Robust scheduling of wireless sensor networks for target tracking under uncertainty

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

An object tracking sensor network (OTSN) is a wireless sensor network designed to track moving objects in its sensing area. It is made of static sensors deployed in a region for tracking moving targets. Usually, these sensors are equipped of a sensing unit and a non-rechargeable battery. The investigated mission involves a moving target with a known trajectory, such as a train on a railway or a plane in an airline route. In order to save energy, the target must be monitored by exactly one sensor at any time. In our context, the sensors may be not accessible during the mission and the target can be subject to earliness or tardiness. Therefore, our aim is to build a static schedule of sensing activities that resists to these perturbations. A pseudo-polynomial two-step algorithm is proposed. First, a discretization step processes the input data, and a mathematical formulation of the scheduling problem is proposed. Then, a dichotomy approach that solves a transportation problem at every iteration is introduced; the very last step is addressed by solving a linear program.

Keywords: wireless sensor networks, target tracking, uncertainty, stability radius, robustness

*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)

1. Introduction

1.1. Context

Since wireless sensors are becoming more and more affordable, more and more applications are now possible such as traffic control or battlefield surveillance [1, 2]. Low-cost sensors are usually autonomous, equipped with a sensing unit and a battery. Their typical purpose is to track targets in their sensing range. They can be randomly deployed from an airplane or an helicopter in places lacking monitoring infrastructures. Sensors relying on technologies like drones and radars are suitable in military or humanitarian assistance contexts, where the infrastructures are destroyed or non-existent. In this paper, the investigated mission is to monitor a target with a known trajectory, such as a train on a railway, a vehicle on a road or a plane in an airline route. Since accessing sensors can be difficult in some environments, we may have no control on them during the mission. Then, in order to save battery lifetime, sensors can be switched off and waken up later. To minimize the energy consumption, the target is monitored by only one sensor at a time. Moreover, the target is subject to perturbations on its path, that may cause advances and delays. Consequently, our challenge is to find a *static* schedule of *sensing activities*, able to monitor the target at any time, without target loss despite perturbation. A *target loss* happens when the target is outside the range of any active sensor. A *sensing activity* is identified by a sensor, a starting date and a duration, to be computed offline, before the mission. During an *activity*, the corresponding sensor wakes up, collects information about the target for a certain amount of time, and then gets back to sleep status. Our aim is to find the most *robust* schedule, i.e. the one that resists to the largest possible earliness and tardiness.

1.2. Related work

There are plenty of WSN protocols for target tracking proposed in the literature, designed to achieve one or more goals. Usually, these protocols are dedicated to the optimization or management of different criteria. We present below a non-exhaustive list of the criteria addressed by those protocols:

- *Energy consumption*: this is one of the most critical aspects since the sensors generally have a non-rechargeable battery. For exemple, the framework designed in [3] configures min-cost convoy trees using dynamic programming in order to save energy. Many protocols that focus on this aspect are based on LEACH [4, 5] or HEED [6].

- *Tracking precision*: can be achieved by selecting more sensors or by predicting the target location. A good precision technique can help deciding which sensors to wake up and make a better use of the energy. [7] and [8] propose protocols based on prediction.

- *Scalability*: a WSN protocol should scale to different network sizes since a dense network can significantly increase the communication cost. Scalable protocols typically use cluster-based or distributed approaches instead of centralized ones. The authors in [9] focus particularly on this aspect.

- *Fault tolerance*: target tracking may fail due to deficiencies or environmental events. This aspect has witnessed a growing interest recently [10–14].

For a more exhaustive review on the criteria and the WSN protocols, the reader is referred to [15]. A classification of target tracking algorithms from the security point of view is proposed in [14]. The problem investigated in this paper is related to track continuity. However, while WSN protocols generally assume that the target trajectory is random, our approach is based on a known trajectory. It could also be combined with a trajectory prediction method as in [16], to be adapted to targets following a random trajectory, by periodically sending the prediction results as an input of our method. This procedure is also suggested by Demigha et al. [17].

Only a few of these protocols are currently implemented using optimization techniques. The survey by Naderan et al. [15] states that only one protocol, designed by Lee et al. [18] and later extended by Yeong-Sung et al. [19], is effectively using optimization techniques. This protocol configures an object tracking tree using a Lagrangian relaxation-based heuristic algorithm based on a 0/1 linear formulation. Their problem only handles communication mission and is frequency-based, i.e. frequencies of target movements from a sensor to another are supposed to be known.

In [20], the problem is to optimize the tradeoff between tracking performance and energy consumption. A scheduling problem is stated as a partially observable Markov decision process. The decision is to choose the set of sensors to activate at each time slot.

A distributed sensor activation algorithm DSA^2 that relies on binary sensors is designed in [21]. The algorithm activates the sensors according to probabilities to detect targets. A robustness study is provided by changing

parameters, such as the maximum velocity of the targets, the sensing range or the sensor density.

One of the most studied WSN problems in the field of optimization is the network lifetime maximization. Assuming that the targets are static, the aim is to select and schedule a sequence of subsets of sensors, in order to maximize the time during which all the targets are covered. Many variants of this problem have been investigated, such as MNLB (Maximizing Network Lifetime under Bandwidth constraints) and MCBB (Minimizing Coverage Breach under Bandwidth constraints), solved using heuristics [22] and further using column generation [23]. Column generation is also a flagship technique to solve network lifetime maximization problems. Carrabs et al. [24] handle heterogeneous networks and speed up the column generation using a genetic algorithm. Castaño et al. [25] take into account communication and multi-roles sensors. To solve the pricing problem, they propose two approaches: constraint programming and Branch-and-Cut based on Benders' decomposition. Singh and Rossi [26] study some ways to schedule groups of active sensors after obtaining an optimal solution and propose a greedy heuristic and a genetic algorithm.

When the energy consumption of a sensor is variable, i.e. proportional to the number of monitored targets, the network lifetime maximization problem becomes polynomially solvable. Liu et al. [27] provide a continous linear formulation under this assumption.

In most of the research papers on WSNs, the notion of robustness is reduced to *survivability*, i.e. the ability to resist to unexpected failures such as enemy attacks or sensor deficiencies [28–30]. This paper focuses on the ability of a sensor schedule to resist to target behavior perturbations, in order to reduce the risk of target loss. A prediction scheme proposed in [31] also aims at maintaining track continuity in ground battlefield surveillance, but supposes that the targets can move on and off a road.

Our previous study [32] proposes an exact approach to solve the minimization of energy consumption and the network lifetime maximization problems, whereas this paper addresses robustness issue of this problem.

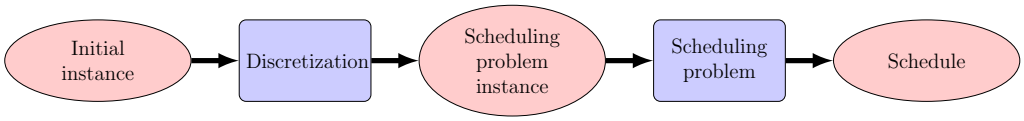

Figure 1: Overview of the steps of the method

For the sake of readability, the problem investigated in this paper is introduced step by step. Section 2 presents a preliminary step, called discretization, to transform the input data into a scheduling problem instance. Such a transformation is necessary to introduce the definition of stability radius, i.e. the measure of robustness, in Section 3. A diagram summarizes the different steps in Figure 1. In Section 4, some upper bounds on the stability radius are provided. Sections 5 and 6 respectively describe the proposed approach to solve the problem and the results of its implementation. Section 7 concludes the paper.

2. Preliminary concepts

2.1. Definitions

A set of m *sensors* $I = \{1, \ldots, m\}$ is randomly deployed in a two-dimensional region in order to monitor a single *moving target*. The positions of the sensors are known and static. Each sensor is able to monitor the target under its sensing range (disc of radius R). The action of monitoring is called a sensing *activity*. An activity consumes energy from sensor i, therefore the total activity duration cannot exceed its battery lifetime E_i, for all $i \in I$.

Without loss of generality, the planned target position is supposed to be exactly known at any time $t \in [0, H]$ where H is the monitoring horizon and defined by a continuous two-dimensional vector function $\mathcal{T}(t)$.

$$\mathcal{T} : t \mapsto (x, y) \text{ where } t \in [0, H], (x, y) \in \mathbb{R}^2$$

Table 1 describes the initial input data of the problem.

I	Set of sensors $\{1, \ldots, m\}$
(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
$\mathcal{T}(t)$	Trajectory of the target for all $t \in [0, H]$

Table 1: Initial problem input

The goal of the problem is to find a robust schedule of sensing activities in order to avoid target loss. The requirements are the following:

- The target must be covered at any time in $[0, H]$.

- The target must be covered by exactly one sensor at a time.

Geometric data undergo a preprocessing called *discretization* before stating *scheduling problem* as a combinatorial optimization problem. The notion of robustness used in this problem is presented in Section 3.

2.2. Discretization

First, the monitored space is partitioned into zones called *faces*. Since the sensing area of each sensor is assumed to be a disk of radius R, the monitored area can be seen as a *planar graph* [33, 34] (Figure 2). The vertices are the intersections between the boundaries of sensors' disks. The edges connect vertices along the boundaries. The different surfaces bounded by the edges are the faces. In fact, all the points inside a given face are covered by the same set of sensors, and the number of faces is at most $m(m-1)+2$, as shown in [33]. We define a *face* by a unique set of covering sensors. For example, in Figure 2, the geometric faces 7 and $7'$ are covered by the same set of sensors $\{s_2\}$, so they are considered as only one face numbered 7. So, multiple faces that have exactly the same set of candidate sensors are considered the same.

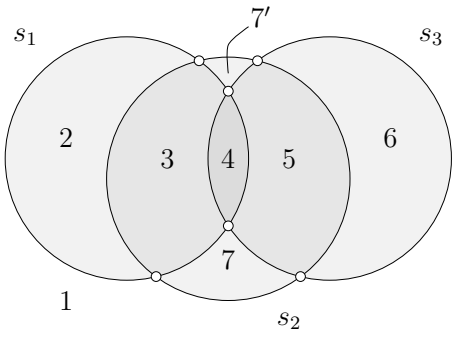

Figure 2: A planar graph based on 3 sensors with 7 faces

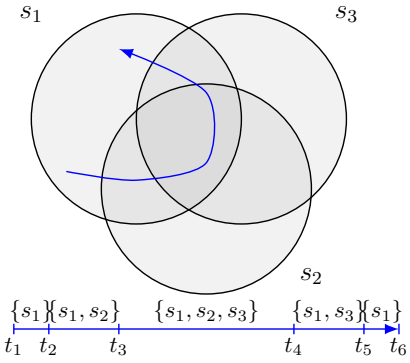

Figure 3: Sets of candidate sensors along the target trajectory

Let F be the set of all faces and $\hat{F} \subseteq F$ be the subset of faces in which the target is moving. The target can be monitored by any candidate sensor covering the face where it is located. Hence, the target trajectory can be seen as a sequence $(f_1, f_2, \ldots, f_p) \in \hat{F}^p$ of the traversed faces. When the target leaves a face and enters into another one, the set of candidate sensors changes (Figure 3).

A *tick*, denoted by t_k, is a date of transition from a face f_i to f_{i+1} and a *time window* the period between two consecutive ticks. The ticks are obtained by computing the intersections between the trajectory and the boundaries of the faces. Thus, the actual target trajectory shape is no longer needed since the time windows and the faces are sufficient to track the target. We denote by $K = \{1, \ldots, p\}$ the set of time windows, t_k the date of the k^{th} tick, and $S(k) \subseteq I$ the set of candidate sensors able to monitor the target during time window k, i.e. for all t in the range $[t_k, t_{k+1}]$. The duration of time window k is denoted by Δ_k and defined by $t_{k+1} - t_k$. Outside $[0, H]$, there is no restriction and no tracking requirement. Therefore, before t_1 and after t_{p+1}, the set of candidate sensors is I, i.e. $S(0) = S(p+1) = I$. Such a convention will be convenient in the sequel, because it avoids handling the ticks t_1 and t_{p+1} as particular cases.

The ticks are partitioned in two classes: the *entering* and the *leaving* ticks. The moment when the target enters the sensing area of a sensor is called an entering tick. Whereas if it leaves a sensing area, it is a leaving tick. Consequently, tick t_k is an entering tick if $S(k-1) \subset S(k)$, and a leaving tick if $S(k) \subset S(k-1)$. Whenever the target enters a sensing area and simultaneously leaves another one, two distinct ticks, one leaving and one entering, are associated to the same date. This case can happen when the target passes through the intersection of two sensing area boundaries. It can be seen in Figure 3 that t_2, t_3 and t_6 are entering ticks (they are shown as opening square brackets in Figure 4), whereas t_1, t_4 and t_5 are leaving ticks (shown as closing square brackets). t_1 is considered as a leaving tick, since by convention $S(0) = I$, then $S(1) \subset S(0)$. On the other hand, t_6 is an entering tick, because $S(p+1) = I$ and $S(p) \subset S(p+1)$.

These definitions allow to introduce *availability intervals*. A sensor is said to be *available* at time t if the target is expected to be inside its sensing range. So, an availability interval of a sensor is a period of time during which the sensor is continuously able to monitor the target. Such an interval typically begins with an entering tick, and ends with a leaving tick, except for those intervals that start at time t_1 or end at time t_{p+1}. Since we cannot select more than one sensor at a time, a sensor cannot be selected outside its availability intervals, since its selection would cause a target loss. Figure 4 shows the availability intervals of the sensors for the instance of Figure 3. In general, a sensor may be associated to more than a single availability interval.

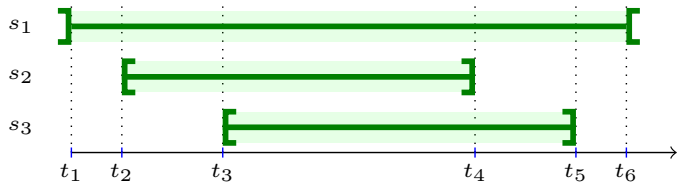

Figure 4: Availability intervals for the instance of Fig. 3

3. Scheduling problem

3.1. Notation

The aim of the scheduling problem is to build a *schedule* $\mathcal{S}$, composed by a set of sensing *activities*, denoted by $\mathcal{A}$. Each activity $a \in \mathcal{A}$ consists in activating a given sensor i at time s_a during d_a units of time. A schedule $\mathcal{S}$ is feasible if:

- All the time horizon $[0, H]$ is covered by activites without overlap.
- Each sensor is activated only inside its availability intervals.
- Each sensor is not used more than its capacity.

I	Set of sensors $\{1, \ldots, m\}$
E_i	Battery lifetime of the sensor $i \in I$
K	Set of time windows $\{1, \ldots, p\}$
t_k	Starting time of time window $k \in K$
Δ_k	Duration of time window $k \in K$
$S(k)$	Set of candidate sensors covering the target during the time window k, i.e. between the ticks t_k and t_{k+1}, $k \in K$

Table 2: Scheduling problem input

Table 2 summarizes the input for the scheduling problem.

The notion of robustness is very popular in optimization, and in particular in scheduling [35]. But since this notion is very rich and may have different meanings according to the problem and the context, it is introduced in details in Section 3.2.

The purpose of the scheduling problem is to find a schedule that remains feasible despite target earliness or tardiness. In order to achieve this goal, the notion of robustness used in this work is introduced in the next section. Then, two upper bounds on the measure of robustness are proposed in Section 4.

3.2. Robustness definition

We assume that the target cannot move outside its path, like a train on a railway. However, the arrival dates at given positions are subject to uncertainty. A tick t_k is the expected arrival date at the position $\mathcal{T}(t_k)$. In other words, at time t_k, the target is supposed to be at the boundary of two faces, denoted by f_{k-1} and f_k. Face f_{k-1} is covered by the sensors in $S(k-1)$ and f_k is covered by $S(k)$. If the target arrives ρ units of time late in $\mathcal{T}(t_k)$, then it leaves later f_{k-1}. Therefore, to avoid target loss, a sensor of $S(k-1) \cap S(k)$ should be active during the interval $[t_k, t_k + \rho]$. Symmetrically, if the target is early in $\mathcal{T}(t_k)$ by ρ units of time, then it should be monitored by a sensor of $S(k-1) \cap S(k)$ during the interval $[t_k - \rho, t_k]$. Consequently, in order to cope with both earliness and tardiness, sensors in $S(k-1) \cap S(k)$ should be selected to cover the target during the interval $[t_k - \rho, t_k + \rho]$.

Earliness can also occur at the beginning of the mission. If the target appears at time $t_1 - \rho$, then the schedule should begin also at the same date. By symmetry, the schedule should end later than t_{p+1} to anticipate possible lateness occuring at the end of the mission.

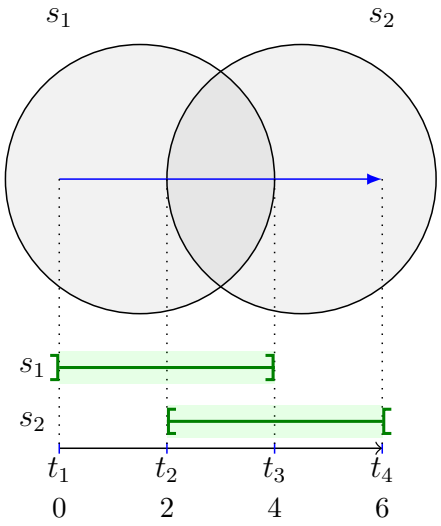

Figure 5: A simple instance with 2 sensors

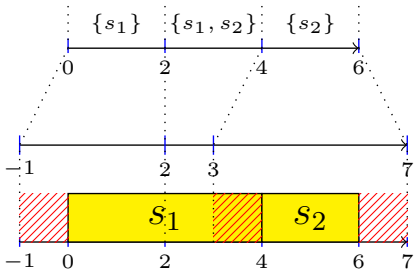

Figure 6: Example of non-robust schedule

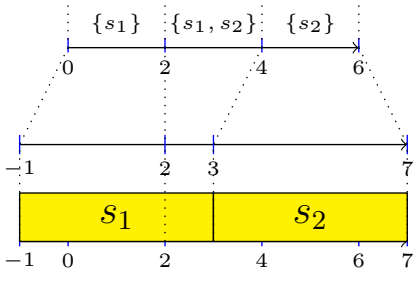

Figure 7: Example of robust schedule

Earliness and tardiness may create discrepancies between expected and actual ticks. Given a schedule $\mathcal{S}$, the aim in this section is to compute the greatest variation of the ticks t_k that do not compromize the feasibility of $\mathcal{S}$. This measure, denoted by ρ and introduced in [36, 37], is called *stability radius*. A schedule with a stability radius of ρ is able to cover the target whenever it is early or late of at most ρ units of time.

In our context, the stability radius is defined by the minimal duration between the instant a sensor is active and the instant it is no longer a candidate. Thus, it is limited by the duration between the boundaries of an activity and the boundaries of its corresponding availability interval.

Figure 5 presents a simple example of the problem with 2 sensors. Here, we assume that the battery lifetime of the sensors is unlimited. Figures 6 and 7 show two valid schedules, respectively denoted by $\mathcal{S}_1$ and $\mathcal{S}_2$, submitted to a perturbated scenario (shown as the second timeline on each figure). Schedule $\mathcal{S}_1$ activates sensor s_1 in time interval $[0, 4]$ and sensor s_2 in time interval $[4, 6]$, whereas $\mathcal{S}_2$ activates sensor s_1 in $[-1, 3]$ and sensor s_2 in $[3, 7]$. Both schedules are valid since they are able to cover the target if no perturbation occurs.

In our perturbated scenario, the target appears earlier and disappears 1 unit of time after the expected completion time of the mission. We can see that schedule $\mathcal{S}_1$ is not designed to resist to these perturbations since it begins at $t = 0$ and ends at $t = H$. Moreover, the target arrives 1 unit of time early in $\mathcal{T}(t_3)$. Since s_1 does not cover the face after t_3, $\mathcal{S}_1$ fails to cover the target in this case. However, schedule $\mathcal{S}_2$ resists to all of these perturbations and has a stability radius of 1, whereas $\mathcal{S}_1$ has a stability radius of 0.

In this paper, our aim is to find a schedule that maximizes the stability radius for the target tracking problem. The description of the solution approach is structured as follows. Section 4 provides upper bounds for the scheduling problem. Section 5 describes the decision version of the scheduling problem based on a Transportation Problem. Finally, the upper bounds are used for a dichotomy on the decision problem described in Section 5.1.

4. Bounds

We propose two kinds of upper bounds on the stability radius that will be used in the solution approach in Section 5. They rely on the idea of expanding the length of the time windows until inevitable exhaustion of the sensors. The first one is based on the distances between pairs of time windows

using the set of candidate sensors they have in common. The second one is computed by considering the energy limitation of the candidate sensors.

4.1. Upper bound based on ticks distances

The following upper bound, denoted by UB1, is obtained by computing the closest distance between time windows having no candidate sensors in common.

$$\text{UB1} = \min_{k,k' \in K} \left\{ \frac{1}{2} \left(t_{k'} - t_{k+1} \right) \mid k < k' \text{ and } S(k) \cap S(k') = \emptyset \right\}$$

If two time windows k and k' have no candidate sensors in common, then there exists no sensor able to cover simultaneously the two faces corresponding to these time windows. From our requirements, only one sensor can be active at a time. At each instant, either we select a sensor in $S(k)$, or we select one in $S(k')$, and only one of the two faces can be covered. Therefore, the expansion of the length of both time windows is limited by the half-distance between them (Figure 8).

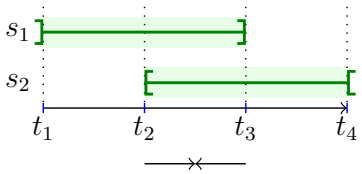

Figure 8: Illustration of UB1

This upper bound can be generalized for all pairs of time windows. In fact, when two time windows have some candidate sensors in common, the stability radius is limited by the sum of their battery lifetime. The generalization of UB1, denoted by UB1′, relies on this idea.

$$\text{UB1}' = \min_{k,k' \in K} \left\{ \frac{1}{2} \left(\sum_{i \in S(k) \cap S(k')} E_i + t_{k'} - t_{k+1} \right) \mid k < k' \right\}$$

When two time windows k and k' have no candidate sensors in common, then $\sum_{i \in S(k) \cap S(k')} E_i = 0$, and $UB1'$ reduces to $UB1$.

4.2. Upper bound based on energy limitation

Each time window is associated a face, and also a set of candidate sensors. Then, the extra duration of stay of the target inside the face cannot exceed the sum of the battery lifetimes of its candidate sensors. This idea is used to design the upper bound UB2.

$$\text{UB2} = \min_{k \in K} \left\{ \frac{1}{2} \left(\sum_{i \in S(k)} E_i - (t_{k+1} - t_k) \right) \right\}$$

The generalization of this upper bound is based on the fact that a given face may be associated with different time windows. This situation typically arises when the target reenters a face. It is recalled that $\hat{F}$ is the set of traversed faces. For each face $f \in \hat{F}$, let S_f be the set of sensors that cover it and ρ_f the upper bound on the stability radius that will be deduced from this face. Initially, ρ_f is set to zero. First, we compute $K_f = \{k \in K \mid S(k) \subseteq S_f\}$ as the set of all the time windows that are completely dependent on the sensors of S_f. We compute the vector $\mathcal{D}$ as the distance between two consecutive time windows in K_f, hence $\mathcal{D}$ has $|K_f| - 1$ elements. We suppose that the elements of $\mathcal{D}$ are sorted by increasing order. All the sensors of S_f are used to cover $|K_f|$ time windows, which means that the stability radius is at most the time during which the sensors can be used after covering all the time windows of K_f divided by $2\,|K_f|$.

In order to illustrate this idea, let us consider that a face is visited three times by the target. Time windows k, k' and k'' are the elements of K_f. These time windows are shown in Figure 9. We also assume that two sensors s_1 and s_2 cover f, with $E_1 = 5$ and $E_2 = 5$. It can be seen in the figure that covering three time windows require $1 + 2 + 1 = 4$ units of time. The residual energy of s_1 and s_2 is then $E_1 + E_2 - 4 = 6$. Then, since $|K_f| = 3$, the stability radius is necessarily less than $\rho_f = \frac{6}{2|K_f|} = 1$.

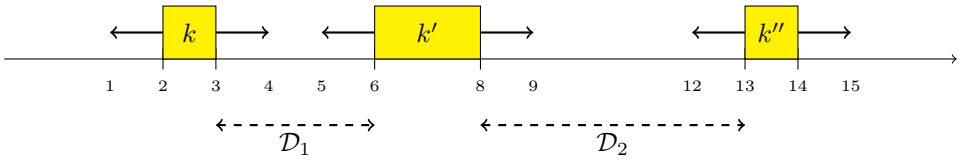

Figure 9: Illustration of UB2′, where $\rho_f = 1$ and $\sum_{i \in S_f} E_i = 10$

However, this figure shows that if the energy of the sensors in $|S_f|$ gets larger, then the stability radius can reach 1.5. In that case, the time windows k and k' become adjacent, i.e. if $\rho_f \geq 1.5$, then it is no longer necessary to increase it in 6 directions, as only 4 directions are necessary. Figure 10 illustrates the case where $\sum_{i \in S_f} E_i = 15$.

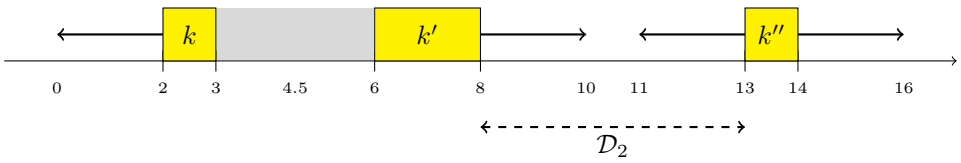

Figure 10: Illustration of UB2′, where $\rho_f = 2$ and $\sum_{i \in S_f} E_i = 15$

Finally, if $E_1 + E_2$ is large enough, ρ_f becomes larger than 2.5, and the time windows k' and k'' become adjacent. It is no longer necessary to increase it in 4 directions, as only 2 directions are necessary (Figure 11).

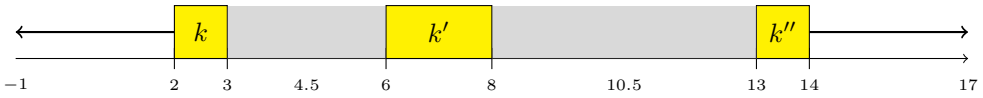

Figure 11: Illustration of UB2′, where $\rho_f = 3$ and $\sum_{i \in S_f} E_i = 18$

Consequently, the calculation of ρ_f is performed stepwise. Let $k_1 < k_2 < \cdots < k_{|K_f|}$ be the time windows in K_f. For all $q \in \{1, \ldots, |K_f|-1\}$, we define $\mathcal{D}_{k_q} = t_{k_{q+1}} - t_{k_q}$ as the distance between two consecutive time windows in K_f. Then, those $|K_f| - 1$ distances are re-indexed by increasing order ($\mathcal{D}_1 \leq \cdots \leq \mathcal{D}_{|K_f|-1}$). Algorithm 1 returns the tightest upper bound on the stability radius. For each face, it computes r, the residual energy after monitoring the time windows in K_f. Whenever the residual energy to monitor the target between the closest time windows is sufficiently large, these time windows are merged by removing the corresponding distance $\min \mathcal{D}$. Then, the number of directions of expansion $2(|\mathcal{D}|+1)$ is also decreased. The algorithm stops when the maximal expansion at each time window extremity, $\frac{r}{2(|\mathcal{D}|+1)}$, is less than or equal to the half-distance between the closest consecutive time windows $\min \mathcal{D}$.

Algorithm 1 Compute UB2′

$\rho_{\min} \leftarrow \infty$
for all $f \in \hat{F}$ **do**
  $r \leftarrow \sum_{i \in S_f} E_i - \sum_{k \in K_f} (t_{k+1} - t_k)$
  while $|\mathcal{D}| > 0$ and $\frac{r}{2(|\mathcal{D}|+1)} > \frac{\min \mathcal{D}}{2}$ **do**
    $r \leftarrow r - \min \mathcal{D}$
    $\mathcal{D} \leftarrow \mathcal{D} \setminus \{\min \mathcal{D}\}$
  end while
  $\rho_{\min} \leftarrow \min \left\{ \rho_{\min}, \frac{r}{2(|\mathcal{D}|+1)} \right\}$
end for
return $\rho_{\min}$

Lemma 1. UB1′ *and* UB2′ *have no performance guarantee on the stability radius* ρ*.*

Proof. There exists an instance for which $\rho = 0$, UB1′ > 0 and UB2′ > 0. Such an instance is given in the appendix. □

5. Solution approach

In this section, we introduce the decision problem associated to the scheduling problem. Afterwards, we exploit the decision problem to solve efficiently the scheduling problem.

5.1. Decision problem

Let ρ be an arbitrary positive value. The *decision problem* consists in answering the question: Does there exist a feasible schedule such that its stability radius is at least ρ? This problem can be reduced to the well-studied *Transportation Problem* (*TP*), that can be solved in polynomial time using a method based on the Kuhn's combinatorial algorithm for the assignment problem [38].

First of all, we consider the case where $\rho = 0$. An instance $\mathcal{I}_0$ of *TP* is built as follows: sensors are the suppliers having capacity E_i and time windows are the customers having demand Δ_k. The decision variables are x_{ik}, the amount of time during which sensor i monitors the target during time window k. To obtain a balanced transportation problem, the total demand

should be equal to the total supply. We set $\Delta_{p+1} = \sum_{i \in I} E_i - \sum_{k \in K} \Delta_k$, and the variables x_{ip+1} are the residual capacities of the sensors. The transportation costs c_{ik} are the penalties on the use of infeasible activities, i.e. activities based on sensor i to cover some time window k such that $i \notin S(k)$. For all $i \in I$, $c_{ip+1} = 0$.

$$\forall i \in I, k \in K : c_{ik} = \begin{cases} 0 & \text{if } i \in S(k) \\ 1 & \text{if } i \notin S(k) \end{cases}$$

Then the objective is to minimize the total time during which infeasible activities are used to cover the target. The problem can be stated as follows:

$$\min \sum_{i \in I} \sum_{k \in K \cup \{p+1\}} c_{ik} x_{ik} \tag{1}$$

$$\sum_{k \in K \cup \{p+1\}} x_{ik} = E_i \qquad \forall i \in I \tag{2}$$

$$\sum_{i \in I} x_{ik} = \Delta_k \qquad \forall k \in K \cup \{p+1\} \tag{3}$$

$$x_{ik} \geq 0 \qquad \forall i \in I, k \in K \cup \{p+1\} \tag{4}$$

Constraint (2) ensures that the total activity of a sensor does not exceed its battery lifetime. Constraint (3) imposes that the target is monitored during all the time windows.

First of all, there exists a feasible schedule if and only if the optimal objective value is zero. The transportation problem provides also a feasible schedule. It can be built using the x_{ik} variables. These variables act as budgets spent for sensing activities. For each strictly positive value of x_{ik}, we create an activity involving the sensor i during the time window k, of duration x_{ik}. Then, in each time window, the activities are sequenced in an arbitrary order, without overlap.

In order to decide the existence of a feasible schedule for a given $\rho > 0$, we generate a new instance $\mathcal{I}_\rho$. The value of ρ must be such that any feasible solution for $\mathcal{I}_\rho$ provides a feasible solution for the original problem such that the stability radius is greater than or equal to ρ. Since the stability radius is bounded by the durations between activities and the boundaries of their corresponding availability intervals, the procedure consists in restricting the availability intervals of the sensors to forbid any activity that would cause

Algorithm 2 BuildInstance

Require: $\mathcal{I}_0 = (\{t_k\}, \{S(k)\})$: the original instance
Require: ρ: the guaranteed stability radius of $\mathcal{I}_\rho$

procedure BUILDINSTANCE($\mathcal{I}_0$,ρ)

$\forall k \in K, \quad \sigma_k \leftarrow \begin{cases} -1 & \text{if } t_k \text{ is a leaving tick} \\ 1 & \text{if } t_k \text{ is an entering tick} \end{cases}$

$\forall k \in K \cup \{p+1\}, \quad t'_k \leftarrow t_k$

$\forall k \in K, \quad S'(k) \leftarrow S(k)$

$\delta \leftarrow \rho$

while $\delta > 0$ **do**

$\Delta_{\min} \leftarrow \min \left\{ \delta, \ \min_{k \in K} \dfrac{t'_{k+1} - t'_k}{2} \mid \sigma_{k+1} - \sigma_k < 0 \right\}$

▷ Find the minimal half-distance between two opposite ticks

for all $k \in K \cup \{p+1\}$ **do**

$t'_k \leftarrow t'_k + \sigma \times \Delta_{\min}$ ▷ Translate all the ticks

end for

for all $k \in K$ **do**

▷ Swap leaving and entering ticks meeting at the same date

if $\sigma_{k+1} - \sigma_k < 0$ **and** $t'_{k+1} - t'_k \leq 0$ **then**

$\sigma_k \leftarrow -\sigma_k$

$\sigma_{k+1} \leftarrow -\sigma_{k+1}$

$S'(k) \leftarrow S'(k-1) \cap S'(k+1)$

end if

end for

$\delta \leftarrow \delta - \Delta_{\min}$

end while

return $\mathcal{I}_\rho = (\{t'_k\}, \{S'(k)\})$ ▷ New instance $\mathcal{I}_\rho$ with t'_k and $S'(k)$

end procedure

the stability radius to be less that ρ. In Section 2.2, we considered two classes of ticks: *entering* and *leaving*. The entering ticks are translated by ρ and the leaving ones by $-\rho$. As soon as an entering tick t_k and a leaving tick t_{k+1} meet together, their position in the ordered list of ticks are exchanged and the set of candidate sensors $S(k)$ is updated as follows:

$$S(k) \leftarrow S(k-1) \cap S(k+1)$$

The procedure is summarized in Algorithm 2. If there exists $k \in K$ such that $S(k) = \emptyset$ and $\Delta_k > 0$, then the current instance is infeasible.

Figure 12 shows an example with 3 sensors, where the target trajectory is represented by the straight line oriented from left to right. The instance $\mathcal{I}_{1.5}$ can be built in two steps. First, the entering ticks are moved by $+1$ and the leaving ticks by -1 (Figure 13). At this point, ticks t_3 and t_4 meet together. Consequently, they are swapped, and the sensor s_3 is not a candidate anymore between these ticks. This is characterized in Figure 14 by the 2 open square brackets connections to s_3. Note that this is not anymore an interval but rather a forbidden interval for s_3. Next, the same procedure is applied by moving the ticks by 0.5. Hence, the schedule shown in Figure 14 achieves a stability radius of 1.5.

5.2. Scheduling problem

The scheduling problem is to find a schedule that maximizes the stability radius ρ. Algorithm 3 addresses this problem with dichotomy. It starts by testing the existence of a feasible schedule for $\rho = 0$ and $\rho = UB$, where $UB = \min\{\text{UB1}', \text{UB2}'\}$. If there exists a feasible schedule for $\rho = UB$, then the problem is solved. If there does not exist a feasible schedule for $\rho = 0$, then the problem is infeasible. Otherwise, our algorithm computes an ordered list $\mathcal{D}$ containing all the positive distances $t_{k'} - t_k < UB$ where t_k is an entering tick and $t_{k'}$ is a leaving tick. The elements of $\mathcal{D}$ are indexed by increasing order. Then, the algorithm finds the maximal value $\mathcal{D}_\ell$ for which there exists a feasible schedule with a stability radius of $\rho \geq \mathcal{D}_\ell$ using the dichotomy method. At each iteration, the decision problem is solved using the transportation problem formulation (SolveTP). The bounds on ρ, $\mathcal{D}_{\ell_{\min}}$ and $\mathcal{D}_{\ell_{\max}}$, are updated according to the fact that the transportation problem has a zero optimal value or not. Then, since $\rho \in [\mathcal{D}_{\ell_{\min}}, \mathcal{D}_{\ell_{\max}})$, a linear program is solved to build an optimal solution. The variables of the linear program are the following:

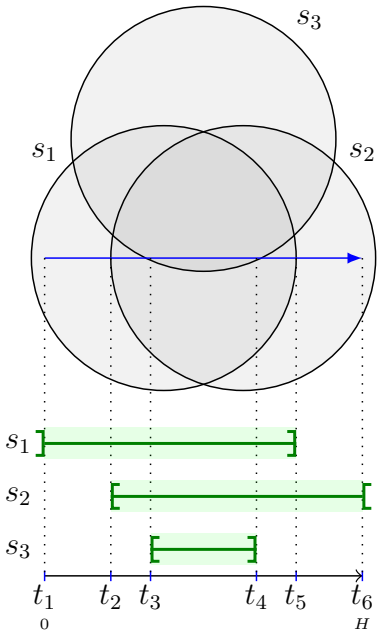

Figure 12: Instance $\mathcal{I}_0$

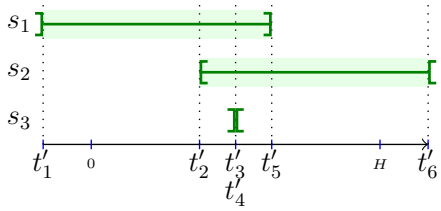

Figure 13: Instance $\mathcal{I}_1$

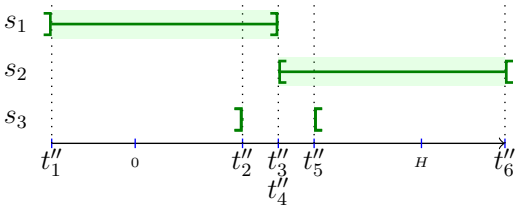

Figure 14: Instance $\mathcal{I}_{1.5}$

$\delta \geq 0$ Improvement of ρ over $\mathcal{D}_{\ell_{\min}}$

$x_{ik} \geq 0$ Amout of sensing activity allocated to the sensor $i \in I$ for the time window $k \in K$

The objective is to maximize the improvement δ. The LP model is described by:

$$\max \delta \tag{5}$$

$$\sum_{k \in K | i \in S(k)} x_{ik} \leq E_i \qquad \forall i \in I \tag{6}$$

$$\sum_{i \in S(k)} x_{ik} = \Delta_k + (\sigma_{k+1} - \sigma_k)\delta \qquad \forall k \in K \tag{7}$$

$$\delta \geq 0 \tag{8}$$

$$x_{ik} \geq 0 \qquad \forall k \in K, i \in S(k) \tag{9}$$

Where $\sigma_k = \begin{cases} 1 & \text{if } t_k \text{ is an entering tick} \\ -1 & \text{if } t_k \text{ is a leaving tick} \end{cases}$.

Constraint (6) limits the total activity duration of a sensor by its battery

lifetime. Constraint (7) allocates a sufficient amount of sensing time between the candidate sensors to monitor the target during each time window.

Once the solution is obtained, the schedule is built as shown in Section 5.1. In the final schedule, the total amount of energy spent by the sensors is $H + 2\rho$, since it begins ρ units of time before t_1 and ends ρ units of time after t_{p+1}. In the sequel, we define 3 lemmas that state the finiteness of ρ and the complexity status of the problem.

Lemma 2. *ρ is finite if and only if $\bigcap_{k\in K} S(k) = \emptyset$ or $\forall i \in \bigcap_{k\in K} S(k)$, E_i is finite.*

Proof. First of all, if $\bigcap_{k\in K} S(k) = \emptyset$, then the target crosses at least two faces with no sensors in common. Therefore, there exists no set of sensors able to cover the whole trajectory and ρ must be finite. If for all i in $\cap_{k\in K} S(k)$, E_i is finite, then the stability radius is at most UB1$' \leq \frac{1}{2}\left(\sum_{i\in\cap_{k\in K} S(k)} E_i + H\right)$. Conversely, if ρ is finite, then any sensor able to cover the whole trajectory, if such a sensor exists, has a finite capacity. □

Lemma 3. *The scheduling problem is solvable in a pseudo-polynomial number of operations, in $\mathcal{O}(mp(m+p)\log p)$.*

Proof. The discretization algorithm, running in polynomial time, generates p time windows. The existence of a feasible schedule, stated as a Transportation Problem in Section 5.1, can be reduced to an equivalent Max Flow Problem by building a graph with vertices associated to sensors and time windows, and by comparing the flow value to the time horizon. The Max Flow Problem is solvable in $\mathcal{O}(|V||E|)$ [39], where $|V| = \mathcal{O}(m+p)$ is the number of vertices and $|E| = \mathcal{O}(mp)$ is the number of arcs. Therefore, since the dichotomy processes a logarithmic number of iterations, the complexity is in $\mathcal{O}(mp(m+p)\log p)$. □

An example in which the number of ticks is as large as desired is shown in the appendix.

Lemma 4. *If the trajectory is a piecewise linear curve composed of q straight segments, then the problem is solvable in a polynomial number of operations, in $\mathcal{O}(q^2m^3\log(qm))$.*

Algorithm 3 StabilityRadius

$UB \leftarrow$ ComputeUpperBound
if $UB < 0$ **then**
  return -1 ▷ Infeasible problem
end if
$z \leftarrow$ SolveTP($\mathcal{I}_{UB}$) ▷ Does a schedule such that $\rho = UB$ exist?
if $z = 0$ **then**
  return $\rho = UB$
end if
$z \leftarrow$ SolveTP($\mathcal{I}_0$) ▷ Does a feasible schedule exist?
if $z > 0$ **then**
  return -1 ▷ Infeasible problem
end if
$\mathcal{D} \leftarrow \{t_{k'} - t_k \in [0, UB) \mid t_k \text{ is entering and } t_{k'} \text{ is leaving}\}$
$\ell_{\min} \leftarrow 1$
$\ell_{\max} \leftarrow |\mathcal{D}|$
while $\ell_{\max} - \ell_{\min} > 0$ **do**
  $\ell \leftarrow \left\lfloor \frac{\ell_{\max} - \ell_{\min}}{2} \right\rfloor$
  $\rho \leftarrow \mathcal{D}_\ell$
  $z \leftarrow$ SolveTP($\mathcal{I}_\rho$)
  if $z = 0$ **then**
    $\ell_{\min} \leftarrow \ell$
  else
    $\ell_{\max} \leftarrow \ell$
  end if
end while
$\rho \leftarrow \mathcal{D}_{\ell_{\min}}$
$\delta \leftarrow$ SolveLP($\mathcal{I}_\rho$)
$\rho \leftarrow \rho + \delta$
return ρ

Control point	1	2	3	4	5
Date	0	2	5	8	11
Coordinates	$(-7,-2)$	$(-4,-3)$	$(0,-2)$	$(-1,2)$	$(-3,4)$

Table 3: Dates and coordinates of control points defining the piecewise linear trajectory

Tick	t_1	t_2	t_3	t_4	t_5	t_6
Date	0	1.026	3.195	7.216	9.685	11
Status*	L	E	E	L	L	E
Entering face	f_1	f_2	f_3	f_4	f_1	-

*E = entering tick, L = leaving tick

Table 4: Result of the discretization algorithm

Proof. The proof is based on the geometry resulting in the intersection of segments and discs. Each segment can cross at most twice each sensors circle. Consequently, the number of ticks is at most $2qm$, so the number of time windows is at most $p = 2qm - 1$. The decision problem, stated as an equivalent Max Flow Problem, can be solved in $\mathcal{O}(|V||E|)$ [39], where $|V| = \mathcal{O}(qm)$ is the number of vertices and $|E| = \mathcal{O}(qm^2)$ is the number of arcs. Since the dichotomy processes a logarithmic number of iterations, the complexity is in $\mathcal{O}(q^2m^3 \log(qm))$. □

5.3. Example

Let us process a simple example similar to the instance shown on Figure 3. There are 3 sensors $\{s_1, s_2, s_3\}$ having respective coordinates $(-3, 0)$, $(0, -5)$, $(3, 0)$. Each sensor has a sensing radius $R = 6$ and an initial capacity $E_i = 15$. The target trajectory is a piecewise linear curve that passes through 5 control points described in Table 3. The speed of the target is uniform along each segment, but is subject to change after passing a control point.

Because the trajectory is composed of segments, the discretization step is performed by solving quadratic equations representing intersections between segments and circles.

The target passes through the faces $f_1 = \{s_1\}$, $f_2 = \{s_1, s_2\}$, $f_3 = \{s_1, s_2, s_3\}$ and $f_4 = \{s_1, s_3\}$. The set of ticks and the sequence of faces obtained are given in the Table 4. The two upper bounds UB1′ and UB2′

ρ	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5
0	1.026	2.169	4.021	2.469	1.315
2.010	5.047	2.169	0	2.469	5.336
3.095	7.216	0	2.169	0.300	7.505
3.165	7.216	0.140	2.169	0.160	7.644

Table 5: Values of Δ_k according to the value of ρ

are computed as described in Section 4.

$$\text{UB1}' = \min\{7.5, 8.584, 10.595, 11.829, 15, 9.51, 10.745, 15, 8.734, 7.5\} = 7.5$$

$$\text{UB2}' = \min\{3.165, 9.5, 17, 9.5\} = 3.165$$

Thus, an upper bound on the stability radius is $UB = 3.165$. The first step is to test whether there exists a feasible instance such that $\rho = UB$. Solving the transportation problem associated to the instance $\mathcal{I}_{3.165}$ gives a strictly positive value, so the optimal value of ρ is guaranteed to be strictly less than 3.165.

The second step is to test the feasibility for $\rho = 0$. Since $\mathcal{I}_0$ is feasible, the dichotomy algorithm has to be applied. The ordered set of distances $t'_k - t_k < UB$ is $\mathcal{D} = \{0, 2.01, 3.095\}$. Table 5 gives the values of Δ_k associated to the different instances according to the value of ρ.

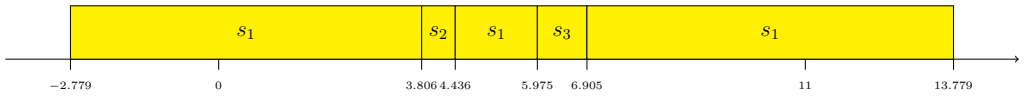

Figure 15: Gantt chart of the solution schedule

The dichotomy processes for the values $\rho = 2.01$ that gives a feasible instance, and $\rho = 3.095$ that gives an infeasible instance. Then, the optimal value of ρ belongs to the interval $[2.01, 3.095)$. Finally, the optimal objective value of the LP defined by equations (5)-(9) is $\delta = 0.769$. Thus, the optimal value of ρ is 2.779. The obtained schedule is shown in Figure 15. We observe that the stability radius is limited specifically by the capacity of the sensor s_1 which is totally consumed. The total energy spent by the sensors is $H + 2\rho = 16.558$ units of energy.

m	p	$E_i = 12$	$E_i = 16$	$E_i = 20$
100	374.26	9/2/30/9	44/0/4/2	46/1/2/1
200	629.68	3/1/26/20	32/2/11/5	45/1/2/2
500	1466.12	0/0/6/44	12/0/26/12	38/0/10/2
1000	2708.54	0/0/8/42	11/0/23/16	29/0/18/3

Table 6: Repartition between instances ($\#\,\mathrm{UB1}/\#\,\mathrm{UB2}/\#\emptyset/\#Inf$) according to the number of sensors and the initial sensor capacity

6. Computational results

The proposed approach is tested on a set of 200 randomly generated instances. The instance generator randomly generates a target path lying inside a $(\sqrt{10m} \times \sqrt{10m})$ square region. This path is a piecewise linear curve composed of 9 segments joining 10 randomly chosen control points. To each control point is associated a date that is the date of passage of the target. The dates are uniformly dispatched along the time horizon. As the control points are completely random, the target can come back in the same face several times. Then, it randomly dispatches a set of m sensors ($m \in \{100, 200, 500, 1000\}$) such that each sensor is able to cover a part of the trajectory. Any part of the trajectory can also be covered by several sensors. The fixed parameters are $R = 10$ (sensing radius) and $H = 10m$ (time horizon). Several values of initial sensor capacity ($E_i \in \{12, 16, 20\}$) have been tested for each instance.

The implementation has been coded in C++ and executed on an Xeon processor W3520 (2.67 GHz $\times$ 8) with 8 GBytes RAM under Linux (Ubuntu 14.04). The linear program defined by equations (5) to (9) is solved using IBM CPLEX 12.6.1. The decision problem is solved with the LEMON library version 1.3.1 from COIN-OR [40], after being reformulated as a maximum flow problem.

Table 6 presents results for the 4 classes of 50 instances grouped by number of sensors ($m \in \{100, 200, 500, 1000\}$). The second column p is the average number of ticks. Each line contains a quadruplet $\#\,\mathrm{UB1}/\#\,\mathrm{UB2}/\#\emptyset/\#Inf$ where $\#\,\mathrm{UB1}$ (respectively $\#\,\mathrm{UB2}$) is the number of instances for which $\mathrm{UB1}'$ (respectively $\mathrm{UB2}'$) is reached, $n_\emptyset$ is the number of instances for which none of the upper bounds is reached, and $\#Inf$ is the number of infeasible instances.

The upper bound $\mathrm{UB1}'$ dominates $\mathrm{UB2}'$ in 93.4% of the feasible instances.

m	$\min\{\text{UB1}', \text{UB2}'\}$	$\emptyset$
100	0.086 s	0.481 s
200	0.215 s	1.206 s
500	1.311 s	8.652 s
1000	5.252 s	33.55 s

Table 7: Average computation times of feasible instances when UB1′ or UB2′ is met, versus when none of these bounds is met

UB2′ seems to be more efficient when the initial capacity of the sensors is critical, i.e. the capacity is just enough to ensure the feasibility of the instance. Thus, the initial capacity of the sensors is a significant factor in the computation time.

When one of the upper bounds is reached, the calculation of the upper bounds represents for 45.8% of the total computation time in average. The generation of the instances $\mathcal{I}_\rho$ in Algorithm 3 runs for 37.6% of the time on average. For all other feasible instances, the calculation of the upper bounds is only 7.58% of the total computational time. The generation of the instances $\mathcal{I}_\rho$ represents 62.3% of the total time, whereas solving the TP is about 22.6% of the time.

As expected, the problem difficulty increases with the instance size (where the instance size can be measured in terms of number of sensors, and also in terms of number of ticks). However, it can be seen in Table 6 that the initial energy also plays a major role in the problem. Indeed, for low initial energy, most of the instances are infeasible. This is particularly noticeable for large instances, where the target is moving in a large area. By constrast, when energy is abundant, most of the instances are feasible, and more specifically, UB1′ is reached in most cases (in particular for small instances). Again, this is due to the fact that when sensors have high initial energy, the stability radius is rarely limited by energy (as in UB2′), but mostly set by the minimum distance between two time windows having no (or few) candidate sensors in common. Intermediate cases ($E_i = 16$) are more interesting, since for a lot of instances, no bound is reached, even if feasible instances are most of the time feasible. In this case, the dichotomy approach of Algorithm 3 is actually used. So these results show that the problem is generally easy to solve for extreme cases (low of high sensor initial energy), and the gap between these two extreme cases is rather narrow.

m	p	E_i	CPU_0	CPU_ρ	Avg. ρ
100	374.26	12	0.076 s	0.321 s	25.9
		16	0.074 s	0.112 s	34.8
		20	0.074 s	0.106 s	35.5
200	629.68	12	0.183 s	0.717 s	25.1
		16	0.184 s	0.432 s	45.9
		20	0.184 s	0.248 s	49.2
500	1466.12	12	1.106 s	1.804 s	26.3
		16	1.159 s	5.342 s	64.3
		20	1.138 s	2.754 s	78.4
1000	2708.54	12	4.311 s	10.47 s	48.5
		16	4.359 s	17.13 s	86.3
		20	4.276 s	14.70 s	106.2

Table 8: Average computation times versus the number of sensors and the initial sensor capacity

Table 7 shows the average CPU time when UB1′ or UB2′ is met (second column) and when none of these bounds is met (third column). Feasible instances only are taken into account for building Table 7. This table shows the efficiency of the upper bounds in solving the problem. The feasible instances for which UB1′ or UB2′ is optimal are in average about 5 to 7 times faster to solve than the rest of the feasible instances. This is due to the fact that whenever one of the upper bounds is reached, Algorithm 3 stops before the dichotomy step.

Table 8 gives the average computation times and the average optimal values of ρ according to the number of sensors and the initial battery capacity. The column CPU_0 contains the average time to reach the beginning of the loop of the dichotomy, i.e., the moment when the first feasible solution with $\rho = 0$ is found. The column CPU_ρ displays the average time to obtain an optimal solution.

A solution with a zero stability radius, such as the one found before the beginning of the dichotomy (see Table 8) consumes less energy than a maximum stability radius solution. This is due to the fact that the sensors have to be active ρ units of time before the target expected starting time, and ρ units of time after the arrival time. So the extra energy cost of a solution having a stability radius of ρ is equal to 2ρ. In our instances, the

extra energy cost represents 4.1% of the time horizon in average, and 16.1% in the worst case.

The computational effort required to obtain an optimal solution highly depends on the instance. When the instance is infeasible, or the upper bound is reached by the optimal solution, CPU_ρ is equal to CPU_0, because the algorithm stops before the dichotomy. Otherwise, CPU_ρ can be significantly larger because of the dichotomy. The value for CPU_ρ shown in Table 8 accounts for situations that lie in between these two extreme scenarios.

7. Conclusions and future lines

In this paper, we investigated a robustness problem of target tracking using wireless sensor networks. To the best of our knowledge, this problem has not been previously studied in the literature. The method proposed finds a robust sensor activity schedule in two steps. The discretization step transforms the input data into a scheduling problem instance. Then a dichotomy algorithm processes the instance in order to maximize its stability radius. By this approach, we prove that the overall problem can be solved in a pseudo-polynomial number of iterations, and a polynomial number of iterations if the target trajectory is a piecewise linear curve. Computational experiments show that the approach is scalable and solves problem instances involving up to 1000 sensors in less than 15 seconds on average. The two proposed upper bounds, especially $\text{UB1}'$, contribute to significatively speed up the method, as they are often reached and avoid to run the dichotomy algorithm. We have also shown the impact of sensors initial energy capacity on the computational effort required by the approach.

As a perspective of our work, another alternative to this problem could be to select sensors in order to save sensor capacity in a particular area. Thus, considering an enemy target that wishes to escape monitoring or to exhaust the sensors in a specific area, a problem variant could be to compute how long and where to send a target in order to compromise monitoring. In this paper, we have not considered the problem of routing collected data to a base station using multi-hop communication. This first study establishes simple and efficient upper bounds, that could be more difficult when communication issues are taken into account. However, the proposed framework can easily be extended to multiple target tracking without communication. Finally, we could consider that the spatial trajectory of the targets is subject to uncertainty. This hypothesis may lead to consider problem variants where the

time horizon is shorter, in order to cover the area in which the target is likely to be found, according to its speed and maneuverability. The robustness-based approach could also be complemented by a reactive approach, in order to cope with uncertainty on long trajectories.

Acknowledgements

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

References

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

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

[3] W. Zhang, G. Cao, DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks, IEEE Transactions on Wireless Communications 3 (5) (2004) 1689–1701, doi:10.1109/twc.2004.833443.

[4] M. Handy, M. Haase, D. Timmermann, Low energy adaptive clustering hierarchy with deterministic cluster-head selection, in: 4th International Workshop on Mobile and Wireless Communications Network, Institute of Electrical & Electronics Engineers (IEEE), 368–372, doi: 10.1109/mwcn.2002.1045790, 2002.

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

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

[7] H. Yang, B. Sikdar, A protocol for tracking mobile targets using sensor networks, in: Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, Institute of Electrical & Electronics Engineers (IEEE), 71–81, doi:10.1109/snpa.2003.1203358, 2003.

[8] Y. Xu, J. Winter, W.-C. Lee, Prediction-based strategies for energy saving in object tracking sensor networks, in: IEEE International Conference on Mobile Data Management, Institute of Electrical & Electronics Engineers (IEEE), 346–357, doi:10.1109/mdm.2004.1263084, 2004.

[9] H. Kung, D. Vlah, Efficient location tracking using sensor networks, in: IEEE Wireless Communications and Networking, vol. 3, Institute of Electrical & Electronics Engineers (IEEE), 1954–1961, doi:10.1109/wcnc.2003.1200686, 2003.

[10] Y. Xie, G. Tang, D. Wang, W. Xiao, D. Tang, J. Tang, A Fault-Tolerant Target-Tracking Strategy Based on Unreliable Sensing in Wireless Sensor Networks, in: IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, Institute of Electrical & Electronics Engineers (IEEE), 2116–2125, doi:10.1109/ipdpsw.2012.261, 2012.

[11] C. Laoudias, M. P. Michaelides, C. G. Panayiotou, ftTRACK: Fault-Tolerant Target Tracking in Binary Sensor Networks, ACM Transactions on Sensor Networks (TOSN) 10 (4) (2014) 1–28, doi:10.1145/2538509.

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

[13] M. Mannan, S. B. Rana, Fault Tolerance in Wireless Sensor Network, International Journal of Current Engineering and Technology 5 (3) (2015) 1785–1788.

[14] A. Oracevic, S. Ozdemir, A survey of secure target tracking algorithms for wireless sensor networks, in: World Congress on Computer Applications and Information Systems (WCCAIS), Institute of Electrical

& Electronics Engineers (IEEE), 1–6, doi:10.1109/wccais.2014.6916628, 2014.

[15] M. Naderan, M. Dehghan, H. Pedram, V. Hakami, Survey of mobile object tracking protocols in wireless sensor networks: a network-centric perspective, IJAHUC 11 (1) (2012) 34, doi:10.1504/ijahuc.2012.049283.

[16] J. Xiao, L. Weirong, Y. He, G. Qin, Energy-efficient sensor scheduling scheme for target tracking in wireless sensor networks, in: The 26th Chinese Control and Decision Conference (CCDC), Institute of Electrical & Electronics Engineers (IEEE), 1869–1874, doi:10.1109/ccdc.2014.6852474, 2014.

[17] O. Demigha, W.-K. Hidouci, T. Ahmed, On Energy Efficiency in Collaborative Target Tracking in Wireless Sensor Network: A Review, IEEE Communications Surveys & Tutorials 15 (3) (2013) 1210–1222, doi:10.1109/surv.2012.042512.00030.

[18] C.-T. Lee, F. Y.-S. Lin, Y.-F. Wen, An Efficient Object Tracking Algorithm in Wireless Sensor Networks, in: Proceedings of the 9th Joint Conference on Information Sciences (JCIS), Atlantis Press, doi:10.2991/jcis.2006.207, 2006.

[19] F. Yeong-Sung, Cheng-Ta, Y.-Y. Hsu, An energy-efficient algorithm for object tracking in Wireless Sensor Networks, in: IEEE International Conference on Wireless Communications, Networking and Information Security, Institute of Electrical & Electronics Engineers (IEEE), 424–430, doi:10.1109/wcins.2010.5544123, 2010.

[20] G. K. Atia, V. V. Veeravalli, J. A. Fuemmeler, Sensor Scheduling for Energy-Efficient Target Tracking in Sensor Networks, IEEE Transactions on Signal Processing 59 (10) (2011) 4923–4937, doi:10.1109/tsp.2011.2160055.

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

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

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

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

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

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

[27] H. Liu, X. Chu, Y.-W. Leung, X. Jia, P.-J. Wan, General Maximal Lifetime Sensor-Target Surveillance Problem and Its Solution, IEEE Transactions on Parallel and Distributed Systems 22 (10) (2011) 1757–1765, doi:10.1109/tpds.2011.42.

[28] R. J. Ellison, D. A. Fisher, R. C. Linger, H. F. Lipson, T. Longstaff, Survivable network systems: An emerging discipline, Tech. Rep., Carnegie-mellon Univ Pittsburgh PA Software Engineering Inst., 1997.

[29] L. Wang, Y. Xiao, A Survey of Energy-Efficient Scheduling Mechanisms in Sensor Networks, Mobile Networks and Applications 11 (5) (2006) 723–740, doi:10.1007/s11036-006-7798-5.

[30] Y.-S. Wang, F. Y.-S. Lin, C.-H. Chan, J.-W. Wang, Maximization of Wireless Mesh Networks Survivability to Assure Service Continuity under Intelligent Attacks, in: IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), Institute of Electrical & Electronics Engineers (IEEE), 583–590, doi:10.1109/aina.2013.31, 2013.

[31] B. Pannetier, J. Dezert, G. Sella, Multiple target tracking with wireless sensor network for ground battlefield surveillance, in: 17th International Conference on Information Fusion (FUSION), 1–8, 2014.

[32] C. Lersteau, M. Sevaux, A. Rossi, Multiple Mobile Target Tracking in Wireless Sensor Networks, in: Swarm Intelligence Based Optimization, Springer Science + Business Media, 123–130, doi:10.1007/978-3-319-12970-9_14, 2014.

[33] P. Berman, G. Calinescu, C. Shah, A. Zelikovsky, Power efficient monitoring management in sensor networks, in: IEEE Wireless Communications and Networking Conference, vol. 4, Institute of Electrical & Electronics Engineers (IEEE), 2329–2334, doi:10.1109/wcnc.2004.1311452, 2004.

[34] S. Slijepcevic, M. Potkonjak, Power efficient organization of wireless sensor networks, in: IEEE International Conference on Communications, vol. 2, Institute of Electrical & Electronics Engineers (IEEE), 472–476, doi:10.1109/icc.2001.936985, 2001.

[35] J.-C. Billaut, A. Moukrim, E. Sanlaville (Eds.), Flexibility and Robustness in Scheduling, Wiley-Blackwell, doi:10.1002/9780470611432, 2008.

[36] Y. Sotskov, A. Wagelmans, F. Werner, On the Calculation of the Stability Radius of an Optimal or an Approximate Schedule, Annals of Operations Research 83 (1998) 213–252.

[37] Y. N. Sotskov, A. Dolgui, M.-C. Portmann, Stability analysis of an optimal balance for an assembly line with fixed cycle time, European Journal of Operational Research 168 (3) (2006) 783–797, doi:10.1016/j.ejor.2004.07.028.

[38] L. R. Ford, D. R. Fulkerson, Solving the Transportation Problem, Management Science 3 (1) (1956) 24–32, doi:10.1287/mnsc.3.1.24.

[39] J. B. Orlin, Max flows in O(nm) time, or better, in: Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13, Association for Computing Machinery (ACM), 765–774, doi:10.1145/2488608.2488705, 2013.

[40] B. Dezső, A. Jüttner, P. Kovács, LEMON – an Open Source C++ Graph Template Library, Electronic Notes in Theoretical Computer Science 264 (5) (2011) 23–45, doi:10.1016/j.entcs.2011.06.003.

Appendix A. Instance with no guarantee on UB1 and UB2

An instance for which $\rho = 0$ and $\min\{\text{UB1}', \text{UB2}'\} > 0$ can be built as follows. The instance is composed of 3 sensors: s_1, s_2 and s_3. Sensor s_1 is available during the interval $[0, 5]$, s_2 during the interval $[2, 10]$ and s_3 during $[7, 12]$. The initial capacity of each sensor is $E_i = 4$. The instance is illustrated in Figure A.16.

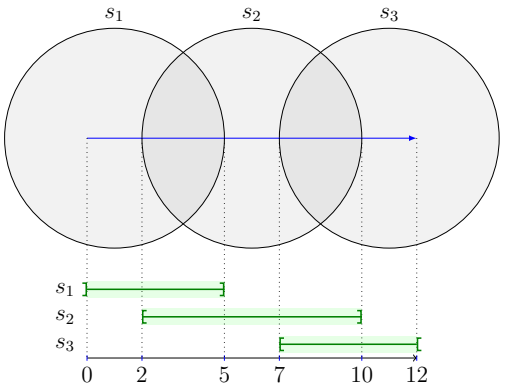

Figure A.16: Instance for which $\rho = 0$ and $\min\{\text{UB1}', \text{UB2}'\} > 0$

There exists a feasible schedule such that $\rho = 0$. The latter enables s_1 during $[0, 4]$, s_2 during $[4, 8]$ and s_3 during $[8, 12]$. The sum of the sensor capacities is 12, equal to the time horizon. Therefore, the stability radius cannot be larger than zero, and the proposed schedule is optimal.

The upper bound UB1$'$ is limited by the distance between the closest pair of time windows having no sensor in common, i.e. the time windows 1 and 3, or 3 and 5. Then $\text{UB1}' = \frac{3}{2} = 1.5$.

The total available capacity in the face composed of s_1 and s_2 (reached during the time window 2) is 8. As the candidate sensors of the time windows 1 and 3 are included in the set of candidate sensors of the time window 2, we assume that s_1 and s_2 will be used at least 7 units of time. Then the remaining capacity is 1 to extend the time windows, in 2 directions. Therefore, $\text{UB2}' = \frac{1}{2} = 0.5$.

Appendix B. Instance solvable in a pseudo-polynomial number of iterations

The instance composed of 2 sensors in Figure B.17 shows that the problem is solvable in a pseudo-polynomial number of operations.

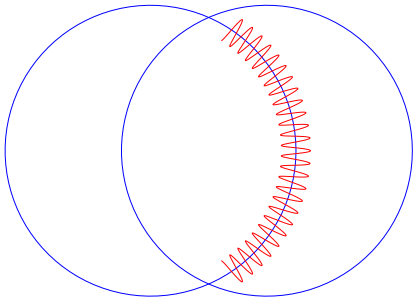

Figure B.17: Instance

The target trajectory is modeled as a sinusoid that moves along the boundary of the sensing area of one of the sensors. Since the period of the sinusoid can be as short as desired, the number of intersections with the boundary (also the number of time windows) can be as large as desired. Therefore, the number of iterations of the algorithm strongly depends on the period of the sinusoid.