Minimum energy target tracking with coverage guarantee in wireless sensor networks

Charly Lersteau[a], André Rossi[b], Marc Sevaux[a,*]

[a] *Université de Bretagne-Sud – Lab-STICC, CNRS, UMR 6285 – Lorient, France*
[b] *Université d'Angers – LERIA-Angers, France*

Abstract

Wireless Sensor Networks (WSN) are composed of low-cost sensors designed to monitor targets inside their sensing range. The sensors are randomly dispatched in a region and have a limited battery capacity. The targets are moving and their trajectory are subject to uncertainty. A way to save energy of the WSN is to activate subsets of sensors covering all the targets. The challenge of this paper is to preserve and balance the residual capacities of the sensors in order to perform further target tracking missions using the same WSN. A two-step exact method is proposed. First, the input data is processed in order to design a mathematical formulation. Second, a column generation algorithm, combined with a GRASP metaheuristic, assigns activation time to sensors.

Keywords: wireless sensor networks, target tracking, column generation, matheuristics

1. Introduction

1.1. Context

Wireless sensor networks have received a particular attention in the past few years, and have been used in a growing number of applications including traffic control and battlefield surveillance [1, 2]. A set of sensors is randomly deployed in a region in order to track targets. Several targets are moving through the sensing ranges of the different sensors. Each sensor has a limited battery capacity and a sensing unit able to collect information about the targets in its sensing range. Since the sensors are autonomous and the batteries are non-rechargeable, an important aspect of wireless sensor networks is the energy consumption. A convenient way to save battery lifetime is to activate a subset of the sensors only when necessary. Supposing that we are provided a prevision of the trajectories of the targets, we aim at scheduling sensor activities, ensuring that all the targets are monitored at any time. The target trajectories are also subject to uncertainty. At any time, their position can deviate up to a certain distance from the prevision.

*Corresponding author
Email addresses: charly.lersteau@univ-ubs.fr (Charly Lersteau), andre.rossi@univ-angers.fr (André Rossi), marc.sevaux@univ-ubs.fr (Marc Sevaux)

The *zone of interest* is a subset of the region that should be monitored in further missions. In our context, further missions are monitoring tasks performed by the same set of sensors once current monitoring tasks are terminated. In order to maintain the WSN ability to monitor the zone of interest for future missions, the challenge is to preserve and balance the residual energy capacities. Thus, the first objective is to maximize the minimum amount of time during which the wireless sensor network can monitor any point of the zone of interest. This duration is referred to as the *coverage guarantee*. Having maximized the coverage guarantee, we search for a solution that minimizes the energy consumption for accomplishing the current target tracking mission, while maintaining the maximum coverage guarantee.

1.2. Related work

Studies about wireless sensor networks are abundant in the literature. Plenty of protocols are proposed, considering aspects including energy consumption [3–6], tracking precision [7, 8], scalability [9], and fault tolerance [10–14]. Naderan et al. [15] presents a more exhaustive review of the criteria used in the WSN protocols. Guo and Zhang [16] survey intelligence-based routing protocols based on optimization techniques to prolong WSN lifetime. Rault et al. [17] propose a taxonomy of WSN applications and a classification of energy-conservation schemes.

1.2.1. WSN for static targets

The optimization problems involving static targets are well studied. One of the most popular is the network lifetime maximization problem. The network lifetime is the amount of time elapsed until a target is not covered anymore. The mission consists in finding a schedule of sensor activites maximizing the network lifetime.

This problem has numerous variants. MNLB (Maximizing Network Lifetime under Bandwidth constraints) and MCBB (Minimizing Coverage Breach under Bandwidth constraints), are solved using heuristics [18] or column generation [19]. A variant involving heterogeneous networks is handled by Carrabs et al. [20], by speeding up a column generation algorithm using a genetic algorithm. Castaño et al. [21] address the problem of network lifetime maximization with communication and multi-roles sensors. In their column generation approach, they solve the pricing problem using constraint programming and Branch-and-Cut based on Benders' decomposition. Carrabs et al. [22] propose a new column generation algorithm combined with a genetic algorithm and improve the formulation proposed in [21] by using fewer integer variables. In [23], they allow partial coverage and describe a genetic algorithm designed to speed up the column generation. Given a set of group of directional sensors, Singh and Rossi [24] propose to solve two scheduling problems: the total rotation minimization problem (TRMP) and the minimum blindness problem (MBP). They show that (MBP) is equivalent to the Hamiltonian path problem and design a greedy heuristic and a genetic algorithm for (TRMP). Lu et al. [25] provide a polynomial-time approximation algorithm for the network lifetime maximization problem where the density of targets is bounded, taking into account communication costs with a base station. Supposing that the energy consumption of the sensors is proportional to the number of monitored targets, then the network lifetime maximization problem is polynomial. Liu et al. [26] provide a continuous linear

formulation. Given a set of covers, i.e. subsets of sensors, the Cover Scheduling Problem (WSN-CSP) is to find a schedule minimizing the longest continuous duration of time for which a target is not covered [27, 28].

1.2.2. WSN for target tracking

In the field of moving target tracking, the survey by Naderan et al. [15] reports only one target tracking protocol relying on optimization techniques, designed by Lee et al. [29] and extended by Yeong-Sung et al. [30]. The optimization problem takes target movement frequencies as input, and consists in minimizing the communication costs. The proposed protocol builds an object tracking tree and uses a Lagragian relaxation-based heuristic algorithm based on 0/1 linear formulation. Atia et al. [31] study the problem of finding a schedule of sensor activities optimizing the tradeoff between tracking performance and energy consumption, using a partially observable Markov decision process. Shi et al. [32] provide a prediction method and a sensor scheduling scheme, aiming at balancing tracking quality and network lifetime. The purpose of the method is to guarantee that the target moving area is covered by at least k sensors with probability of at least α.

1.2.3. Robustness for WSN

Because targets and sensors are immersed in a changing, uncertain environment, robustness is a highly desirable feature for a target tracking system. Chen et al. [33] propose a distributed sensor activation algorithm (DSA^2) suitable for binary sensors, activating sensors with probabilities. Robustness is studied according to the maximum velocity of the targets, the sensing range or the sensor density. Many of the works in robustness are focused on network survivability, i.e. the ability to resist to failures including enemy attacks or sensor deficiencies [34–36]. A prediction scheme for maintaining track continuity for ground battlefield surveillance is proposed in [37]. Targets are supposed to be able to move on and off a road under several motion models. Our previous study [38] aims at finding a robust schedule covering a single target at any time, assuming that the target is subject to temporal uncertainty, i.e. the target can be early or late from a prevision.

1.2.4. Contribution and paper organization

This paper focuses on the ability of a sensor schedule to resist to target behavior perturbations, in particular spatial uncertainty. Although this paper is to schedule sensor activities, it does not rely on the classical scheduling paradigm, where tasks are part of the problem input. By contrast, tasks can be seen as the result of the scheduling activity, which consists in deciding when and how long each sensor should be used. An comprehensive survey about robustness in scheduling can be found in [39].

In [40], we consider the total energy consumption minimization and the network lifetime maximization problems for moving targets having a fully deterministic and known trajectory. The present work takes a step further from these approaches in two major directions. First, by considering the global context of the WSN, i.e. by taking into account the long-term objectives of the WSN manager, with the introduction of the zone of interest and the coverage guarantee. Hence, all the optimization efforts are not spent on the current mission, at the

expense of the future use of the WSN. Second, by considering not only one but many targets, and by explicitly taking into account the fact that the spatial position of a moving target cannot be exactly known at any time. To do so, the notion of uncertainty disc is introduced for modeling the zone in which a target is likely to be found, but the approach proposed in this paper can be applied to those cases where this zone of uncertainty is not a disc, as in [41].

This paper is organized as follows. Section 2 presents the problem investigated in this work. A preliminary step called *discretization* is introduced, in order to transform the input data into a scheduling problem instance. Such a procedure is necessary to present the mathematical formulations in Section 3. The column generation procedure is presented in Section 4. Section 5 presents and discusses numerical results and Section 6 concludes this paper.

2. Preliminaries

This section introduces the notations used throughout the paper, and presents the data processing procedure called discretization. A complexity analysis of the three subproblems into which the main problem is decomposed is performed, and two particular cases where they can be solved in polynomial time are presented. A procedure for reducing the instance size is given before proposing a mathematical formulation of these three subproblems. Finally, two upper bounds are provided for the last two subproblems.

2.1. Notations

A wireless sensor network is composed of a set of m sensors (denoted by I) randomly dispatched in a two-dimensional region. Its purpose is to continuously monitor a set of n moving targets (denoted by J) in the sensing range of the sensors (discs of radius R). The expected trajectory of each target is known and is denoted by $\mathcal{T}_j(t)$. Without loss of generality, we consider that $\mathcal{T}_j(t)$ is defined for $t \in [0, H]$, where H is the time horizon. Each sensor i is equipped with a battery having lifetime E_i. Table 1 describes the initial input data of the problem.

I	Set of sensors $\{1, \ldots, m\}$
J	Set of targets $\{1, \ldots, n\}$
(x_i, y_i)	Position of sensor i, $(x_i, y_i) \in \mathbb{R}^2$
R	Radius of the sensing disc of the sensors
E_i	Battery lifetime of sensor i
H	Time horizon
F^*	Zone of interest
$\mathcal{T}_j(t)$	Expected trajectory of the target j for all $t \in [0, H]$

Table 1: Initial problem input

A sensing *activity* is the action of activating a sensor for monitoring targets during a certain amount of time. Then, we aim at finding a schedule of activities meeting these requirements:

- The sum of the activities of a sensor i cannot exceed its battery lifetime E_i.
- Every target must be monitored by at least one sensor throughout the mission.

The *zone of interest*, denoted by F^*, defines a subset of the region in which the sensors will be needed for further missions. The problem investigated in this paper is decomposed into three consecutive subproblems. The first one, denoted by MRC for *Maximize minimal Residual Capacity*, is to find a feasible schedule for the current target tracking mission, if such a solution exists. The so-called residual capacity of a sensor is the amount of time during which that sensor can be used after the current mission. After solving MRC to optimality, if there exists a sensor with a strictly negative residual capacity, then the problem instance is infeasible, as the current mission cannot be accomplished without violating physical battery constraints of at least one sensor. The second one is denoted by MCG for *Maximize Coverage Guarantee in Target Tracking*. MCG is to find a feasible schedule for the current target tracking mission that maximizes the coverage guarantee. The *coverage guarantee* is the minimum amount of time that the sensor network is able to monitor any point of the zone of interest after the current mission. The third subproblem, denoted by MEC for *Minimize Energy Consumption in Target Tracking*, is to find a feasible schedule that minimizes the total amount of energy required for the current target tracking mission, while ensuring that the coverage guarantee is maximum (i.e. has the value returned by MCG). Thus, we aim at finding a solution that preserves the network coverage ability while minimizing the total amount of energy required to complete the mission.

The subproblems are consecutive in the sense that the output of one subproblem is the input of the next subproblem. Indeed, solving MRC ensures feasibility of MCG, and the value of the coverage guarantee computed in MCG is needed in MEC.

The spatial trajectories of the targets are subject to uncertainty. At any instant t, a target j can be far up to a distance of δ from the expected position $\mathcal{T}_j(t)$. This implies the need to cover an uncertainty disc. We define the uncertainty disc as the set of all the possible locations of target j at time t. The uncertainty disc is centered at $\mathcal{T}_j(t)$ and has a radius of δ. In this model, uncertainty is defined by δ, where δ may vary over time.

2.2. Discretization

The monitored region is partitioned into zones called *faces*. It can be seen as a planar graph [42, 43], in which the vertices represent the intersections between the boundaries of the sensors' discs and the edges join the vertices. Thus, the faces are delineated by the vertices and the edges. All the points inside a face are covered by the same set of sensors. This property is convenient since the shape of the trajectory is not needed to solve the problem. Then, we define a *face* by a unique set of covering sensors. For example, the geometric faces 7 and 7′ in Figure 1 are considered as only one face numbered 7, since they are covered by the same set of sensors $\{s_2\}$. The set of all faces is denoted by $\hat{F}$, and the subset of faces in which the targets are moving is $F \subseteq \hat{F}$. Then, the zone of interest F^* can be formally defined as a subset of the faces of F.

First, we consider the case where the target positions are exactly known, i.e. each target is represented as a single point (which is equivalent to $\delta = 0$). For each target $j \in J$,

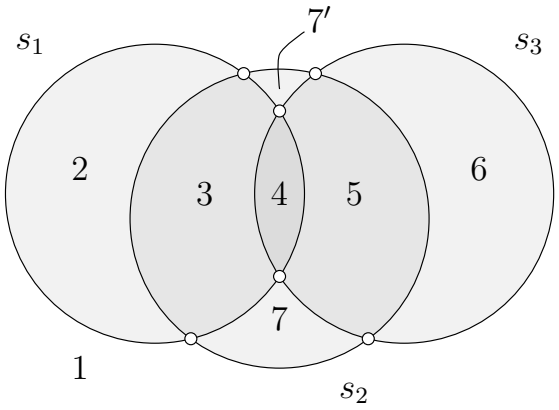

Figure 1: A planar graph based on 3 sensors with 7 faces

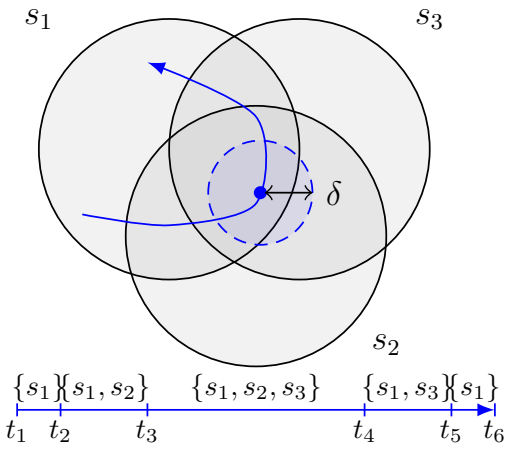

Figure 2: Sets of candidate sensors along the target trajectory

the trajectory can be formulated as a sequence $(f_1^j, f_2^j, \ldots, f_p^j) \in F^p$ of the traversed faces, where p is the number of times that the target enters a face. A *tick* t_k^j is defined by a date of transition of a target from the face f_{k-1}^j to f_k^j. As soon as we compute the union of all the ticks t_k^j into one unique set of ticks t_k, the moving target problem can be seen as a sequence of face covering problems, since between two consecutive ticks t_k and t_{k+1}, each target stays in the same face.

The case where trajectories are subject to uncertainty is now considered. Then, at any instant of time, the uncertainty disc of a target can overlap several faces. In Figure 2, an example of uncertainty disc, represented as a dashed circle, overlaps simultaneously two faces. Consequently, the coverage requirement of the target implies the coverage of all the faces that have a nonempty intersection with the uncertainty disc at the considered instant of time. Naturally, the number of sensors needed to cover the corresponding target can only increase with δ. However, under uncertainty, the moving target problem can still be seen as a sequence of face covering problems. In this case, a tick corresponds to a date for which the set of faces to cover changes, i.e. a target disc meets a new face or leaves another one. Consequently, the problem where trajectories are subject to uncertainty can be reduced to a larger instance of face covering problem, without loss of generality.

The time horizon is partitionned into *time windows*, where a time window is a period between two consecutive ticks. Let K be the set of time windows. The duration of a time window $k \in K$ is denoted by $\Delta_k = t_{k+1} - t_k$. The set of faces to cover during the time window k is denoted by $T(k) \subseteq F$. A candidate sensor for a face $f \in F$ is a sensor that is able to cover this face. The set of all the candidate sensors for face f is denoted by $S(f) \subseteq I$.

2.3. Example

Consider a WSN of 3 sensors (denoted by s_1, s_2, s_3) monitoring 2 targets (denoted by t_1, t_2) as illustrated in Figure 3. The monitored area is discretized into faces denoted by

f_k where k is the index of the face. We suppose that the time horizon is $H = 150$ units of

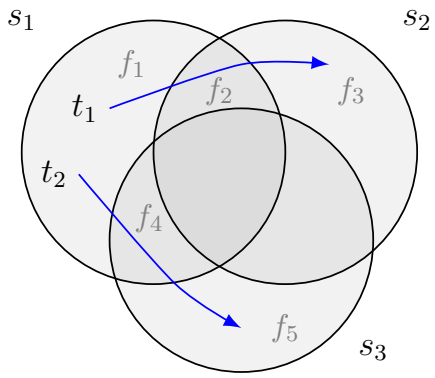

Figure 3: A simple example of 3 sensors and 2 targets

time and the targets are not subject to uncertainty. Along their trajectory, we assume that the two targets cross the sensor boundaries simultaneously. This means that the target t_1 reaches the face f_2 exactly when the target t_2 reaches the face f_4, and that t_1 reaches f_3 exactly when t_2 reaches f_5. Therefore, the time horizon is partitionned into 3 time intervals of 50 units of time. The initial battery lifetime of each sensor is $E_i = 100$ units of time.

Suppose we want a schedule that minimizes the total energy consumption. This is achieved by activating first s_1 during 100 units of time, then s_2 and s_3 simultenaously during 50 units of time. However, the residual capacity of s_1 would be zero. If a new target further entered the face f_1, no sensor would be able to monitor it. Then we aim at maximizing the ability of the WSN to monitor the zone of interest. Suppose that the face f_1 is the zone of interest. An optimal schedule activates s_1 during 50 units of time, then s_2 and s_3 during 100 units of time. Then the WSN is able to further monitor f_1 during 50 units of time. If the zone of interest is composed of all the visited faces, an optimal schedule activates s_1 during 75 units of time, then s_2 and s_3 simultaneously during 75 units of time. This guarantees that each face can be further monitored during 25 units of time.

Lemma 1. *Deciding if MRC is feasible is NP-complete.*

Proof. Consider an instance of MRC with only one time window of duration Δ. Then the problem is equivalent to the decision problem version of MNLB [19], which is NP-complete, i.e. does there exist a schedule such that the network lifetime is greater than or equal to Δ? □

Corollary: By definition of MCG and MEC, it can also be deduced that deciding if each of these two problems is feasible is NP-complete too.

It is recalled that F is the set of all the faces that are visited by a target for a strictly positive amount of time. For all $\Omega \subseteq F$, we define $H(\Omega) = \bigcup_{f \in \Omega} S(f)$ as the set of sensors that can cover at least one face in Ω.

Lemma 2. (Sufficient condition for infeasibility)
If there exists $\Omega \subseteq F$ *such that* $\sum_{i \in H(\Omega)} E_i < \sum_{k \in K | \Omega \cap T(k) \neq \emptyset} \Delta_k$, *then MRC is infeasible.*

Proof. If the amount of time during which a collection of faces must be monitored exceeds the total time during which the sensors able to monitor them are available, the problem is obviously infeasible. □

Lemma 3. (Sufficient condition for feasibility)
Let $K(i) = \{k \in K | \exists f \in T(k), i \in S(f)\}$ *be the set of time windows in which sensor* i *is candidate. If* $\exists i \in S(f), E_i \geq \sum_{k \in K(i)} \Delta_k$ *for all face* $f \in F$, *then MRC is feasible.*

Proof. If for each face there exists a sensor that can monitor it (and is also able to monitor any other face it can cover) for a duration that exceeds the total monitoring time of the face, then a feasible solution can be built by activating this sensor every time the face has to be covered. □

Corollary: These sufficient conditions for infeasibility and feasibility also hold for MCG and MEC.

2.4. Two particular cases where MCG and MEC can be solved in polynomial time

The following two particular cases are extreme scenarios where MCG and MEC can be solved to optimality (or be proven infeasible) in polynomial time. In the first case, all the targets are assumed to be coverable by any sensor at any time. This corresponds to situations where the sensing range R of the sensors is very large. By contrast, the second case models situations where the sensing range is so small that no sensor can cover more than one target at a time. In both cases, MCG is formulated as a linear program of reasonable size. Then, the optimal solution to MCG is also proven to be optimal for MEC. Since solving a linear program is achievable in polynomial time with interior point methods, then so are MCG and MEC in these two special cases.

2.4.1. Long-range sensors

Let us first consider the case where all the sensors are able to cover all the targets at any time. In that case, the optimal objective value of MEC is obviously H as a single sensor is used at any time. In addition, it can be deduced that all the targets are moving in the same face together, so there is no need for discretization. It is now shown how MCG can be solved to optimality or be proven infeasible, in polynomial time. MCG is formulated as the following linear program denoted by LP_{long} and defined by the equations (1)-(6). Let x_i be the continuous, nonnegative decision variable representing the total amount of time during which sensor i is active. The continuous, nonnegative variable T^{min} is the duration of the coverage guaranty offered for the faces in F^*.

$$\max \ T_{\min} \tag{1}$$

$$\sum_{i \in I} x_i = H \tag{2}$$

$$x_i \leq E_i \qquad \forall i \in I \tag{3}$$

$$\sum_{i \in S(f)} (E_i - x_i) \geq T_{\min} \qquad \forall f \in F^* \tag{4}$$

$$T_{\min} \geq 0 \tag{5}$$

$$x_i \geq 0 \qquad \forall i \in I \tag{6}$$

Constraint (2) ensures that the mission duration is H units of time, and (3) enforces that the battery of the sensors are not exceeded. Constraint (4) enforces the coverage guaranty for all the faces of interest. LP_{long} has $\mathcal{O}(m)$ variables and $\mathcal{O}(m^2)$ constraints, because the number of faces is less than m^2 [42]. Hence, since its size does not increase exponentially with the problem data, it can be dealt with by a solver without having to apply decomposition techniques.

LP_{long} is infeasible if and only if $\sum_{i=1}^{m} E_i < H$. In that case, the sensors do not have enough energy to cover the targets for H units of time, therefore MCG and MEC are both infeasible too. If LP_{long} is feasible, then its optimal solution is used to build an optimal solution to MCG and MEC as follows. The sensors are activated in any order, and sensor i is used for x_i units of time for all $i \in I$. This solution is obviously optimal for MEC because constraint (2) ensures that the total amount of energy spent over the mission is minimum.

2.4.2. Short-range sensors

Let us now consider the case where the targets are located in such a way that no sensor can cover more than one target at a time. A sufficient condition for this to happen is when the distance between any pair of targets is always strictly larger than $2R$.

In that particular case, the optimal objective value of MEC is nH. Indeed, since no sensor can cover two targets, then n sensors should be active at any time, hence the total energy required by the mission is nH in any feasible solution. It is now shown how MCG can be solved to optimality or be proven infeasible, in polynomial time.

Let $F^j \subseteq F$ be the set of faces visited by target j, for all $j \in J$. Note that if the same face is visited multiple times by target j, then it appears only once in F^j. The total amount of time spent by target j in face $f \in F^j$, is denoted by Δ^{jf} for all $f \in F^j$. It should also be noted that $F^j \cap F^{j'}$ may be nonempty for $j \neq j'$, but the faces in $F^j \cap F^{j'}$ are never visited by targets j and j' at the same time by hypothesis.

MCG is formulated as the following linear program, denoted by LP_{short} and defined by the equations (7)-(12). Let x_i^{jf} be the continuous, nonnegative decision variable representing the total amount of time spent by sensor i monitoring the face f for tracking target j. The continuous, nonnegative variable $T_{\min}$ is the duration of the coverage guaranty offered for the faces in F^*.

$$\max \; T_{\min} \tag{7}$$

$$\sum_{i \in S(f)} x_i^{jf} = \Delta^{jf} \qquad \forall j \in J, \forall f \in F^j \tag{8}$$

$$\sum_{j \in J} \sum_{f \in F^j} x_i^{jf} \leq E_i \qquad \forall i \in I \tag{9}$$

$$\sum_{i \in S(f)} \left(E_i - \sum_{j \in J} \sum_{f' \in F^j | i \in S(f')} x_i^{jf'} \right) \geq T_{\min} \qquad \forall f \in F^* \tag{10}$$

$$T_{\min} \geq 0 \tag{11}$$

$$x_i^{jf} \geq 0 \qquad \forall j \in J, f \in F^j, \forall i \in I \tag{12}$$

Constraint (8) ensures that each face is covered for the required amount of time, and (9) enforces that the battery of the sensors are not exceeded. Constraint (10) enforces the coverage guaranty for all the faces of interest. $\mathrm{LP}_{\mathrm{short}}$ has $\mathcal{O}(nm^3)$ variables and $\mathcal{O}(nm^2)$ constraints, so its size is reasonably low to be dealt with by a solver.

If $\mathrm{LP}_{\mathrm{short}}$ is infeasible, then MCG and MEC are both infeasible too, as the initial energy of the sensors is insufficient to cover all the targets. If $\mathrm{LP}_{\mathrm{short}}$ is feasible, then its optimal solution is used to build a solution to MCG and MEC as follows. For each target j and each face $f \in F^j$, a sensor in $S(f)$ such that $x_i^{jf} > 0$ is used for x_i^{jf} units of time. If $\Delta^{jf} > x_i^{jf}$, then another sensor i' such that $x_{i'}^{jf} > 0$ is used. The sensors can be used in any order, as the scheduling problem is preemptive (it is recalled that a face can be visited multiple times by the targets). The solution can be easily checked optimal for MEC by summing up the constraint (8) for all j (the right-hand side amounts to H for all j, since each target is to be monitored for H units of time). Then, summing up the n resulting equalities, the total amount of energy spent is found to be nH.

2.5. Problem reduction

Due to the density of the sensor network, the number of time windows can be very high. However, redundancy can be exploited to reduce the instance size.

Definition 1. (Dominance relation between faces)
Let $k \in K$ be a given time window, and let f, f' be two distinct faces of $T(k)$. The face f dominates f' if $S(f) \subset S(f')$.

If a face $f \in T(k)$ dominates $f' \in T(k)$, then the face f' can be removed from the sensing requirement and ignored, since selecting a sensor in $S(f)$ is sufficient to cover both faces f and f'. Removing all dominated faces increases the opportunity to find two time windows k and k' such that $T(k) = T(k')$. Then, time windows k and k' are merged, which is a desirable transformation as the problem instance size decreases. Algorithm 1 reduces the number of time windows.

The duration of the time window k' is added to the time window k ($\Delta_k \leftarrow \Delta_k + \Delta_{k'}$) and the time window k' is removed. The index k' is then associated to k in an index array $\mathcal{K}$. The number of time windows is therefore reduced and the original time windows can

Algorithm 1: Time windows reduction

```
// 𝒦 is the ordered list of time windows
∀k ∈ K, 𝒦(k) ← k
foreach k ∈ K do
    foreach k' ∈ K | k' ≥ k + 1 do
        if T(k) = T(k') then
            Δ_k ← Δ_k + Δ_k'
            K ← K \ {k'}
            𝒦(k') ← k  // Associate k' to k
```

be recovered using the ticks and $\mathcal{K}$. These reductions are useful for the column generation procedure discussed in Section 4, because a NP-hard pricing problem is to be solved for each time window.

Lemma 4. *MRC can be transformed in such a way that the number of time windows, $|K|$, is upper bounded by $2^{m(m-1)+1}$.*

Proof. Let $T(k)$ be the set of faces to be monitored during time window k. If $T(k) = T(k')$ with $k \neq k'$, then time windows k and k' can be merged, leading to delete time window k', and to extend the duration of time window k to $\Delta_k \leftarrow \Delta_k + \Delta_{k'}$. This elimination procedure is repeated as long as there is no two time windows having the same set of faces. Since the maximum number of faces is less than $m(m-1)+2$ [42], the maximum number of time windows is less than $2^{m(m-1)+1}$. This upper bound can naturally be improved by considering $F_{max} = \max_{k \in K} |T(k)|$ the maximum number of faces to monitor in a time window. Then, the number of time windows is upper bounded by $2^{F_{max}-1} \leq 2^{m(m-1)+1}$. □

This lemma shows that the maximum theorical number of time windows is independent of the number of targets and of the trajectories. However, in practice, few time windows are produced when the targets visit few faces.

Corollary: The number of time windows is the same in MRC, MCG and MEC.

3. Mathematical formulations

Figure 4: The sequence of three subproblems

After discretization, the problem is decomposed into three consecutive subproblems as illustrated in Figure 4. Subproblems MRC, MCG and MEC can be formulated as linear programming models respectively denoted by MPMRC, MPMCG and MPMEC. Solving

consecutively these three subproblems results in an optimal solution that is a schedule of cover activities. Additional nonlinear programs and mixed integer linear programs are provided in Appendix A, Appendix B and Appendix C to show the relevance of the column generation decomposition.

3.1. MRC subproblem

The MRC subproblem is designed to build a feasible set of covers to achieve the current mission or to prove infeasibility. It is recalled that a cover is a set of sensors that can cover all the targets at a given instant of time. Solving MRC is necessary to address MCG and MEC because the solution process of both these subproblems require a feasible set of covers as input. The objective of MRC is to maximize the smallest residual capacity among the sensors.

Let $\mathcal{C}$ be the set of all the covers and let $\mathcal{C}(k)$ be the set of feasible covers for the time window $k \in K$. Let a_{ic} be a binary constant equal to 1 if the sensor $i \in I$ belongs to the cover $c \in \mathcal{C}$, 0 otherwise. The decision variables of the linear program MPMRC, defined by the equations (13)-(19), are the following:

- d_c^k is the activity duration of the cover $c \in \mathcal{C}(k)$ during the time window $k \in K$
- r_i is the residual capacity of sensor $i \in I$
- $r_{\min}$ is the smallest residual capacity over all the sensors of I

The dual variables, π_i and μ_k appear in brackets.

MPMRC

$$\max\ r_{\min} \tag{13}$$

s.t.

$$\sum_{k \in K} \sum_{c \in \mathcal{C}(k)} a_{ic} d_c^k + r_i = E_i \qquad \forall i \in I \quad [\pi_i] \tag{14}$$

$$\sum_{c \in \mathcal{C}(k)} d_c^k = \Delta_k \qquad \forall k \in K \quad [\mu_k] \tag{15}$$

$$r_i \geq r_{\min} \qquad \forall i \in I \tag{16}$$

$$d_c^k \geq 0 \qquad \forall k \in K, c \in \mathcal{C}(k) \tag{17}$$

$$r_i \in \mathbb{R} \qquad \forall i \in I \tag{18}$$

$$r_{\min} \in \mathbb{R} \tag{19}$$

Constraint (14) states that the total activity of a sensor cannot exceed its capacity when r_i is nonnegative. Constraint (15) ensures that the targets are covered at any time during each time window. Finally, constraint (16) ensures that $r_{\min}$ is equal to the smallest residual capacity. Note that the domain of r_i is $\mathbb{R}$. MRC is feasible if and only if the optimal value of $r_{\min}$ is positive or zero. Thus, computing the optimal value of $r_{\min}$ is not always needed since non-negativity of $r_{\min}$ is equivalent to feasibility of MRC, MCG and MEC.

3.2. MCG subproblem

Inside the zone of interest, each face has a *monitoring potential* defined as the sum of the capacities of the candidate sensors, after the current mission. The coverage guarantee, denoted by $T_{\min}$, is defined as the minimal monitoring potential among all the faces of interest. Thus, $T_{\min}$ is the minimum time during which the wireless sensor network can monitor any face in the zone of interest F^* after the end of the current mission. However, we do not ensure that two faces of interest can be simultaneously monitored during this amount of time. The objective of MCG is to compute $T^*_{\min}$, the maximum value of the coverage guarantee $T_{\min}$, in order to fix this value in the MEC subproblem.

MCG can also be decomposed into a master problem and a pricing problem to apply a column generation algorithm. The sets of covers $\mathcal{C}$ and $\mathcal{C}(k)$ are identical to the ones introduced in MRC. Let a_{ic} be a binary constant equal to 1 if the sensor $i \in I$ belongs to the cover $c \in \mathcal{C}$, 0 otherwise. The formulation of MPMCG include the following variables:

- d_c^k is the activity duration of the cover $c \in \mathcal{C}(k)$ during the time window $k \in K$
- r_i is the residual capacity of sensor $i \in I$
- $T_{\min}$ is the duration of the coverage guarantee

MPMCG

$$\max T_{\min} \tag{20}$$

s.t.

(14) – (15)

$$\sum_{i \in S(f)} r_i \geq T_{\min} \qquad \forall f \in F^* \quad [\lambda_f] \tag{21}$$

$$T_{\min} \geq 0 \tag{22}$$

$$r_i \geq 0 \qquad \forall i \in I \tag{23}$$

(17)

Constraint (21) ensures that $T_{\min}$ is equal to the coverage guarantee. MPMRC and MPMCG share the same constraints (14)-(15), for enforcing the battery capacity constraints and the target coverage requirements of the current mission. Then every feasible cover for MPMRC is feasible for MPMCG.

3.3. MEC subproblem

MCG subproblem can have several optimal solutions, corresponding to schedules with different energy consumptions for achieving the current mission. MEC is then to return a solution that minimizes the energy consumption incured by the current mission, among those that maximize the coverage guarantee. The sets of covers $\mathcal{C}$ and $\mathcal{C}(k)$ are identical to the ones used for MRC and MCG. Let a_{ic} be a binary constant equal to 1 if the sensor $i \in I$ belongs to the cover $c \in \mathcal{C}$, 0 otherwise. The decision variables of the following linear program MPMEC are the following:

- d_c^k is the activity duration of the cover $c \in \mathcal{C}(k)$ during the time window $k \in K$
- r_i is the residual capacity of sensor $i \in I$

The objective of MEC is to minimize the total energy consumption, that is equivalent to maximize the sum of the residual capacities. For the sake of clarity, the maximization version is considered in the rest of the paper. As a result, the subproblems MRC, MCG and MEC share the same pricing problem for the column generation algorithm introduced in Section 4.

$$\text{MPMEC} \qquad \min \sum_{i \in I} (E_i - r_i) \Leftrightarrow \max \sum_{i \in I} r_i \tag{24}$$

s.t.

(14) − (15)

$$\sum_{i \in S(f)} r_i \geq T^*_{\min} \qquad \forall f \in F^* \quad [\lambda_f] \tag{25}$$

(17), (23)

Constraint (25) ensures the coverage guarantee for each face of interest. MPMCG and MPMEC share the same constraints (14)-(15). Consequently, any feasible set of covers for MPMCG is feasible for MPMEC.

However, MPMCG and MPMEC do not share the same solutions. Indeed, there exist feasible schedules for MPMCG such that the coverage guarantee is strictly less than $T^*_{\min}$. In a similar manner, feasible solutions of MPMRC are not necessarily feasible for MPMCG, since MPMRC allows negative residual capacities. In fact, from MPMRC to MPMCG and from MPMCG to MPMEC, the solution space is reduced, as a new constraint is added, and feasible solutions of MPMEC (respectively MPMCG) are included in the feasible set of MPMCG (respectively MPMRC).

3.4. Bounds

Bounds are proposed for MCG and MEC to evaluate the gap between the current objective value and the optimal one. When the gap is zero, the expensive last iteration of the column generation algorithm presented in Section 4 can be avoided, since the current solution is proven optimal.

3.4.1. Bounds for MCG

Let ${T_{\min}}^*$ be the maximal coverage guarantee. A trivial upper bound on ${T_{\min}}^*$ computes on each face the difference between the total energy of its candidate sensors and the coverage demand of the targets crossing it.

$$\text{UB1}^{\text{MCG}} = \min_{f \in F^*} \left\{ \sum_{i \in S(f)} E_i - \sum_{k \in K | f \in T(k)} \Delta_k \right\}$$

The upper bound UB1^{MCG} can be extended to set of faces. Let $\mathcal{F} \subseteq F^*$ be a subset of the faces overlapping the critical zone. Then UB2^{MCG} is computed in a similar way, using union of faces instead of a single one.

$$\text{UB2}^{\text{MCG}} = \min_{\mathcal{F} \subseteq F^*} \left\{ \sum_{i \in \bigcup_{f \in \mathcal{F}} S(f)} E_i - \sum_{k \in K | \mathcal{F} \cap T(k) \neq \emptyset} \Delta_k \right\}$$

In practice, computing UB2^{MCG} may be very expensive because it needs to enumerate all the subsets of faces.

3.4.2. Bounds for MEC

An upper bound for the objective of MEC, denoted by UB1^{MEC} is given by the difference between the cumulated lifetime of the sensors and the time horizon.

$$\text{UB1}^{\text{MEC}} = \sum_{i \in I} E_i - \sum_{k \in K} \Delta_k$$

This bound can be improved by considering α_k a lower bound on the minimal number of sensors to activate in each time window k.

$$\text{UB2}^{\text{MEC}} = \sum_{i \in I} E_i - \sum_{k \in K} \alpha_k \Delta_k$$

4. Solution approach

4.1. Column generation algorithm

The number of feasible covers for each subproblem is in $\mathcal{O}(|K|2^m)$. Enumerating all the covers is generally not practicable. Then column generation is a suitable and efficient method to solve linear programs when there is a very large number of columns and all of them cannot be considered explicitely. Let MP be any of the linear programs MPMRC, MPMCG and MPMEC. We denote by RMP, for *restricted master problem*, a model based on MP with a restricted number of covers. Its main idea is to exploit the fact that only a subset of the columns is needed to solve MP, because most of them will be non-basic in the optimal solution. The procedure starts with a restricted master problem RMP containing a small set of initial columns. When solved, RMP gives a dual solution (π, μ) and a lower bound z_{RMP} on MP. Then, the procedure is to iteratively add columns improving the objective function, until none can be found. At each iteration, a pricing problem PP is addressed. The purpose of PP is to find improving columns by maximizing their reduced cost. MP and PP are solved alternatively until no profitable column can be found, then the final solution of MP is proven optimal. In order to speed up the column generation, we also choose to use a metaheuristic called GRASP for solving PP. Such an approach where a metaheuristic is hybridized with a mathematical programming technique is called a matheuristic [44]. However, to prove optimality of the incumbent solution, the column generation algorithm needs either to solve PP optimally at least once or to obtain an objective value equal to an

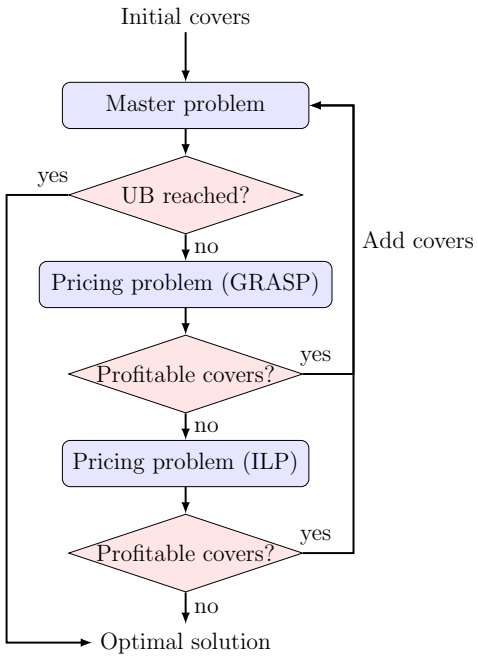

Figure 5: Column generation algorithm for each subproblem

upper bound. In order to avoid the use of ILP that is expensive in computation time, upper bounds presented in Section 3.4 are computed and compared to the current objective value. When the current objective value reaches an upper bound, solving PP is not necessary since optimality is proven by the upper bound. For MRC, the upper bound is zero since feasibility is guaranteed by finding a minimum residual capacity greater than or equal to zero. The upper bound used for MCG is UB1^{MCG}. UB2^{MCG} is computationnally expensive because it relies on the enumeration of all the subsets of faces. Finally, UB2^{MEC} is used for MEC, since it dominates UB1^{MEC}.

The column generation algorithm for each subproblem is an exact method illustrated in Figure 5. MPMRC, MPMCG and MPMEC are designed to be consecutively solved, so the final set of columns of the previous subproblem is the input for the current one. The three subproblems also share the same pricing problem.

4.2. Pricing problem

The pricing problem PP can be decomposed into $|K|$ independent subproblems PP_k, one for each time window $k \in K$. PP_k is formulated as a linear program defined by the equations (26)-(29). The π_i and μ_k coefficients are the optimal values of the dual variables of RMP. The binary decision variable a_{ic} is equal to 1 if sensor i is selected in the new cover c, 0 otherwise.

$$\text{PP}_k \qquad \max \; -\sum_{i \in I} \pi_i a_{ic} - \mu_k \tag{26}$$

s.t.

$$\sum_{i \in S(f)} a_{ic} \geq 1 \qquad \forall f \in T(k) \tag{27}$$

$$\sum_{i \in I} a_{ic} \leq |T(k)| \tag{28}$$

$$a_{ic} \in \{0, 1\} \qquad \forall i \in I \tag{29}$$

The objective (26) is to maximize the reduced cost of the new column. Constraint (27) ensures that each face of $T(k)$ is covered. To cover all the faces, a requirement is to select at least one sensor among the candidate sensors in $S(f)$ for each face f. Constraint (28) is a valid inequality that bounds the number of activated sensors. Indeed, the number of sensors required to cover $|T(k)|$ faces cannot exceed $|T(k)|$.

Since finding a positive reduced cost is sufficient, it is not always necessary to solve PP_k to optimality. Moreover, PP_k is equivalent to a set covering problem, that is NP-hard. The column generation procedure can also be accelerated by metaheuristics like GRASP [45]. Such an approach combining metaheuristics and mathematical programming is called a matheuristic [44]. This combination can be designed in two ways. The first approach is to use a mathematical programming technique either to improve or to design a metaheuristic. The second approach, used in this paper, is to use a metaheuristic to improve a mathematical programming technique [46].

GRASP (for Greedy Randomized Adaptive Search Procedure) builds a cover iteratively. At each iteration, the procedure computes a utility score $u_i = \frac{n_i}{\pi_i}$ for each sensor s_i, where n_i is the number of uncovered faces that the sensor i is able to cover. The utility score is designed to select in priority sensors that maximize the reduced cost and cover a larger number of faces. Using a parameter $\alpha \in [0, 1]$, we compute a threshold $u_{\text{limit}} = u_{\min} + \alpha(u_{\max} - u_{\min})$. All the sensors that have a utility greater than or equal to u_{limit} are put in the RCL (Restricted Candidate List). Then, one sensor is chosen from the RCL and added to the cover. GRASP performs new iterations until all the targets are covered. The procedure is complemented with a local search algorithm using a 2-flip neighborhood as suggested in [45]. First, the algorithm searches for sensors that can be removed without uncovering a target. Second, we process 1-1 exchanges and 2-1 exchanges that improve the reduced cost while keeping the cover feasible. The 1-1 exchange attempts to replace one sensor by another that has a lower cost. Similarly, the 2-1 exchange tries to replace two sensors by one while guaranteeing target coverage. The exchanges are processed as soon as they are found, without checking whether a better move exists in the whole neighborhood.

Due to the fact that there are many independent auxiliary subproblems and several methods to solve them, there are many possible strategies to generate the columns. Because GRASP returns good solutions much faster than an ILP solver, a smart strategy is to run it as much as possible before attempting to solve PP_k with an ILP solver.

To summarize, our method solves three subproblems (MRC, MCG and MEC) consecutively using column generation as illustrated in Figure 6. Since the pricing problem is the same for all the subproblems, the set of covers found for the previous subproblem is used as

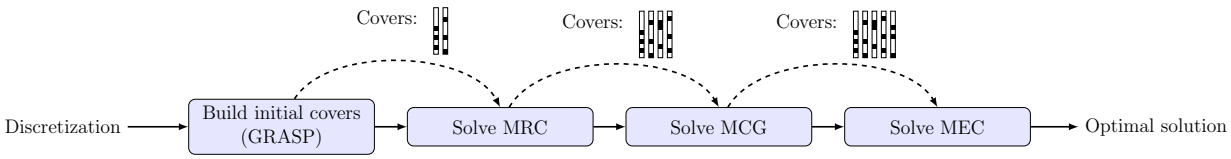

Figure 6: General algorithm

initial covers of the current one. However, Constraint (15) of the models MPMRC, MPMCG and MPMEC cannot be satisfied without building an initial set of covers. Since the dual variables (π, μ) are not provided at first, an initial set of covers is obtained by first solving the pricing problem with $\pi_i = 1, \forall i \in I$ and $\mu_k = -|T(k)| - 1, \forall k \in K$. This reduces the pricing problem to the so-called Set Covering Problem with unit costs. The objective is then to find covers with a minimum number of sensors, that are likely to be active in the optimal solution since they minimize energy consumption. GRASP is chosen as the method to compute covers of low cardinality in a short amount of time.

4.3. Dual bounds

Solving the pricing problems allows to compute additional bounds, thanks to the dual formulation of the master problem. The dual of MPMCG is denoted by DMPMCG, and defined by the equations (30)-(36).

DMPMCG

$$\min \sum_{i \in I} E_i \pi_i + \sum_{k \in K} \Delta_k \mu_k \tag{30}$$

s.t.

$$\pi_i - \sum_{f : i \in S(f)} \lambda_f \geq 0 \qquad \forall i \in I \quad [r_i] \tag{31}$$

$$\sum_{f \in F^*} \lambda_f \geq 1 \qquad [T_{\min}] \tag{32}$$

$$\sum_{i \in I} a_{ic} \pi_i + \mu_k \geq 0 \qquad \forall k \in K, c \in \mathcal{C}(k) \quad [d_c^k] \tag{33}$$

$$\pi_i \in \mathbb{R} \qquad \forall i \in I \tag{34}$$

$$\mu_k \in \mathbb{R} \qquad \forall k \in K \tag{35}$$

$$\lambda_f \leq 0 \qquad \forall f \in F^* \tag{36}$$

Now we consider the restricted linear programming master denoted by RMPMCG. It is made of a subset of the columns of MPMCG, and contains all the r_i and $T_{\min}$. The optimal dual variables of RMPMCG are denoted by $\widetilde{\pi}$, $\widetilde{\mu}$ and $\widetilde{\lambda}$. Hence, we define $|K|$ subproblems denoted by PP_k as:

$$\max_{c \in \mathcal{C}(k)} z_k = -\sum_{i \in I} \widetilde{\pi}_i a_{ic} - \widetilde{\mu}_k$$

Let z_k^* be the optimal objective value of PP_k. By definition, $z_k^* \geq -\sum_{i\in I} \widetilde{\pi}_i a_{ic} - \widetilde{\mu}_k$ for all $k \in K$, and for all $c \in \mathcal{C}(k)$. This can be written as $\sum_{i\in I} \widetilde{\pi}_i a_{ic} + (\widetilde{\mu}_k + z_k^*) > 0$ for all $k \in K$, and for all $c \in \mathcal{C}(k)$.

Then, setting:

$$\begin{bmatrix} \pi \\ \mu \\ \lambda \end{bmatrix} = \begin{bmatrix} \widetilde{\pi} \\ \widetilde{\mu} + z^* \\ \widetilde{\lambda} \end{bmatrix}$$

yields a feasible solution to DMPMCG. Indeed, the constraint (33) is satisfied by definition of $z^* = \left[z_1^* \dots z_{|K|}^*\right]^\top$, the constraints (31) and (32) are also satisfied by $\widetilde{\pi}$ and $\widetilde{\lambda}$ because all the r_i variables and $T_{\min}$ are present in RMPMCG (so their corresponding constraints are also present in the dual of RMPMCG).

By the weak duality property, the objective value of any feasible solution to DMPMCG is an upper bound on the optimal objective value of MPMCG, hence:

$$T_{\min} \leq \sum_{i\in I} E_i \widetilde{\pi}_i + \sum_{k\in K} \Delta_k (\widetilde{\mu}_k + z_k^*)$$

Now we define the dual of MPMEC, denoted by DMPMEC. For the sake of brevity, the maximization version of MPMEC has been considered to formulate DMPMEC, since the latter uses the same variables.

DMPMEC

$$\min \sum_{i\in I} E_i \pi_i + \sum_{k\in K} \Delta_k \mu_k + T_{\min}^* \sum_{f\in F^*} \lambda_f \tag{37}$$

s.t.

$$\pi_i - \sum_{f: i\in S(f)} \lambda_f \geq 1 \qquad \forall i \in I \quad [r_i] \tag{38}$$

(33) – (36)

Using the same reasoning for DMPMEC, we obtain:

$$\sum_{i\in I} r_i \leq \sum_{i\in I} E_i \widetilde{\pi}_i + \sum_{k\in K} \Delta_k (\widetilde{\mu}_k + z_k^*)$$

5. Computational results

We performed our experiments on a set of 100 randomly generated instances. Each instance describes the trajectories of n targets ($n \in \{5, 10, 15\}$) lying inside a 100×100 square region. A trajectory is represented as a piecewise linear curve composed of 4 segments joining 5 randomly chosen control points. The dates of passage associated to each control point are uniformly distributed along the time horizon $H = 100$ units of time. Thus, the first control point is at $t = 0$, the last control point is at $t = H$, and the speed of the targets is

constant between every consecutive pair of control points. Consequently, a target may cross the same face several times. A set of m sensors ($m \in \{50, 100, 150\}$) is randomly dispatched, ensuring that each sensor covers a part of at least one target trajectory. We set $R = 40$ for the sensing radius of the sensors. The values of the initial capacities lie randomly in the interval $[0, 100]$. The zone of interest considered is the set of faces traversed by the targets, so $F^* = F$. For each parameter set (m, n), 10 instances are generated, then we obtain 3×3 groups of 10 instances. A set of 10 additional instances with $m = 50$ sensors and $n = 30$ targets is added, making a total of 100 instances. We have also tested our method on instances with 200 sensors and more. However, when the number of sensors exceeds 200, the proposed approach cannot be used on our computer because of a lack of memory.

The algorithms have been implemented in C++ and executed on an Intel Xeon processor W3520 (2.67 GHz $\times$ 8) with 8 GBytes RAM under Linux (Ubuntu 14.04). The linear programs MPMRC, MPMCG, MPMEC are solved using IBM CPLEX 12.6.1. The pricing problems PP_k are solved approximately using a custom implementation of GRASP with a local search, and exactly using CPLEX. Several randomization parameters α of GRASP, lying in $\{0.6, 0.7, 0.8, 0.9, 1\}$, have been tested. The value α has no significant impact as shown in Table 2. The number of selected sensors is always less than or equal to the number of targets, and we observed that there are in average less than 4 sensors per cover in our instances. In the rest of the experiments, the value of α has been empirically set to 0.8. Each variant and each parameter set has been executed 10 times on each instance.

m	n	Avg. $\lvert K\rvert$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$	$\alpha = 1$
50	5	515	0.576 s	0.602 s	**0.566** s	0.572 s	0.589 s
	10	936	1.419 s	1.420 s	**1.411** s	1.449 s	1.478 s
	15	1304	2.455 s	2.456 s	**2.301** s	2.506 s	2.490 s
	30	1955	5.623 s	5.400 s	**5.189** s	5.483 s	5.699 s
100	5	1080	1.468 s	1.445 s	1.486 s	**1.378** s	1.428 s
	10	2100	4.259 s	**3.990** s	4.059 s	4.247 s	4.245 s
	15	3061	8.614 s	8.427 s	8.115 s	**8.058** s	8.003 s
150	5	1640	**2.793** s	3.059 s	3.023 s	2.944 s	2.930 s
	10	3260	10.05 s	**9.042** s	9.234 s	11.17 s	10.55 s
	15	4797	18.93 s	18.69 s	**17.96** s	19.11 s	19.10 s

Table 2: Average CPU time according to the value α of GRASP ($\delta = 0$)

First, we consider that the target positions are exactly known (uncertainty disc radius $\delta = 0$).

Table 3 compares the CPU times of the matheuristic approach (combining GRASP metaheuristic and CPLEX solver) denoted by GRASP+ILP, and the CPU times of the classical column generation, that solves each PP_k problem using CPLEX only (ILP). The third column contains the average number of time windows (after instance size reduction).

The matheuristic GRASP+ILP reduces the CPU time by a factor of 3 to 4 in average compared to the ILP approach. Moreover, the instances are more difficult to solve as their

m	n	Avg. $\lvert K\rvert$	GRASP+ILP	ILP
50	5	515	0.566 s	1.938 s
	10	936	1.411 s	4.749 s
	15	1304	2.301 s	7.926 s
	30	1955	5.189 s	16.30 s
100	5	1080	1.486 s	5.191 s
	10	2100	4.059 s	15.70 s
	15	3061	8.115 s	28.29 s
150	5	1640	3.023 s	11.52 s
	10	3260	9.234 s	29.20 s
	15	4797	17.96 s	68.88 s

Table 3: Average CPU time of GRASP+ILP and ILP according to the number of sensors

size increases, in particular when the number of time windows increases. Indeed, the number of pricing problems to solve at each iteration of the column generation is equal to the number of time windows.

m	n	$\delta = 0$	$\delta = 1$	$\delta = 2$
50	5	515	598	607
	10	936	1163	1210
	15	1304	1677	1760
	30	1955	2779	2938
100	5	1080	1295	1619
	10	2100	2687	3379
	15	3061	4072	5114
150	5	1640	2181	2769
	10	3260	4646	5787
	15	4797	7107	8781

Table 4: Average number $\lvert K\rvert$ of time windows according to the radius δ of the uncertainty disc

Considering spatial uncertainty modifies the number of time windows generated by discretization. Indeed, an uncertainty disc may cross more boundaries of sensing discs than a point. Table 4 shows the impact of the uncertainty disc on the number of time windows. The uncertainty radius also impacts the CPU times of the discretization algorithm, as shown in Table 5.

The instance reduction procedure plays a major role when target positions are uncertain. Table 6 shows the efficiency of this procedure according to the number of sensors, the number of targets, and the uncertainty disc radius. When $\delta = 0$, the number of deleted time windows increases significantly with the number of targets. Indeed, more targets increase the probability to obtain redundant faces to cover, i.e. faces such that the set of candidate sensors is a subset of the candidate sensors of another face. This offers more opportunity to fuse time windows.

m	n	$\delta = 0$	$\delta = 1$	$\delta = 2$
50	5	0.023 s	0.279 s	0.638 s
	10	0.076 s	1.080 s	2.557 s
	15	0.162 s	2.436 s	5.779 s
	30	0.623 s	9.899 s	23.54 s
100	5	0.089 s	2.814 s	7.960 s
	10	0.326 s	11.84 s	34.17 s
	15	0.718 s	27.58 s	79.31 s
150	5	0.200 s	11.72 s	37.50 s
	10	0.761 s	52.15 s	169.0 s
	15	1.691 s	124.6 s	396.2 s

Table 5: Average CPU time of discretization according to uncertainty disc radius δ

m	n	$\delta = 0$	$\delta = 1$	$\delta = 2$
50	5	7 %	46 %	45 %
	10	16 %	48 %	46 %
	15	22 %	50 %	47 %
	30	41 %	58 %	55 %
100	5	3 %	42 %	27 %
	10	7 %	41 %	25 %
	15	10 %	40 %	25 %
150	5	2 %	35 %	17 %
	10	4 %	32 %	14 %
	15	6 %	30 %	14 %

Table 6: Average percentage of deleted time windows after instance size reduction

When the uncertainty disc radius is equal to 1, about 30% to 60% of time windows are deleted in average. The reduction procedure performs better when we consider an uncertainty disc, as the initial number of time windows is significantly larger. However, this phenomenon is limited, as shown in the case where $\delta = 2$. Indeed, the discretization may produce less time windows for larger uncertainty radii. For instance, let's consider a particular case of one target following a sinusoidal trajectory along the boundary of a sensing disc of a sensor i. If the uncertainty radius is sufficiently small, the target crosses twice the boundary at each period. Consequently, each entrance and exit of the sensing disc produces a tick. If δ grows large enough, then the uncertainty disc covers the boundary all the time. Thus, the number of ticks is reduced, so the instance size reduction procedure may delete significantly less time windows. On our instances, the number of time windows is generally slightly lower with $\delta = 2$ than with $\delta = 1$.

m	n	$\delta = 0$	$\delta = 1$	$\delta = 2$
50	5	0.004 s	0.076 s	0.113 s
	10	0.048 s	0.546 s	0.784 s
	15	0.198 s	1.837 s	2.638 s
	30	2.164 s	20.76 s	28.45 s
100	5	0.011 s	0.440 s	0.669 s
	10	0.114 s	3.493 s	5.619 s
	15	0.459 s	16.44 s	21.66 s
150	5	0.019 s	1.204 s	2.306 s
	10	0.176 s	12.43 s	18.54 s
	15	0.686 s	46.66 s	57.72 s

Table 7: Average CPU time of the instance size reduction according to the uncertainty disc radius δ

As δ increases, more and more faces can be crossed by the uncertainty disc. The number of deleted faces in the $T(k)$ sets is significantly higher for $\delta = 2$ than for $\delta = 1$. However, the number of fused time windows is smaller for $\delta = 2$. Increasing the uncertainty disc radius can also decrease the efficiency of the instance size reduction procedure.

The impact of instance size reduction on CPU time is shown in Table 7. When the target position is subject to uncertainty, the CPU time of the reduction procedure can reach one minute in average. However, the reduction procedure is profitable most of the time, as can been seen by comparing the CPU times with Table 8, showing the CPU times of column generation with and without reduction procedure. The columns denoted by Wo.R. display the CPU times on non-reduced instances. The columns denoted by W.R. show the CPU times on reduced instances, plus the CPU time of the reduction procedure. The columns "Gain" show the percentage of CPU time saved thanks to the reduction procedure. For one case ($\delta = 2$, $m = 150$ sensors and $n = 15$ targets without reduction), the program has been aborted because the CPU time exceeded 3600 seconds. The most significant time savings are generally obtained with large values of δ and when the number of targets is greater than 15.

		$\delta = 0$			$\delta = 1$			$\delta = 2$		
m	n	Wo.R.	W.R.	Gain	Wo.R.	W.R.	Gain	Wo.R.	W.R.	Gain
50	5	0.645 s	0.570 s	12 %	1.303 s	0.693 s	47 %	1.457 s	0.806 s	45 %
	10	1.737 s	1.459 s	16 %	3.886 s	2.195 s	44 %	5.213 s	2.850 s	45 %
	15	3.364 s	2.499 s	26 %	7.821 s	4.936 s	37 %	10.88 s	6.354 s	42 %
	30	10.92 s	7.353 s	33 %	26.43 s	27.73 s	-5 %	38.40 s	37.14 s	3 %
100	5	1.463 s	1.497 s	1 %	3.882 s	2.339 s	40 %	6.763 s	3.500 s	48 %
	10	4.319 s	4.173 s	3 %	13.92 s	9.048 s	35 %	27.81 s	14.65 s	47 %
	15	9.212 s	8.574 s	7 %	37.46 s	28.91 s	23 %	68.04 s	40.95 s	40 %
150	5	3.099 s	3.042 s	2 %	10.84 s	5.813 s	46 %	19.55 s	9.453 s	52 %
	10	9.753 s	9.410 s	4 %	45.38 s	27.88 s	39 %	117.2 s	44.82 s	62 %
	15	18.93 s	18.65 s	2 %	139.0 s	81.32 s	41 %	N/A	114.0 s	N/A

Table 8: CPU time of column generation, with and without instance size reduction

		$\delta = 0$			$\delta = 1$			$\delta = 2$		
m	n	$UB1^{MCG}$	$UB1^{MEC}$	$UB2^{MEC}$	$UB1^{MCG}$	$UB1^{MEC}$	$UB2^{MEC}$	$UB1^{MCG}$	$UB1^{MEC}$	$UB2^{MEC}$
50	5	< 0.01 %	2.18 %	0.28 %	0 %	2.18 %	0.94 %	< 0.01 %	2.18 %	0.19 %
	10	0 %	3.67 %	0.39 %	0 %	3.67 %	1.21 %	< 0.01 %	3.67 %	0.29 %
	15	0 %	5.10 %	0.51 %	< 0.01 %	5.10 %	1.23 %	< 0.01 %	5.10 %	0.31 %
	30	0 %	7.78 %	0.85 %	0 %	7.78 %	1.46 %	0 %	7.78 %	0.33 %
100	5	0 %	0.90 %	0.11 %	0 %	0.90 %	0.15 %	0 %	0.90 %	0.02 %
	10	0 %	1.53 %	0.13 %	0 %	1.53 %	0.15 %	0 %	1.53 %	0.05 %
	15	0 %	2.23 %	0.22 %	0 %	2.23 %	0.20 %	0 %	2.23 %	0.06 %
150	5	0 %	0.54 %	0.06 %	0 %	0.54 %	0.01 %	0 %	0.54 %	0.01 %
	10	0 %	0.96 %	0.09 %	0 %	0.96 %	0.04 %	0 %	0.96 %	0.04 %
	15	0 %	1.40 %	0.15 %	0 %	1.40 %	0.10 %	0 %	1.40 %	0.08 %

Table 9: Average gap between the upper bounds and the optimal values

Table 9 shows the average gap in percents between the upper bounds and the optimal values of the objective functions of the MCG and MEC subproblems. The $UB2^{MCG}$ upper bound has not been implemented, since it requires a complete enumeration of all subsets of faces, which is very expensive in practice. The $UB2^{MEC}$ upper bound needs to solve (approximately or exactly) a maximum clique problem. For that, we use the LEMON library [47], and in particular the implementation of an iterated local search algorithm proposed by Grosso, Locatelli, and Pullan [48]. We recall that $UB1^{MCG}$ computes the differences between the faces potential (sum of residual capacities of candidate sensors) and the coverage demand of the crossing targets. $UB1^{MCG}$ reaches the optimal value for 94 % of the instances. When the optimal value is not met, the gap is less than 2 %. On any case, $UB2^{MEC}$ dominates $UB1^{MEC}$, since $UB2^{MEC}$ is an improved version of $UB1^{MEC}$. The upper bound $UB2^{MEC}$ performs very well thanks to the bounds on the maximum clique problem. These bounds, in particular for MEC, can be useful when the column generation algorithm cannot return an optimal solution in a reasonnable time. The decision maker can also stop the algorithm if solution quality is judged satisfactory.

m	n	MRC	MCG	MEC
50	5	11.2 %	30.4 %	58.4 %
	10	11.8 %	33.4 %	54.8 %
	15	12.8 %	29.5 %	57.7 %
	30	7.2 %	23.0 %	69.8 %
100	5	9.3 %	27.6 %	63.1 %
	10	7.5 %	27.6 %	64.9 %
	15	7.2 %	25.5 %	67.3 %
150	5	7.3 %	23.1 %	69.5 %
	10	6.3 %	25.1 %	68.6 %
	15	6.9 %	23.5 %	69.6 %

Table 10: Average of percentages of CPU time for solving each subproblem

Table 10 shows the average percentage of CPU time spent for each subproblem. We observe that the average time spent for MRC is less than 15 %. This is explained by the fact that MRC is stopped as soon as the objective value is greater than or equal to zero. More than half of the CPU time is spent for solving MEC. Moreover, since the upper bound $UB1^{MCG}$ reaches optimality in 94 % of the instances, for MCG the algorithm is more likely to be able to skip the ILP which is expensive in CPU time. The bounds for MEC being slightly less efficient, this results in more time spent solving the ILP formulation for MEC.

6. Conclusion

In this paper, we addressed a wireless sensor network optimization problem with coverage guarantee and target position uncertainty. The proposed method builds a schedule using a two-phase algorithm. The discretization phase transforms the input data into a scheduling problem instance. Target position uncertainty is handled during the discretization. The

scheduling problem is then decomposed into three subproblems solved using column generation. The three master problems are designed to share the same pricing problem and to allow column reuse from a master problem to the one of the next subproblem. As a consequence, the final set of columns of the previous suproblem is the input of the current one. A GRASP metaheuristic complemented with a local search is proposed to accelerate the column generation, the algorithm is then a matheuristic.

Numerical results show that the matheuristic approach reduces significantly the CPU time compared to a classical column generation. The CPU time is even more reduced by the instance size reduction procedure that performs best when the number of targets is large and when target positions are uncertain.

As an extension of the problem investigated in this paper, we suggest to take communication costs into account. Indeed, wireless sensor networks are typically connected to a base station in order to analyze the data collected by the sensors. Sending and routing the data can drain energy from the sensor batteries. An idea could be to modify the pricing problem to generate covers that include the communication paths.

Acknowledgements

We thank *Direction Générale de l'Armement* (DGA) for a financial support to this work.

References

References

[1] Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.. Wireless sensor networks: a survey. Computer Networks 2002;38(4):393–422. doi:10.1016/s1389-1286(01)00302-4.

[2] Yick, J., Mukherjee, B., Ghosal, D.. Wireless sensor network survey. Computer Networks 2008;52(12):2292–2330. doi:10.1016/j.comnet.2008.04.002.

[3] Zhang, W., Cao, G.. DCTC: Dynamic convoy tree-based collaboration for target tracking in sensor networks. IEEE Transactions on Wireless Communications 2004;3(5):1689–1701. doi:10.1109/twc.2004.833443.

[4] Handy, M., Haase, M., Timmermann, D.. Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In: 4th International Workshop on Mobile and Wireless Communications Network. IEEE; 2002, p. 368–372. doi:10.1109/mwcn.2002.1045790.

[5] Jindal, P., Gupta, V.. Study of energy efficient routing protocols of wireless sensor networks and their further researches: a survey. International Journal of Computer Science and Communication Engineering 2013;2:57–62.

[6] Younis, O., Fahmy, S.. HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. IEEE Transactions on Mobile Computing 2004;3(4):366–379. doi:10.1109/tmc.2004.41.

[7] Yang, H., Sikdar, B.. A protocol for tracking mobile targets using sensor networks. In: Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003. IEEE; 2003, p. 71–81. doi:10.1109/snpa.2003.1203358.

[8] Xu, Y., Winter, J., Lee, W.C.. Prediction-based strategies for energy saving in object tracking sensor networks. In: IEEE International Conference on Mobile Data Management, 2004. Proceedings. 2004. IEEE; 2004, p. 346–357. doi:10.1109/mdm.2004.1263084.

[9] Kung, H., Vlah, D.. Efficient location tracking using sensor networks. In: Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE; vol. 3. IEEE; 2003, p. 1954–1961. doi:10.1109/wcnc.2003.1200686.

[10] Xie, Y., Tang, G., Wang, D., Xiao, W., Tang, D., Tang, J.. A fault-tolerant target-tracking strategy based on unreliable sensing in wireless sensor networks. In: 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum. IEEE; 2012, p. 2116–2125. doi:10.1109/ipdpsw.2012.261.

[11] Laoudias, C., Michaelides, M.P., Panayiotou, C.G.. fttrack: Fault-tolerant target tracking in binary sensor networks. ACM Transactions on Sensor Networks 2014;10(4):1–28. doi:10.1145/2538509.

[12] Jin, Y., Ding, Y., Hao, K., Jin, Y.. An endocrine-based intelligent distributed cooperative algorithm for target tracking in wireless sensor networks. Soft Computing 2014;19(5):1427–1441. doi:10.1007/s00500-014-1352-3.

[13] Mannan, M., Rana, S.B.. Fault tolerance in wireless sensor network. International Journal of Current Engineering and Technology 2015;5(3):1785–1788.

[14] Oracevic, A., Ozdemir, S.. A survey of secure target tracking algorithms for wireless sensor networks. In: 2014 World Congress on Computer Applications and Information Systems (WCCAIS). IEEE; 2014,doi:10.1109/wccais.2014.6916628.

[15] Naderan, M., Dehghan, M., Pedram, H., Hakami, V.. Survey of mobile object tracking protocols in wireless sensor networks: a network-centric perspective. International Journal of Ad Hoc and Ubiquitous Computing 2012;11(1):34–63. doi:10.1504/ijahuc.2012.049283.

[16] Guo, W., Zhang, W.. A survey on intelligent routing protocols in wireless sensor networks. Journal of Network and Computer Applications 2014;38:185–201. doi:10.1016/j.jnca.2013.04.001.

[17] Rault, T., Bouabdallah, A., Challal, Y.. Energy efficiency in wireless sensor networks: A top-down survey. Computer Networks 2014;67:104–122. doi:10.1016/j.comnet.2014.03.027.

[18] Wang, C., Thai, M.T., Li, Y., Wang, F., Wu, W.. Optimization scheme for sensor coverage scheduling with bandwidth constraints. Optimization Letters 2008;3(1):63–75. doi:10.1007/s11590-008-0091-8.

[19] Rossi, A., Singh, A., Sevaux, M.. Column generation algorithm for sensor coverage scheduling under bandwidth constraints. Networks 2011;60(3):141–154. doi:10.1002/net.20466.

[20] Carrabs, F., Cerulli, R., D'Ambrosio, C., Gentili, M., Raiconi, A.. Maximizing lifetime in wireless sensor networks with multiple sensor families. Computers & Operations Research 2015;60:121–137. doi:10.1016/j.cor.2015.02.013.

[21] Castaño, F., Bourreau, E., Velasco, N., Rossi, A., Sevaux, M.. Exact approaches for lifetime maximization in connectivity constrained wireless multi-role sensor networks. European Journal of Operational Research 2015;241(1):28–38. doi:10.1016/j.ejor.2014.08.013.

[22] Carrabs, F., Cerulli, R., D'Ambrosio, C., Raiconi, A.. An exact algorithm to extend lifetime through roles allocation in sensor networks with connectivity constraints. Optimization Letters 2016;doi:10.1007/s11590-016-1072-y.

[23] Carrabs, F., Cerulli, R., D'Ambrosio, C., Raiconi, A.. Extending lifetime through partial coverage and roles allocation in connectivity-constrained sensor networks. IFAC-PapersOnLine 2016;49(12):973–978. doi:10.1016/j.ifacol.2016.07.902.

[24] Singh, A., Rossi, A.. Group scheduling problems in directional sensor networks. Engineering Optimization 2014;47(12):1651–1669. doi:10.1080/0305215x.2014.982633.

[25] Lu, Z., Li, W.W., Pan, M.. Maximum lifetime scheduling for target coverage and data collection in wireless sensor networks. IEEE Transactions on Vehicular Technology 2015;64(2):714–727. doi:10.1109/tvt.2014.2322356.

[26] Liu, H., Chu, X., Leung, Y.W., Jia, X., Wan, P.J.. General maximal lifetime sensor-target surveillance problem and its solution. IEEE Transactions on Parallel and Distributed Systems 2011;22(10):1757–1765. doi:10.1109/tpds.2011.42.

[27] Rossi, A., Sevaux, M., Singh, A., Geiger, M.J.. On the cover scheduling problem in wireless sensor networks. In: Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2011, p. 657–668. doi:10.1007/978-3-642-21527-8_73.

[28] Gopinadh, V., Singh, A.. Swarm intelligence approaches for cover scheduling problem in wireless sensor networks. International Journal of Bio-Inspired Computation 2015;7(1):50. doi:10.1504/ijbic.2015.067987.

[29] Lee, C.T., Lin, F.Y.S., Wen, Y.F.. An efficient object tracking algorithm in wireless sensor networks. In: Proceedings of the 9th Joint Conference on Information Sciences (JCIS). Atlantis Press. ISBN 978-90-78677-01-7; 2006,doi:10.2991/jcis.2006.207.

[30] Yeong-Sung, F., Cheng-Ta, , Hsu, Y.Y.. An energy-efficient algorithm for object tracking in wireless sensor networks. In: 2010 IEEE International Conference on Wireless Communications, Networking and Information Security. IEEE; 2010, p. 424–430. doi:10.1109/wcins.2010.5544123.

[31] Atia, G.K., Veeravalli, V.V., Fuemmeler, J.A.. Sensor scheduling for energy-efficient target tracking in sensor networks. IEEE Transactions on Signal Processing 2011;59(10):4923–4937.

[32] Shi, K., Chen, H., Lin, Y.. Probabilistic coverage based sensor scheduling for target tracking sensor networks. Information Sciences 2015;292:95–110. doi:10.1016/j.ins.2014.08.067.

[33] Chen, J., Cao, K., Li, K., Sun, Y.. Distributed sensor activation algorithm for target tracking with binary sensor networks. Cluster Computing 2009;14(1):55–64. doi:10.1007/s10586-009-0092-0.

[34] Ellison, R.J., Fisher, D.A., Linger, R.C., Lipson, H.F., Longstaff, T.. Survivable network systems: An emerging discipline. Tech. Rep.; DTIC Document; 1997.

[35] Wang, L., Xiao, Y.. A survey of energy-efficient scheduling mechanisms in sensor networks. Mobile Networks and Applications 2006;11(5):723–740. doi:10.1007/s11036-006-7798-5.

[36] Wang, Y.S., Lin, F.Y.S., Chan, C.H., Wang, J.W.. Maximization of wireless mesh networks survivability to assure service continuity under intelligent attacks. In: 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA). IEEE; 2013, p. 583–590. doi:10.1109/aina.2013.31.

[37] Pannetier, B., Dezert, J., Sella, G.. Multiple target tracking with wireless sensor network for ground battlefield surveillance. In: FUSION 2014. SALAMANQUE, Spain; 2014,URL: https://hal-onera.archives-ouvertes.fr/hal-01070361.

[38] Lersteau, C., Rossi, A., Sevaux, M.. Robust scheduling of wireless sensor networks for target tracking under uncertainty. European Journal of Operational Research 2016;252(2):407–417. doi:10.1016/j.ejor.2016.01.018.

[39] Billaut, J.C., Moukrim, A., Sanlaville, E.. Flexibility and Robustness in Scheduling. ISTE-Wiley publishing,London; 2008. ISBN 978-1-84821-054-7. URL: https://hal.archives-ouvertes.fr/hal-00445442.

[40] Lersteau, C., Sevaux, M., Rossi, A., Cerulli, R., Raiconi, A.. Maximization of residual capacities for target tracking in wireless sensor networks. In: International Symposium on Combinatorial Optimization. 2016,.

[41] Ning, S.. Estimation of area of uncertainty for moving target tracking. In: Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC). IEEE; 2013,doi:10.1109/mec.2013.6885196.

[42] Berman, P., Calinescu, G., Shah, C., Zelikovsky, A.. Power efficient monitoring management in sensor networks. In: 2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733); vol. 4. IEEE; 2004, p. 2329–2334. doi:10.1109/wcnc.2004.1311452.

[43] Slijepcevic, S., Potkonjak, M.. Power efficient organization of wireless sensor networks. In: ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240); vol. 2. IEEE; 2001, p. 472–476. doi:10.1109/icc.2001.936985.

[44] Boschetti, M.A., Maniezzo, V., Roffilli, M., Röhler, A.B.. Matheuristics: Optimization, simulation and control. In: Hybrid Metaheuristics. Springer Berlin Heidelberg; 2009, p. 171–177. doi:10.1007/978-3-642-04918-7_13.

[45] Feo, T.A., Resende, M.G.C.. Greedy randomized adaptive search procedures. Journal of Global Optimization 1995;6(2):109–133. doi:10.1007/bf01096763.

[46] Singh, A., Rossi, A., Sevaux, M.. Matheuristic approaches forQ-coverage problem versions in wireless sensor networks. Engineering Optimization 2013;45(5):609–626. doi:10.1080/0305215x.2012.687732.

[47] Dezső, B., Jüttner, A., Kovács, P.. LEMON – an open source c++ graph template library. Electronic Notes in Theoretical Computer Science 2011;264(5):23–45. doi:10.1016/j.entcs.2011.06.003.

[48] Grosso, A., Locatelli, M., Pullan, W.. Simple ingredients leading to very efficient heuristics for the maximum clique problem. Journal of Heuristics 2007;14(6):587–612. doi:10.1007/s10732-007-9055-x.

Appendix A. Nonlinear model for MCG

We propose a mathematical formulation for MCG, denoted by NLMPMCG, and defined by equations (A.1)-(A.10). The decision variables are as follows:

- d_c^k is the activity duration of the cover $c \in \mathcal{C}(k)$ during the time window $k \in K$
- r_i is the residual capacity of sensor $i \in I$
- $T_{\min}$ is the duration of the coverage guarantee
- x_{ic}^k is equal to 1 if and only if sensor $i \in I$ is part of cover $c \in \mathcal{C}(k)$ in time window $k \in K$

$$\max T_{\min} \tag{A.1}$$

$$\sum_{k \in K} \sum_{c \in \mathcal{C}(k)} x_{ic}^k d_c^k + r_i = E_i \qquad \forall i \in I \tag{A.2}$$

$$\sum_{c \in \mathcal{C}(k)} d_c^k = \Delta_k \qquad \forall k \in K \tag{A.3}$$

$$\sum_{i \in S(f)} r_i \geq T_{\min} \qquad \forall f \in F^* \tag{A.4}$$

$$\sum_{i \in S(f)} x_{ic}^k \geq 1 \qquad \forall k \in K, \forall c \in \mathcal{C}(k), \forall f \in T(k) \tag{A.5}$$

$$d_c^k \leq d_{c+1}^k \qquad \forall k \in K, \forall c \in \mathcal{C}(k) \setminus \{|\mathcal{C}(k)|\} \tag{A.6}$$

$$T_{\min} \geq 0 \tag{A.7}$$

$$r_i \geq 0 \qquad \forall i \in I \tag{A.8}$$

$$x_{ic}^k \in \{0, 1\} \qquad \forall i \in I, \forall k \in K, \forall c \in \mathcal{C}(k) \tag{A.9}$$

$$d_c^k \geq 0 \qquad \forall k \in K, c \in \mathcal{C}(k) \tag{A.10}$$

Constraint (A.2) enforces that the battery capacity of each sensor is satisfied. Constraint (A.3) enforces that the monitoring duration of each time window is equal to Δ_k. Constraint (A.4) defines $T_{\min}$ as the coverage guarantee. Constraint (A.5) ensures that at least one candidate sensor is selected for monitoring each face. Constraint (A.6) is a symmetry breaking inequality that may help the solver to converge faster.

Appendix B. MILP model for MCG

Because of constraint (A.5), model NLMPMCG is nonlinear. So we introduce u_{ic}^k as a new continuous variable for replacing $x_{ic}^k d_c^k$. The model IMPMCG, defined by the equations (B.1)-(B.14), is then a mixed integer linear program.

$$\max T_{\min} \tag{B.1}$$

$$\sum_{k\in K}\sum_{c\in\mathcal{C}(k)} u_{ic}^k + r_i = E_i \qquad \forall i \in I \tag{B.2}$$

$$\sum_{c\in\mathcal{C}(k)} d_c^k = \Delta_k \qquad \forall k \in K \tag{B.3}$$

$$\sum_{i\in S(f)} r_i \geq T_{\min} \qquad \forall f \in F^* \tag{B.4}$$

$$\sum_{i\in S(f)} x_{ic}^k \geq 1 \qquad \forall k \in K, \forall c \in \mathcal{C}(k), \forall f \in T(k) \tag{B.5}$$

$$d_c^k \leq d_{c+1}^k \qquad \forall k \in K, \forall c \in \mathcal{C}(k) \setminus \{|\mathcal{C}(k)|\} \tag{B.6}$$

$$u_{ic}^k \leq d_c^k \qquad \forall k \in K, \forall i \in \bigcup_{f\in T(k)} S(f), \forall c \in \mathcal{C}(k) \tag{B.7}$$

$$u_{ic}^k \leq E_i x_{ic}^k \qquad \forall k \in K, \forall i \in \bigcup_{f\in T(k)} S(f), \forall c \in \mathcal{C}(k) \tag{B.8}$$

$$u_{ic}^k \geq \Delta_k(x_{ic}^k - 1) + d_c^k \qquad \forall k \in K, \forall i \in \bigcup_{f\in T(k)} S(f), \forall c \in \mathcal{C}(k) \tag{B.9}$$

$$T_{\min} \geq 0 \tag{B.10}$$

$$r_i \geq 0 \qquad \forall i \in I \tag{B.11}$$

$$u_{ic}^k \geq 0 \qquad \forall i \in I, \forall k \in K, \forall c \in \mathcal{C}(k) \tag{B.12}$$

$$x_{ic}^k \in \{0,1\} \qquad \forall i \in I, \forall k \in K, \forall c \in \mathcal{C}(k) \tag{B.13}$$

$$d_c^k \geq 0 \qquad \forall k \in K, c \in \mathcal{C}(k) \tag{B.14}$$

Constraint (B.2) is an updated version of constraint (A.2). Constraints (B.7) to (B.9) are added for enforcing that u_{ic}^k is equal to $x_{ic}^k d_c^k$.

Appendix C. Computational results

Table C.11 compares the CPU time of solving IMPMCG and GRASP+ILP, which is the most efficient column generation-based approach proposed in this paper. The comparison is performed on small instances, where the number of sensors ranges from $m = 20$ to $m = 40$. A time limit of 3600 seconds has been set for each instance. The CPU time is significantly higher for the IMPMCG approach. Moreover, the fact that the time limit was reached on several small instances shows the unstability of the IMPMCG approach.

In order to assess the computational effort required by solving IMPMCG, 30 instances have been generated in the way described in Section 5, except that the instances have $m \in \{20, 30, 40\}$ sensors and $n = 4$ targets. These 30 instances are smaller than the minimum size instances used in Section 5, but as can be seen in Table C.12, IMPMCG cannot be

	$m = 20$		$m = 30$		$m = 40$	
Instance	IMPMCG	CG	IMPMCG	CG	IMPMCG	CG
1	1.27 s	0.13 s	3.96 s	0.21 s	$>$ 3600 s	0.29 s
2	1.28 s	0.09 s	4.34 s	0.15 s	10.0 s	0.24 s
3	$>$ 3600 s	0.14 s	$>$ 3600 s	0.08 s	$>$ 3600 s	0.18 s
4	0.86 s	0.11 s	2.50 s	0.18 s	$>$ 3600 s	0.24 s
5	0.92 s	0.07 s	$>$ 3600 s	0.14 s	59.8 s	0.19 s
6	0.49 s	0.10 s	$>$ 3600 s	0.27 s	5.72 s	0.23 s
7	0.80 s	0.09 s	$>$ 3600 s	0.19 s	8.15 s	0.29 s
8	0.68 s	0.11 s	3.17 s	0.19 s	9.69 s	0.27 s
9	1.06 s	0.09 s	3.37 s	0.12 s	$>$ 3600 s	0.27 s
10	0.92 s	0.15 s	203 s	0.19 s	$>$ 3600 s	0.27 s

Table C.11: Average CPU time of IMPMCG and GRASP+ILP on small instances

Number of sensors	Number of instances solved to optimality (out of 10)
$m = 20$	9
$m = 30$	6
$m = 40$	5

Table C.12: Number of instances solved to optimality in less than one hour with IMPMCG

solved to optimality for all these instances, whereas the column generation-based approaches introduced in this paper solves them all in less than one second. The number of instances requiring more than one hour to solve is quickly increasing with instance size. As in [19], this clearly shows the efficiency of column generation-based approaches compared to IMPMCG.