columns) lead to a very modest increase in lifetime. Consequently, the lifetime found after 1 h is often very close to the optimal lifetime for $n \leq 300$. It is necessary to consider large instances for observing a significant gap in terms of solution quality: 1 h of computation time is not enough for solving LM-ASR on the set of instances n600m180 and n600m360, but it can be seen that using *GA* leads to much better results. Indeed, the instances must be sufficiently large for avoiding the tail effect of column generation to mask the gap between *GA* and *ILP*. For smaller instances, this gap is not visible because *ILP*, which is much slower than *GA*, is still in the tail of the search when the time limit of 1 h is elapsed. The benefit of using *GA* in terms of solution quality is more and more visible for difficult problems; this general trend is clearly visible in the last two rows of Table 6. At first sight, this table also suggests that the benefit of using *GA* is less visible for difficult problems like LM-ASR. But this is not the case: the benefit of using *GA* is clearly visible for large instances (see Tables 4 and 5), and these instances are not taken into account here as the CPU time cut is computed only for instances that are solved to optimality by both approaches. And as can be seen in Table 4, there are less and less large instances solved to optimality with *ILP* for difficult problems.

As stated in Lemma 3, the maximum lifetime is reached for LM-ASR. Naturally, when the number of predefined sensing ranges increases, the lifetime in LM-PSR converges toward its value in LM-ASR.

LM-PSR_1 is the easiest problem as sensors have only two states (sensing with range R^{max} or not sensing at all), LM-PSR is more and more difficult as the number of predefined sensing ranges increases. LM-ASR is the most difficult problem, which is not surprising as each sensor can have a large number of power levels if it covers a lot of targets. Furthermore, Eq. (20) ensures that the number of non-zero decision variables in the auxiliary problem in LM-PSR is less than in its counterpart in LM-ASR, whatever M_{PSR}.

5. Conclusions

In this paper we have proposed an exact approach for maximizing the lifetime of a wireless sensor network where sensing ranges of sensors can be adjusted. Two versions of the problem are considered. In the first version, sensing ranges can be continuously adjusted within the maximum sensing range, whereas in the second one, sensing ranges can be adjusted only within the set of predefined values. We have modelled these problems using a column generation scheme and devised a matheuristic (i.e., a combination of a metaheuristic within an exact approach) to efficiently solve the corresponding models. Theoretical results are provided so as to characterize efficient covers, and some relationships between the problems are also highlighted. The proposed approach addresses the auxiliary problem with an efficient genetic algorithm. Computational results demonstrate that for both versions, the use of genetic algorithm within the column generation scheme significantly reduces the computation time, without compromising optimality.

Our results suggest that the use of non-dominated covers could be efficiently adapted to different versions of the lifetime maximization problems. Similar approaches can be designed for other scheduling problems in wireless sensor networks. As a future work, we intend to experiment with other population based metaheuristics within the column generation scheme to assess their suitability for such use.

References

[1] Akyildiz I, Su W, Sankarasubramaniam Y, Cayirci E. Wireless sensor networks: a survey. Computer Networks 2002;38(4):393–422.
[2] Alfieri A, Bianco A, Brandimarte P, Chiasserini CF. Maximizing system lifetime in wireless sensor networks. European Journal of Operational Research 2007;181:390–402.
[3] Cardei M, Wu J, Lu M, Pervaiz M. Maximum network lifetime in wireless sensor networks with adjustable sensing ranges. In: IEEE international conference on wireless and mobile computing, networking and communications (WiMob'2005); 2005. p. 438–45.
[4] Chvátal V. Linear programming. New York: W. H. Freeman and Company; 1980.
[5] Davis L. Handbook of genetic algorithms. New York: Van Nostrand Reinhold; 1991.
[6] Dhawan A, Vu C, Zelikovsky A, Li Y, Prasad S. Maximum lifetime of sensor networks with adjustable sensing range. In: Seventh ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing; 2006. p. 285–9.
[7] FICO. Xpress-MP 〈http://www.dashoptimization.com/〉; 2009.

[8] Gentili M, Raiconi A. α-Coverage to extend network lifetime on wireless sensor networks. Optimization Letters (2012) 1–16.
[9] Gu Y, Ji Y, Li J, Zhao B. Theoretical treatment of target coverage problem in wireless sensor networks. Journal of Computer Science and Technology 2011;26:117–29.
[10] GLPK (GNU Linear Programming Kit) 〈http://www.gnu.org/software/glpk/〉; February 2012.
[11] Jia J, Chen J, Chang G, Wen Y, Song J. Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius. Computers & Mathematics with Applications 2009;57:1767–75.
[12] Latré B, Braem B, Moerman I, Blondia C, Demeester P. A survey on wireless body area networks. Wireless Networks 2011;17:1–18.
[13] Lübbecke ME, Desrosiers J. Selected topics in column generation. Operations Research 2005;53:1007–23.
[14] Rossi A, Singh A, Sevaux M. A column generation algorithm for sensor coverage scheduling under bandwidth constraints. Networks, http://dx.doi.org/10.1002/net.20466, in press.
[15] Wang C, Thai MT, Li Y, Wang F, Wu W. Optimization scheme for sensor coverage scheduling with bandwidth constraints. Optimization Letters 2009;3(1):63–75.
[16] Wang J, Medidi S. Energy efficient coverage with variable sensing radii in wireless sensor networks. In: Third IEEE international conference on wireless and mobile computing, networking and communications (WiMob 2007). White Plains, New York, USA; 2007.
[17] Wu J, Yang S. Coverage issue in sensor networks with adjustable ranges. In: Proceedings of the international conference on parallel processing workshop, ICPP 2004; 2004. p. 61–8.
[18] Zhou Z, Das S, Gupta H. Variable radii connected sensor cover in sensor networks. ACM Transactions on Sensor Networks 2009;5(1):8:1–36.