Robust scheduling for target tracking using wireless sensor networks

Florian Delavernhe, Charly Lersteau, André Rossi, Marc Sevaux

Robust scheduling for target tracking using wireless sensor networks

Florian Delavernhe[a,*], Charly Lersteau[b], André Rossi[c], Marc Sevaux[d]

[a] *Université d'Angers, LERIA, F-49045 Angers, France*
[b] *Huazhong University of Science and Technology, State Key Laboratory of Digital Manufacturing Equipment & Technology, Wuhan 430074, China*
[c] *Université Paris-Dauphine, LAMSADE UMR 6285 CNRS, F-75016 Paris, France*
[d] *Université Bretagne Sud, Lab-STICC, F-56321 Lorient, France*

Abstract

A wireless sensor network (WSN) is a group of sensors deployed in an area, with all of them working on a battery and with direct communications inside the network. A fairly common situation, addressed in this work, is to monitor and record data with a WSN about vehicles (planes, terrestrial vehicles, boats, etc) passing by an area with damaged infrastructures. In such a context, an activation schedule for the sensors ensuring a continuous coverage of all the targets is required. Furthermore, the collected data, in order to be treated, have to be transmitted to a base station in the area, near the sensors. In this work, the future monitoring missions of the network are also taken into account, as well as the energy consumption of the current mission. We also consider that the spatial trajectories of the targets are known, whereas the speed of the targets along their trajectories are estimated, and subject to uncertainty. Hence, the main objective is to seek solutions that can withstand earliness and tardiness from the previsions. We propose a formulation of the problem with three different objectives and a solution method with experiments and results. The objectives are treated in a lexicographic order as follows (i) maximize the robustness schedule to cope with the advances and delaqui leys of the targets, (ii) maximize the minimum of monitoring time we can guarantee in priority areas, (iii) maximize the amount of energy left in the sensor batteries. We propose new upper bounds on the robustness measure, that are exploited by the solution approach whose complexity is shown to be pseudo-polynomial. The solution approach is based on a preprocessing step called discretisation, and the resolution of a series of linear programs.

Keywords: Linear Programming, Sensor Network, Robust Optimization, Target Tracking

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

1. Introduction

Wireless Sensor Networks (WSN) (Akyildiz et al. 2002, Yick et al. 2008) is a technology becoming more and more prominent in industry nowadays, with numerous applications and a bright promising future (Rawat et al. 2014, Modieginyane et al. 2018, Xu et al. 2018, Aalsalem et al. 2018). It involves the deployment of a network in an infrastructure-free area. Each node of the network is a small, cheap, easy to configure and reliable sensor, that will capture data about its environment. Eventually, the purpose of the network is fulfilled, even with a few defective sensors, by gathering and processing together the data collected by the nodes. They may include a wireless communication module, diverse sensing capabilities (humidity, temperature, light, movement and many others), wheels, different levels of batteries, etc. As an example in (Werner-Allen et al. 2006), to monitor volcanoes, the sensors are equipped with a seismometer, a microphone and a long-range radio. The sum of their data collected will warn when an eruption is likely to happen. Nonetheless, there are plenty of other different applications for WSNs (Yick et al. 2008), using their ability to monitor an area or track one or more targets. Many applications have high impact goals, where human lives are at stake. The military field is notably using the WSNs (Đurišić et al. 2012). Indeed such networks are easily deployed in enemy territory and are able to work while several sensors are discovered and destroyed by the enemy. In most of the applications, the networks are deployed in a remote area or an area dangerous to access, due to natural disaster or war conflict. Therefore, with the lack of infrastructure, the batteries of the sensors cannot be refilled. Consequently, the lifetime of the network (the period of time during which the network can fully serve its purpose) is limited. This leads to the implementation of sensor management optimization methods in order to provide efficient solutions to various problems.

We consider, in this work, the goal of tracking a set of targets by a sensor network (Liu and Liang 2005). For instance, in the military case, it refers to a WSN randomly deployed in a battlefield, with one or more targets traversing the battlefield. Once activated, a sensor is able to monitor the targets inside its sensing range and record data. The sensing range is the maximum distance at which a target can be to be monitored by a sensor. Thus, the basic goal in this problem is to find an activation schedule, that will alternate between active and inactive state for the sensors. The use of this kind of schedule will highly extend the lifetime of the network compared to a continuous activation of all the sensors (Benini et al. 2000). Keeping in mind that the schedule has to guarantee a full and continuous monitoring of the targets whenever and wherever the targets are located (at every instant a target, if it is under the range of at least one sensor, has to be monitored). Though, the network lifetime is limited by the sensing ranges and the battery capacities of the sensors.

The activity of sensing a given set of targets is called a mission, and to the best of our knowledge, all the contributions in optimization to wireless sensor networks are focused on a single mission for the monitoring of moving target. The problem addressed in this paper is to decide when to activate each sensor so as to maximize the ability of the resulting schedule to keep being feasible despite uncertainty affecting the targets speed along their trajectory for the current mission. The secondary objective is to maximize the wireless sensor network ability (*i.e.* the amount of time available with

the batteries) to monitor a given set of zones of interest for future missions. The last objective is to minimize the total amount of energy required by the current monitoring mission.

In the case where targets are terrestrial vehicles moving on roads or tracks, their geographical trajectory can be predicted quite naturally (*e.g.* buses or trains). For boats or aircrafts, the trajectory can be estimated at least for a short amount of time. In this work, we assume that such a trajectory prediction is available, but the considered uncertainty is about the speed at which the targets move along their trajectory.

The main contributions of this paper are the following ones. First, a more realistic model than the one in (Lersteau et al. 2016) is considered for the WSN, where sensor communication is taken into account in the new model. In most of the applications of WSN, the communication of the data collected is mandatory and since the power consumption due to communication is much larger than the consumption due to sensing (Anastasi et al. 2009), the new model proposed in this paper is much more accurate. Furthermore, the previous work (Lersteau et al. 2016) is also extended to consider multiple targets in the problem. As a result, we introduce a new upper bound on the stability radius defined in Section 6, and the previous bounds introduced in (Lersteau et al. 2016) are naturally extended to include communication and multiple targets. The stability radius (Sotskov et al. 1998) is a measure of the ability of the network to remain feasible despite uncertainty, it is formally defined in Section 3. Second, this paper proposes an extended and comprehensive approach to produce robust solutions, refined with the two additional criteria: the results in (Lersteau et al. 2016) are extended to more than one targets, and hop-communication (communication between sensors to transfer the collected data to a base station) is taken into account. Finally, this approach also has energy considerations added to the robust scheduling, with priority areas.

This paper is organized as follows: Section 2 presents the related work, it is followed by a reminder of the original definition of the stability radius in Section 3. Section 4 defines the problem. Afterwards, we present a discretisation phase, as a preamble to the solution method in Section 5 and propose an upper bound for the stability radius in Section 6; Section 7 introduces the solution method; Section 8 presents the experiments and their results and finally, Section 9 concludes this paper.

2. Related Works

The WSN field has plenty of possible applications already addressed in the literature, with different types of wireless networks considered and different optimization problems. We outline here several problems divided in two types of WSN: mobile and static ones.

On the one hand, as presented in (Mohamed et al. 2017) in a mobile WSN, sensors are still able to move after their initial deployment as they are motorized. Thus, in these problems, the literature is based mostly on coverage related objectives where the advantage of the mobility of the sensors is used to fill uncovered parts or to assure a minimum level of coverage. As an example in (Elhoseny et al. 2018) the k-coverage problem is addressed. The aim is to cover all locations of the field with at least k sensors. In an other example (Liu and Liang 2005) treats the θ-coverage where a full coverage of the zone is impossible and therefore the problem is to cover at least θ%

of the area. Moreover, there are different other objectives. In (Patel et al. 2005), the authors addressed a cluster-based problem. Cluster-heads are often used in networks for performing data fusion on the data collected by the sensors. They can be assigned to various potential locations that cover different sets of sensors. Since the sensors are mobile over the horizon of time, the quality (data coverage) of a location is fluctuating. Therefore, the network is able to relocate several times the cluster-heads to different locations, with a cost for each relocation. Consequently, in their work, the authors proposed a column generation heuristic to find an optimal trade-off between the data coverage and the relocation costs.

On the other hand, in the static WSN, the sensors being deployed cannot move. A popular application of static WSN is the target tracking, where different objectives can be optimized, such as energy consumption, scalability, fault tolerance and tracking precision.

Energy consumption and network's lifetime are critical since batteries are often assumed to be not refillable. There exists a very extensive literature about these two objectives in target tracking. For example, there are two prominent protocols used to save energy, LEACH (Handy et al. 2002) and HEED (Younis and Fahmy 2004). In the case of static targets, Cardei and Du (2005) increase the lifetime of the network by organizing the set of sensors into a maximal number of disjoint subsets of sensors that cover all the targets. The authors showed that if the sets are activated in turn, the lifetime of the network is extended. Furthermore, an efficient power management method is presented in (Campos-Nañez et al. 2008), using a game-theoretic approach to propose a distributed scheme. The network lifetime is often maximized with an efficient schedule of the sensors' activity. For example, (Castaño et al. 2014) propose a column generation approach to compute a schedule for maximizing the network lifetime under connectivity and coverage constraints. In (Carrabs et al. 2015), the network lifetime is also maximized, but they consider that each sensor is assigned to a family and each family has a coverage requirement. The lifetime maximisation may also consider connectivity constraints (*i.e.*, the data collected is transmitted to a central processing node) and thus multi-role sensors. In these cases, the power consumption of sensors depends on their roles, *i.e.*, idle, relaying or monitoring, as in (Carrabs et al. 2016, 2017, Castaño et al. 2015). Lifetime maximization in the context of target tracking is currently a hot topic: as an example, Alibeiki et al. (2019) propose a genetic-based approach for a directional sensor network with adjustable range. The heuristic approach finds efficient solution to monitor non-moving target while maximizing the lifetime of the network. A sensor is only able to monitor the targets inside its sensing range, that is adjustable, in its direction of activation. The greater the sensing range, the more energy is consumed, and the sensor has several working directions, but it can use only one of them at a time.

The criterion of scalability as presented in (Kung and Vlah 2003, Naderan et al. 2012) is also an important one. With this objective, a WSN protocol should scale well to large numbers of sensors or targets, since a dense network or an important number of targets can significantly increase the communication consumption. Scalable protocols typically use cluster-based or distributed approaches instead of centralized ones. The tracking precision criterion optimizes how far the estimated locations or passing times of the targets can be derived from the real locations or times (Lersteau

et al. 2016, Naderan et al. 2012). Literature covers the probability to lose a target, the recovery process, robustness against delays and advances or trust region for the location, noise sensitivity, etc. For the fault detection criterion, the algorithms detect flaws in the network or unexpected external events and reconfigure the network as quickly as possible, as in (Jin et al. 2015).

In our study, we extend the work of (Lersteau et al. 2016) where the authors addressed a target tracking robustness problem in Wireless Sensor Networks. The aim is to find a schedule that covers a single target at any time, with the target position supposed to be exactly known over a time horizon. However, the targets can be subjected to delays or advances and the schedule is protected from these perturbations by the stability radius ρ. Indeed, whatever the delays or advances of the targets at any point of their trajectory, the schedule remains feasible as long as these values remain less than the stability radius. In the way it is presented in (Lersteau et al. 2016), this problem has some common features with an assembly line such as in (Sotskov et al. 2006), where the aim is to schedule operations on a minimum number of stations with possible variations in the processing time, under a cycle time constraint. The method presented in (Lersteau et al. 2016) starts with a discretisation phase, *i.e.*, the transformation of the geometrical problem into data used to model a combinatorial optimization problem. The covered space is partitioned into faces (this term is defined in Section 5). The target has several time windows as it moves through different faces, with each transition between two faces called a tick. The target trajectory is no longer needed and is turned into a succession of time windows, with a list of available sensors to monitor the target during each such time window. The authors proposed a pseudo-polynomial two-step algorithm. They noticed that the increase of the stability radius has an impact on the time windows, and their solution approach relies on a bisection method to find a feasible set of time windows with the highest possible stability radius. Each step of the bisection methods solves a transportation problem. Finally, a linear program is solved to maximize the stability radius for the time windows returned by the bisection method. The present paper also relies on the discretisation and bisection phases but extends the problem by considering multiple targets and communication constraints. Thus, we now need to find a route for the collected data: from the activated sensors, relayed by several sensors, to a base station. Communication heavily impacts the sensor batteries and thus the returned solutions. In addition, we have added different objectives, aiming to save energy for future missions, which is again an extension of (Lersteau et al. 2016).

3. Formal definition of the stability radius of a schedule

A formal definition of the stability radius of a schedule can be seen in (Sotskov et al. 1998). In the scheduling problem, the stability radius is an indicator on the greatest variation on the processing times of the jobs for which the optimal schedule remains optimal. In such problems, each job i of the set of jobs Q has a processing time p_i and, in a schedule s, a completion time $c(s)_i$. Uncertain events can impact the processing time of a job i with a variation of ϵ_i such that the new processing time is equal to $p_i \pm \epsilon_i$, which will also affect the completion time. A schedule has a stability radius of ϱ if and only if it is always the optimal schedule for all the vectors of processing time $p' \in O_\varrho(p)$, with $O_\varrho(p)$ the closed ball centered on $p = \{p_1, p_2, \ldots, p_{|Q|}\}$ the original

processing times and with a radius equals to ϱ. It means that the schedule remains feasible if for each job i, $\epsilon_i \leq \varrho$.

After introducing the problem in the next section, the original definition of the stability radius is adapted to the target tracking problem in Section 5.

4. Problem Definition

The problem of multiple targets tracking by a wireless sensor network, with hop-communications of data to a base station is addressed. Hop-communication means that a sensor that cannot communicate directly with the base station can sends its data to the base station through intermediate sensors, that serve as relays. This problem is referred to as P_c^n where n is the number of targets and c stands for communication. In this problem, during a given time horizon T, a set J of n targets with their spatial trajectories already known, will traverse a zone monitored by a set I of m sensors. A function $\tau_j(t)$ estimates the position of the target j at each instant t.

First, the network has to guarantee a constant monitoring of each target. If the trajectory of a target is not continuously under the monitoring range of at least one sensor, the problem is considered infeasible. Note that if these targets are neglected and removed from our problem, we may obtain a new feasible problem. Second, the network has to transfer all the data collected to a base station which is connected to a permanent source of energy and is able to forward the data to a remote control center. For that purpose, the sensors are equipped with communication modules to transmit and receive data. Once activated they are able to form a path from the monitoring sensors to the base station, with several sensors used as relays if necessary.

The sensors can only communicate with other sensors or with the base station if they are under its communication range R_c. The sensor can monitor only the targets that are under its sensing range R_s, *i.e.*, at an instant t, targets that are more than R_s meters away from a sensor i cannot be monitored by this sensor. $N(i)$ is the set of all the sensors in the communication range of the sensor i, *i.e.* those who can send and receive information from sensor i. Each sensor $i \in I$ has a limited battery with an energy of E_i joules.

The sensors are multi-role (Castaño et al. 2016) and therefore we consider three kinds of energy consumption:

- Monitoring a target requires a power of p^S watts (a watt is a joule per second)
- Emitting data requires p^T watts.
- Receiving data from another sensor requires p^R watts.

A monitoring activity by a sensor collects data that is necessarily transmitted to the base station. Thus, if a sensor i is monitoring a target for s seconds, i will also transmit s seconds of data, hence i will consumes $(p^S + p^T) \times s$ joules. Likewise, in a relay activity, if the sensor i receives s seconds of data, i will transmit s seconds of data, hence i will consume $(p^R + p^T) \times s$ joules. Therefore, monitoring a target for s seconds, draws $(p^S + p^T) \times s$ joules out of the battery of the monitoring sensor and $(p^R + p^T) \times s$ joules out of each battery of the sensors used to relay the collected data to the base station. Note

that the sensors are always sending the collected information. These activities (sensing, receiving and transmitting) can take place in the sensor at the same time. Moreover, if a sensor is monitoring x targets at the same time, it also consumes x times its monitoring power. For example, at an instant $t \in T$, if a sensor monitors two targets and receives data from another sensor, it will transmit the data from these three activities. Hence, for this sensor, the instant power consumption at t is $2p^S + p^R + 3p^T$ watts. *i.e.*, it consumes energy for the monitoring of both targets, plus consumes for receiving information, plus consumes three times the emitting consumption since it transmits both the data collected while sensing and the data received from another sensor. Consequently, a non sensing nor transmitting nor receiving sensor is not consuming energy. More detailed and complex energy consumption models are presented in (Halgamuge et al. 2009, Miller and Vaidya 2005).

A long-term solution of the problem is expected to preserve the network ability to respond to future monitoring missions. Hence in the sequel, we only activate one sensor at a time to monitor a target since activating multiple sensors will just collect redundant data and waste energy. However, several sensors can be activated at the same instant for the transmission task, or to monitor different targets. Furthermore, we also consider priority areas, also called hot spots in the coverage related literature (Huang and Tseng 2005). The aim is to preserve and balance the residual capacities of the batteries for future target tracking missions (Lersteau et al. 2018). The decision makers define multiple priority areas where the solution should preserve as much monitoring time as possible. Moreover, because all the areas may not be considered equally important, the network managers set a rank $\ell \in C$ to each area, where C is the set of ranks and r is the maximum rank. The rank of an area is fixed, it does not evolve during the monitoring, and the higher the rank is the more important the area is. The rank expresses a preference for the preservation of energy in the considered areas. We treat the problem as a multi-objective one, with a lexicographic order of the objectives. The primary objective is to find an activation schedule for the sensors that respects the constraints (continuously monitor the targets, transfer data, respect battery constraints) and that maximizes the stability radius. The general notion of stability radius is defined in Section 3 and a definition adapted to our problem in Section 5. Thus, the schedule remains feasible even if the targets are late or early to any point of their trajectories, provided that earliness or lateness stay below the stability radius.

The second objective is to maximize the minimal amount of monitoring time guaranteed in the priority areas in each rank $\ell \in C$, while respecting the priority as expressed by the ranks of areas. This means that the guaranteed monitoring time available anywhere inside a priority area of rank r is maximized, then, among all the solutions that offer this guaranteed coverage, we maximize the coverage guarantee everywhere inside the priority areas of rank $r - 1$, and so on up to rank 1.

Finally, as a third objective, we minimize the total amount of energy required by the current mission, while maintaining all the previous objective to their optimal values.

Table 1 summarizes the notations.

$I = \{1, \ldots, m\}$	Set of sensors
$J = \{1, \ldots, n\}$	Set of targets
E_i	Energy of the battery of sensor i
p^S	Power consumption in watts for sensing
p^R	Power consumption in watts for receiving data
p^T	Power consumption in watts for transmitting data
R_c	Communication range of a sensor
R_s	Sensing range of a sensor
T	Time horizon
$C = \{1, \ldots, r\}$	Set of ranks of areas
$\tau_j(t)$	Position of target $j \in J$ at time $t \in [0, T]$
$N(i)$	Set of all the sensors in range of communication of sensor i

Table 1: Summary of the notations

5. Discretisation

The problem input is a set of geographical data. The sensors with their characteristics and their position, are deployed in a zone, along with the priority areas, the targets and their routes. However, to determine the schedule of the sensors' activity and the routing of the collected data, the problem instance has to be discretised. Discretisation is based on (Lersteau et al. 2016), and has been extended to the case of multiple targets, with communications and priority areas. Let us consider the example of Figure 1, where the yellow disks represent the sensing range of three sensors, the black arrow the route of a target, B the base station and the gray disks model the communication range. Two priority areas are shown as green polygons, they have two different ranks, the smallest one being the most important one.

A *face* is defined as the set of all the locations (*i.e.*, spatial points in the zone) monitored by the same subset of sensors (in Figure 1, face {1,2,3} is the set of all location points that are under the range of sensors 1, 2 and 3). Each face f is associated a set of candidate sensors $S(f)$, *i.e.*, any sensor that can monitor f is in $S(f)$. The discretisation phase turns the trajectory of each target as a sequence of visited faces, as doing so allows to select one sensor in the set of the candidate sensors of a face to monitor the target. The instant when a target moves from a face to another one is called a tick. When a target enters the range of a sensor, it defines an entering tick; it is a leaving tick when it leaves the range of a sensor. The k-th tick of target j is denoted by t_k^j which also denotes its date of appearance, and σ_k^j is an integer value which is $+1$ if the tick is *entering*, and -1 if it is *leaving*. By convention, the first and last ticks are respectively leaving and entering (Lersteau et al. 2016). In addition, a time window Δ_k^j is the duration between two successive ticks t_k^j and t_{k+1}^j.

In the example of Figure 1, three faces only are considered (the other ones are not visited by the target): {1}, {1,2} and {1,2,3}. Hence this defines four ticks as presented in Table 2.

For the priority areas and their given rank, the problem is discretized as follows. For a rank, with its priority represented by an integer $\ell \in \{1, \ldots, r\}$, where r is the

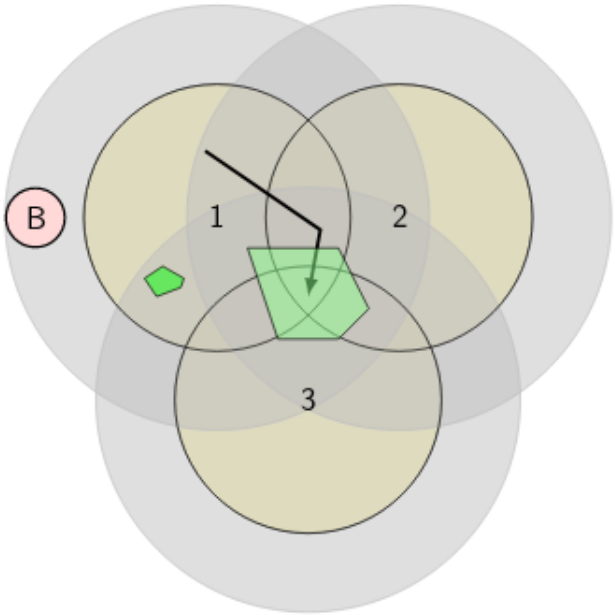

Figure 1: A three-sensor example

Face	{1}	{1,2}	{1,2,3}	
Time window length (given)	$\Delta_0^1 = 5$	$\Delta_1^1 = 10$	$\Delta_2^1 = 2.5$	
Tick	$t_0^1 = 0$ $\sigma_0^1 = -1$	$t_1^1 = 5$ $\sigma_1^1 = +1$	$t_2^1 = 15$ $\sigma_2^1 = +1$	$t_3^1 = 17.5$ $\sigma_3^1 = +1$

Table 2: The four ticks of example of Figure 1

number of different ranks, wherever a potential target appears inside a priority area whose rank is ℓ, the network should provide the same monitoring time guaranteed. For each rank $\ell \in \{1, \ldots, r\}$, $\mathcal{F}(\ell)$ is the set of all the faces that have a non-empty intersection with an area of rank ℓ, and no intersection with an area of higher rank. Indeed, if a face has a non-empty intersection with two areas having ranks ℓ and ℓ' with $\ell < \ell'$, this face is part of $\mathcal{F}(\ell')$ only since ℓ' is the most important rank. The set of all the sensors that can monitor at least one face in $\mathcal{F}(\ell)$ is denoted by $T(\ell)$. More formally, $T(\ell) = \cup_{f \in \mathcal{F}(\ell)} S(f)$ for all $\ell \in \{1, \ldots, r\}$. In order to guarantee the same amount of monitoring time wherever a potential target might be in an area of rank ℓ, we should guarantee the same amount of monitoring time for every face $f \in \mathcal{F}(\ell)$. Therefore, the priority areas and ranks are now discretized into sets of faces $\mathcal{F}(\ell)$ for each rank ℓ. For example, in Figure 1, if the smallest area has rank 2 and the other one has rank 1, then $\mathcal{F}(1) = \{\{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}, \}$ and $\mathcal{F}(2) = \{1\}$. The reason why face $\{1\}$ is not in $\mathcal{F}(1)$ is that it belongs to $\mathcal{F}(2)$. The candidate sensors for the two ranks are $T(1) = \{1, 2, 3\}$, and $T(2) = \{1\}$, it can be seen that the same sensor may appear in different $T(\ell)$ sets with no inconvenience. Finally, for a given

rank ℓ, we can reduce $\mathcal{F}(\ell)$ by keeping only the faces with no other faces from $\mathcal{F}(\ell)$ included in them, *i.e.*, we keep only a face $f \in \mathcal{F}(\ell)$ if $\forall f' \in \mathcal{F}(\ell), f' \notin f$. Indeed, if a set of sensors s is guaranteed an amount of monitoring time t, then each super-set s' of s has at least the amount of time t guaranteed. In the example of Figure 1, we have $\mathcal{F}(1) = \{\{2\}, \{3\}\}$ and $\mathcal{F}(2) = \{\{1\}\}$. $\mathcal{F}(1)$ is reduced to only two faces since, for example, the face $\{1, 2, 3\}$ is not considered because it has at least as much covered time guaranteed as the face $\{2\}$.

The communication between sensors is represented by the communication digraph $\vec{G_c} = (I \cup \{B\}, A)$ with B the base station and A the pairs of sensors that can communicate. For all the collected data, an optimal flow from the monitoring sensor to the base station is computed inside $\vec{G_c}$. With the example of Figure 1, we obtain the following $\vec{G_c}$ (Figure 2):

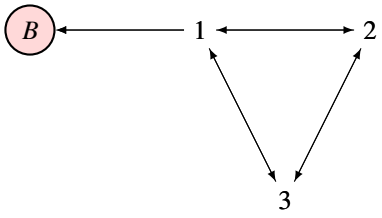

Figure 2: The communication digraph $\vec{G_c}$ for the example of Figure 1

The new notations introduced in this section appear in Table 3.

t_k^j	Time of tick k of target j
σ_k^j	$\begin{cases} +1 & \text{if the tick } k \text{ of target } j \text{ is entering,} \\ -1 & \text{otherwise.} \end{cases}$
$T(\ell)$	Potential subsets of candidate sensors that can monitor a target in a face ranked ℓ
r	Total number of ranks for areas
$\mathcal{F}(\ell)$	Set of all faces that have a nonempty intersection with an area of rank ℓ and an empty intersection with each rank $\ell' > \ell$
$S(f)$	Set of sensors covering the face f

Table 3: Summary of the notations introduced for discretisation

Finally, the stability radius of a feasible schedule of the sensors' activity is a measure of its ability to maintain a full coverage to the targets despite their advances and delays. More specifically, we assume that after the discretisation phase, target $j \in J$ crosses $K_j \geq 1$ time windows, and should spend Δ_k^j units of time in time window k, for all $k \in \{1, \ldots, K_j\}$.

The stability radius of a feasible schedule $\mathcal{S}$ can be defined using the following notations (see [(Sotskov et al. 1998)]):

- $\vec{\zeta} = (\zeta_{jk} \in \mathbb{R})_{j \in J, k \in K_j}$ is the set of possible advances and delays, *i.e.*, ζ_{jk} is the deviation of the actual time spent by target j in time window k compared to the estimated amount of time Δ_k^j. $\zeta_{jk} < 0$ corresponds to an advance and $\zeta_{jk} > 0$ to

a delay. Thus $\zeta_{jk} \geq -\Delta_k^j$ for all $j \in J$ and for all $k \in K_j$, because a target cannot spend a strictly negative amount of time in a time window.

- $\Omega(\Lambda)$ is the set of feasible schedules for Λ, with Λ the set of all λ_{jk} such that target j spends λ_{jk} units of time in time window k, for all $j \in J$ and for all $k \in K_j$.

The stability radius of solution $\mathcal{S}$ can be written as

$$\rho(\mathcal{S}, \Theta) = \max\{\varepsilon > 0 \mid \forall \zeta \in B(\varepsilon), \mathcal{S} \in \Omega(\Theta + \zeta)\}$$

Where $B(\varepsilon) = \{\zeta \in \vec{\zeta} | \, \|\zeta\|_\infty \leq \varepsilon\}$, and Θ is the set of all Δ_k^j.

Hence, the stability radius $\rho(\mathcal{S}, \Theta)$ of schedule $\mathcal{S}$ is the maximum amount of delay or advance of that the targets can have in each time window, without compromising the ability of $\mathcal{S}$ to ensure a full coverage of all the targets.

6. Upper bound on the stability radius of a schedule

In this section, we present different upper bounds. The final upper bound used in the sequel is the smallest bound out of three upper bounds, two of them are natural extensions from (Lersteau et al. 2016) and are presented in 6.1, the last one is a novel contribution of this work, introduced in Section 6.2.

6.1. Definition of two general upper bounds on the stability radius

We aim at producing a schedule of the sensors' activity that maximizes the value of the stability radius, in order to offer the maximum protection against delays and advances. Upper bounds are useful for achieving this goal, and are part of the solution approach introduced in Section 7.

The two bounds presented in (Lersteau et al. 2016) are both built upon the expansion of the timespan of each time window. The first one, denoted by UB_1, is based on the time windows and their set of common candidate sensors, and the second one, UB_2, relies on the sensor capacities. These two upper bounds on the stability radius, that have originally been introduced for a single target without communication, are extended to the case of multiple targets and now also consider the energy consumption for communication. Thus, we obtain two tighter bound:

$$\mathrm{UB}_1 = \min_{\substack{j \in J \\ (k,k') \in K_j^2}} \left\{ \frac{1}{2} \left(\frac{\sum_{i \in S_j(k) \cap S_j(k')} E_i}{p^S + p^T} + t_{k'}^j - t_{k+1}^j \right) | k < k' \right\}$$

where $S_j(k)$ is the set of candidate sensors for the target j during time window k. $t_{k'}^j$ is the tick delimiting the beginning of the time window k' and t_{k+1}^j the leaving tick associated with the time window k.

$$\mathrm{UB}_2 = \min_{f \in F} \left(\frac{\sum_{i \in S(f)} E_i}{2 \times (p^S + p^T) \times n_f} \right)$$

where F is the set of faces visited by the targets, $S(f)$ is the set of sensors covering the face f and n_f the number of times the face f is visited (*i.e.*, the number of times a target enter in the face f).

6.2. Upper bound based on the communication digraph

In this section, we introduce a new upper bound on the stability radius based on the communication consumption and sensing consumption. Indeed, the previous bounds have been introduced in a context where communication was not taken into account. The extensions brought to these bounds in this work are not considering the data relay tasks, *i.e.*, the reception and transfer of data from other sensors. These extensions only consider the transfer of the data by the sensors that collected it. However, it is more likely that the relay of the data is needed a lot and therefore will impact significantly the batteries. Since the sensors may need multiples relays to send the data to the base station and considering that the transmission and reception powers are not negligible compared to sensing, the performance of the computed schedule will heavily depend on the routes available to transfer the data.

In the application context, some subsets of sensors will have to carry all the communication to reach the base station, typically the sensors surrounding the base station. For each of these sets of sensors, increasing the lifetime of the network induces more communication, and then more power consumption. Thus, it is more likely that one of those sets of sensors has first all batteries drained and depleted, and imposes a limitation on the stability radius value, so the sum of the battery capacities of all sensors in this set defines an upper bound on the value of the stability radius for any feasible schedule. Hence, for each face f, we find the set of sensors for which all the data collected will get relayed, we call it a *restraining* set of f. Moreover, we note that any feasible schedule covers at least the case with no delay nor advance. In such a case, the computation of the minimal amount of data that need to be collected in a face is straightforward, *i.e.*, it is equal to the amount of time spent by the targets in that face. Hence, we also compute the minimum amount of collected data that is then transmitted by all the restraining sets. Therefore, we compute an upper bound on the stability radius based on the battery level of all the restraining sets, and on the minimum amount of collected data.

We remind that the communication digraph $\vec{G}_c = (I \cup \{B\}, A)$ models all different paths for the data transmission between the sensors and the base station (see Section 5). Let f be a face that is visited by at least one target for a nonzero duration, then the data collected in f require sensors for receiving and transmitting data for $\sum_{j \in J} \left(2\nu_f^j \rho + \sum_{\substack{k \in K_j \\ S_j(k) = S(f)}} \Delta_k^j \right)$ units of time, where ν_f^j is set to one if and only if target j visits the face f, and zero otherwise. Indeed, if two different targets visit the same face, they both could spend 2ρ units of extra time in that face, with ρ the stability radius. 2ρ is the maximal extra time a target can spend in a face while being covered by a schedule with a stability radius ρ. This data transmission is still possible provided that for each vertex separator $V_f \subset I$ of $\vec{G}_c$ that partitions $I \cup \{B\} \backslash V_f$ into two connected components (the first one containing B, and the second one containing $S(f)$, the sensors from

f), the energy of the sensors of V_f should be sufficient to ensure data reception and transmission, this can be stated as:

$$\sum_{j\in J}\left(2\nu_f^j\rho+\sum_{\substack{k\in K_j\\ S_j(k)=S(f)}}\Delta_k^j\right)(p^R+p^T)\leq\sum_{i\in V_f}E_i$$

In order to obtain the tighter possible upper bound on ρ, the vertex separator V_f may be such that the sum of the energy of the batteries is minimal. Furthermore, the above inequality can be tightened by considering that V_f splits the vertices of $I\cup\{B\}\backslash V_f$ into two sets X_B and X_f. X_B is the vertex set that includes B, and X_f is the vertex set that includes $S(f)$. Consequently, the vertex separator V_f separates B from all the faces whose candidate sensors are a subset of X_f. Consequently the sensors of V_f should have enough energy to:

- receive and transmit all the data collected from the faces with sensors in X_f and without any sensors in V_f,
- receive or monitor, and then transmit the data from the faces with sensors in X_f and with some sensors in V_f but not all of them,
- monitor and transmit the data from the faces with all sensors include in X_f and V_f.

We obtain:

$$\sum_{j\in J}\left(2\mu_{V_f}^j\rho+\sum_{\substack{k\in K_j\\ S_j(k)\subseteq V_f}}\Delta_k^j\right)p^S+\sum_{j\in J}\left(2\gamma_{V_f}^j\rho+\sum_{\substack{k\in K_j\\ S_j(k)\nsubseteq V_f\\ S_j(k)\cap V_f\neq\emptyset}}\Delta_k^j\right)\min(p^S,p^R)+$$

$$\sum_{j\in J}\left(2\varepsilon_{X_f\backslash V_f}^j\rho+\sum_{\substack{k\in K_j\\ S_j(k)\subseteq X_f\backslash V_f}}\Delta_k^j\right)p^R+\sum_{j\in J}\left(2\nu_{X_f}^j\rho+\sum_{\substack{k\in K_j\\ S_j(k)\subseteq X_f}}\Delta_k^j\right)p^T\leq\sum_{i\in V_f}E_i$$

where for all $j\in J$, the following constants $\mu_{V_f}^j$, $\gamma_{V_f}^j$ and $\varepsilon_{X_f\backslash V_f}^j$ are defined by:

- $\mu_{V_f}^j$ is set to one if and only if target j spends a nonzero amount of time in at least one face whose candidate sensors form a subset of V_f, otherwise $\mu_{V_f}^j=0$,
- $\gamma_{V_f}^j$ is set to one if and only if target j spends a nonzero amount of time in at least one face whose candidate sensors have a non-empty intersection with V_f, but do not form a subset of V_f, otherwise $\gamma_{V_f}^j=0$,
- $\varepsilon_{X_f\backslash V_f}^j$ is set to one if and only if target j spends a nonzero amount of time in at least one face whose candidate sensors form a subset of $X_f\backslash V_f$, otherwise $\varepsilon_{X_f\backslash V_f}^j=0$.

As a consequence, UB_3 can be defined as :

$$\rho \leq \mathrm{UB}_3 = \min_{f \in F^*} \left(\frac{\sum_{i \in V_f} E_i - \sum_{j \in J} \left(\sum_{\substack{k \in K_j \\ S_j(k) \subseteq V_f}} \Delta_k^j p^S + \sum_{\substack{k \in K_j \\ S_j(k) \nsubseteq V_f \\ S_j(k) \cap V_f \neq \emptyset}} \Delta_k^j \min(p^S, p^R) + \sum_{\substack{k \in K_j \\ S_j(k) \subseteq X_f \setminus V_f}} \Delta_k^j p^R + \sum_{\substack{k \in K_j \\ S_j(k) \subseteq X_f}} \Delta_k^j p^T \right)}{2 \sum_{j \in J} \left(\mu_{V_f}^j p^S + \gamma_{V_f}^j \min(p^S, p^R) + \varepsilon_{X_f \setminus V_f}^j p^R + \nu_{X_f}^j p^T \right)} \right)$$

7. Solution Method

Since the problem has three objectives handled in a lexicographical order, we separate it in three successive problems: P1 for maximizing the stability radius, P2 for maximizing the time guaranteed in priority areas, and P3 for minimizing the energy consumption. They are solved sequentially, starting from P1 to P3 where the objective of a problem becomes a constraint in the next one. Therefore, the solution method has three successive steps. First, a schedule with a maximum stability radius is sought (P1 is presented in details in Section 7.1). Since P1 cannot be addressed by simply solving a linear program because the set of faces to be covered depends on the stability radius, we solve this problem using first a bisection method. Each step creates and solves a new decision problem for a fixed value of the stability radius Δ with a linear program (LP) called LP_Δ. Once the maximum value of Δ, called Δ_{opt}, is found in a discrete set of potential values with a bisection method, a final linear program called LP_ρ is solved. It finds the optimal value of the stability radius $\rho = \Delta_{opt} + \delta$, with δ the objective function of the optimal solution of LP_ρ.

Next, P2 is addressed. For each rank ℓ, from the highest one to the lowest one, we find a solution maximizing the time during which a target in a priority area of this rank is guaranteed to be monitored. To reach this goal, for each face of a rank, a linear program denoted by LP_ℓ is solved.

The third step is to solve P3: the energy consumption to monitor all the targets is minimized. This is done by solving a linear program denoted by LP_E.

Algorithm 1 summarizes the overall approach.

7.1. P1: Maximization of the stability radius

The maximization of the stability radius is achieved by enhancing the bisection method presented in (Lersteau et al. 2016). In this subsection, we give an overview of this method while providing more details on the proposed adaptation to our problem. The main observation is that increasing the stability radius leads to postpone the entering ticks and advance the leaving ticks. Whenever two ticks from different time windows interchange their order of appearance, a time window disappears, and a new one is created, with a reduced set of candidate sensors. As a result, the constraints of the problem have to be updated. It should be noted that there is a discrete set of

Algorithm 1: Solving P_1, P_2 then P_3

```
// P1 - Maximization of the robustness
while (Δ ← getNextValueDichotomy()) ≠ null do
  MakeLP_Δ (Δ)
  SolveLP_Δ ()
  if IsFeasibleLP_Δ () then
    Δ_opt ← Δ
  end if
end while
MakeLP_ρ (LP_Δopt )
SolveLP_ρ ()
// P2 - Maximization of the guarantees in the priority areas
for all ℓ ∈ C do
  if ℓ = 0 then
    MakeLP_ℓ (LP_ρ,ℓ)
  else
    MakeLP_ℓ (LP_ℓ−1,ℓ)
  end if
  SolveLP_ℓ ()
end for
// P3 - Minimization of the energy consumption
MakeLP_E (LP_|C|)
SolveLP_E ()
return GetSolutionLP_E ()
```

values for the stability radius that causes such updates, these values being the distances between entering and leaving ticks belonging to different time windows. We use the upper bound of the stability radius, presented in Section 6, to reduce this set of values by removing all the values greater than the upper bound.

The first part in solving P1 consists in finding the maximum value of the stability radius in this discrete set for which the problem is feasible. This is done with a bisection method that checks the existence of a feasible schedule for a given value in this set, until the largest one is found. The first value tested is always the upper bound of the stability radius. Indeed, if a feasible schedule with such a stability radius is found, then it is optimal for P1. If this value is not feasible, a stability radius of 0 is tested. If no solution is found in that case, then there is no feasible schedule for P1 and the algorithm stops. This typically happens when a target gets out of the range of any sensor, or when the network is so sparse that the collected data cannot be sent to the base station. If a solution is found with a nonnegative stability radius, then the bisection method is used to find the largest stability radius in the discrete set. This method differs from the one introduced in (Lersteau et al. 2016) in the search of a feasible schedule. Indeed, due to communication requirements, we have to solve a linear program LP_Δ instead of a transportation problem. This linear program has no objective function. It is shown in Model 1, and is addressed with a solver.

The decision variables are x_{jik}, the monitoring time of the target j during its time window k by the sensor i, and $f_{ii'}$ the amount of data transferred from sensor i to sensor i'. We introduce H_j^i as the set of all time windows of target j for which i is a candidate sensor, *i.e.*, the set of all $k \in K_j$ such that i belongs to $S_j(k)$. The linear program LP_Δ (with no objective function) is the following:

$$\sum_{j \in J} \sum_{k \in H_j^i} x_{jik} p^S + p^R \sum_{i' \in N(i)} f_{i'i} + p^T \sum_{i' \in N(i)} f_{ii'} \leq E_i \qquad \forall i \in I \quad (1)$$

$$\sum_{j \in J} \sum_{k \in H_j^i} x_{jik} + \sum_{i' \in N(i)} f_{i'i} - \sum_{i' \in N(i)} f_{ii'} = 0 \qquad \forall i \in I \quad (2)$$

$$\sum_{i \in S_j(k)} x_{jik} = \Delta_k^j \qquad \forall j \in J, k \in K_j \quad (3)$$

$$x_{jik} \geq 0 \qquad \forall j \in J, k \in K_j, i \in S_j(k) \quad (4)$$

$$f_{ii'} \geq 0 \qquad \forall i \in I, i' \in N(i) \quad (5)$$

Model 1: LP_Δ, the linear program solved at each iteration of the bisection method

The first constraints (1) represent the limitation of the battery for each sensor, (2) model the transfer of all data to the base station (it is a flow conservation equation). Finally (3) enforce the coverage constraints, *i.e.*, each target is continuously covered by a sensor at any time.

At the end of the bisection method, we obtain the maximum value Δ_{opt} for which the problem is feasible. Afterwards, while considering the time windows in that case, we maximize the stability radius increase δ, by solving a linear program similar to the one used for determining Δ_{opt} in LP_Δ. The new linear program is called LP_ρ, presented in Model 2, it is identical to the one used in the bisection method, with in addition an objective function (6), a constraint (7) and an updated constraint (3').

$$\delta_{opt} = \text{Maximize } \delta \quad (6)$$

$$\sum_{i \in S_j(k)} x_{jik} = \Delta_k^j + (\sigma_{k+1}^j - \sigma_k^j)\delta \qquad \forall j \in J, k \in K_j \quad (3')$$

$$\delta \geq 0 \quad (7)$$

Subject to (1), (2), (4), (5)

Model 2: LP_ρ solved to maximize the stability radius

Since δ is the stability radius increase from Δ_{opt}, it impacts the duration of the time windows in (3'). σ_k^j is the value of the k-th tick of the target j (-1 if leaving, 1 if entering). The optimal stability radius numerical value is then $\rho = \Delta_{opt} + \delta$.

The example of Figure 3 illustrates the method.

In this example, with different values of ρ there are different time windows, with different candidate sensors. The stability radius ρ may be in the interval $[0, 97)$, or in $[97, 189)$, or in $[189, \text{Upper bound})$, or be equal to Upper bound. During the bisection method, we find $\Delta_{opt} = 97$, which means that there is a feasible schedule with a stability

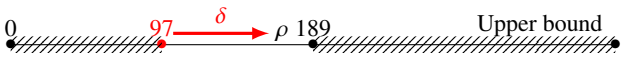

Figure 3: Illustration of the solution method for P1

radius of 97. Therefore all schedules with a stability radius which is strictly less than 97 are not considered anymore. This results also implies that there is no solution either with a value larger than or equal to 189. The second step in solving P1 is then to maximize δ with $0 \leq \delta < (189 - 97)$. Solving LP_ρ yields the optimal value of ρ, which is equal to $97 + \delta$ in this example.

7.2. P2: Maximizing the coverage guarantee in the priority areas

Now that a feasible solution with the best stability radius is found, we search for a schedule that maximizes the coverage guarantee in the priority areas. Let us remind that a rank corresponds to a set of faces, and a face can only appear in one rank.

In this phase, for each rank from the highest one to the lowest one, we create a new problem, where we maximize T_ℓ, *i.e.*, the monitoring time guaranteed in all areas whose rank is ℓ after the current mission. For each rank, a linear program denoted by LP_ℓ is addressed. It is recalled that a rank ℓ fixes the optimal values from the previous problems solved (the stability radius and the $T_{\ell'}$ for all $\ell' > \ell$), and is focused on the maximization of T_ℓ. Therefore the linear programs solved are built incrementally, by adding and modifying constraints and variables from the previous linear programs solved ($LP_{\ell+1}$ if $\ell < r$, LP_ρ otherwise).

From the previous solved linear program, we fix first its optimal value in a new constraint to keep the optimal values of the previous phases. We then change the objective function to maximize T_ℓ, the coverage guaranteed in the areas of rank ℓ. Next, we add virtual targets in all the faces included in $\mathcal{F}(\ell)$ and monitor them. These targets all have the same time guarantee, T_ℓ, and model the covering requirement induced by the faces of rank ℓ. Furthermore, a virtual routing of the data (new set of flow constraints) is added to route the potential data recorded from the virtual targets. Thus, when monitoring those faces will be required, the set of candidate sensors will have enough residual energy after the current mission to track targets in those faces for T_ℓ units of time, and there will be a path of sensors with enough energy left to transmit the data to the base station.

The new decision variables are the following:

- $x_{if\ell}$ the time during which sensor $i \in S(f)$ monitors a fictitious target in face f having rank ℓ.
- $f^1_{ii'}$ the amount of data generated by a fictitious target, transmitted from i to i'.

The LP model for a rank ℓ is the following:
Constraint (1') is the constraints (1) modified to take into account the virtual flow and fictive targets that are added. It can be seen that by comparison to LP_ρ, sensors can now be used to perform sensing and data transmission after the current mission, to insure the network ability to monitor the priority areas.
Constraints (2), (3'), (4), (5) are the same as in P1

Maximize T_ℓ (6')

$$\sum_{j\in J}\sum_{k\in H_j^i} x_{jik}p^S + p^R\sum_{i'\in N(i)} f_{i'i} + p^T\sum_{i'\in N(i)} f_{ii'}$$
$$+p^S\sum_{f\in \bigcup_{\ell\le\ell'\le r}\mathcal{F}(\ell')|i\in S(f)} x_{if\ell'} + p^R\sum_{i'\in N(i)} f^1_{i'i} + p^T\sum_{i'\in N(i)} f^1_{ii'} \le E_i \quad \forall i\in I \quad (1')$$

$$\delta = \delta_{opt} \quad (8)$$

$$\sum_{i\in S(f)} x_{if\ell'} = T_{\ell'} \quad \forall \ell'\in C, \ell'\ge\ell, \ell'\le r, \forall f\in\mathcal{F}(\ell') \quad (9)$$

$$\sum_{f\in \bigcup_{\ell\le\ell'\le r}\mathcal{F}(\ell')|i\in S(f)} x_{if\ell'} + \sum_{i'\in N(i)} f^1_{i'i} - \sum_{i'\in N(i)} f^1_{ii'} = 0 \quad \forall i\in I \quad (10)$$

$$f^1_{ii'} \ge 0 \quad \forall i\in I, i'\in N(i) \quad (11)$$

$$T_{\ell'} = T^{opt}_{\ell'} \quad \forall \ell'\in C, \ell' > \ell \quad (12)$$

$$x_{if\ell'} \ge 0 \quad \forall \ell'\in C, \ell'\ge\ell, \ell' < r, \forall f\in\mathcal{F}(\ell'), i\in S(f) \quad (13)$$

Subject to (2), (3'), (4), (5)

Model 3: LP_ℓ Model solved for a rank ℓ

Constraint (8) sets δ to δ_{opt}, the optimal objective value of LP_ρ.
Constraints (9) ensure the tracking of the fictive targets for a rank ℓ' for a duration of $T_{\ell'}$ units of time.
Constraints (10) ensure the flow balance for the data collected generated by fictive targets, at each sensor i.
Constraints (12) set the value for the previous ranks ℓ'.

7.3. P3: Minimizing energy consumption

Problem P3 is to minimize f_3, the total energy consumed to achieve the current mission. In order to solve it, the linear program LP_E is built from the last linear program solved in P2, by fixing its objective value in a new constraint, changing the objective function and resolving it.

While solving P3, the schedule and the routing can be changed, but the stability radius value and the coverage guarantees found in P1 and P2 are maintained to their optimal respective values.

The energy consumed by a sensor is the left-hand side of constraints (1), this quantity can be written as:

$$\sum_{i\in I}\Bigg(\sum_{j\in J}\sum_{k\in H_j^i} x_{jik}p^S + p^R\sum_{i'\in N(i)} f_{i'i} + p^T\sum_{i'\in N(i)} f_{ii'}$$
$$+p^S\sum_{f\in \bigcup_{\ell\le\ell'\le r}\mathcal{F}(\ell')|i\in S(f)} x_{if\ell} + p^R\sum_{i'\in N(i)} f^1_{i'i} + p^T\sum_{i'\in N(i)} f^1_{ii'}\Bigg)$$

The virtual flow and fictive targets are not actual consumptions since they are not used during the current mission, but are spared for possible future missions, so they are subtracted from the previous quantity, leading to:

$$\sum_{i \in I} \left(\sum_{j \in J} \sum_{k \in H_j^i} x_{jik} p^S + p^R \sum_{i' \in N(i)} f_{i'i} + p^T \sum_{i' \in N(i)} f_{ii'} \right)$$

Finally, since the total recording time of the target is a constant (the stability radius is fixed), we just have to minimize the transferred data in the current mission, so f_3, the objective function of P3 is:

$$f_3 = \text{Minimize} \sum_{i \in I, i' \in N(i)} f_{i'i}$$

7.4. Complexity analysis

There are two important parts in the proposed algorithms, the discretisation phase and the solution method (*i.e.*, the solution of P1, P2 and P3).

First, the discretisation can be achieved using a pseudo-polynomial algorithm. A precise analyze of the complexity of the discretization can be found in (Lersteau et al. 2016), that shows that the number of faces is bounded by a polynomial on the number of sensors. Thus, the modifications brought in this work (communication, priority areas and multiple targets), that still rely on the numbers of sensors and faces, does not change the complexity.

Secondly, P1 is a dichotomy that requires a logarithmic number of iterations, with at each iteration, a linear program solved. The dichotomy is done on a set of discrete values, that in the worst case is equals to all possible intersections between the ticks. And, if the targets have polygonal trajectories, the number of ticks cannot exceed $2qm$, with q the number of segments in the trajectories and m still the number of sensors. In P2, we solve as many linear programs as the number of ranks and finally in P3, there is only one linear program to solve.

8. Numerical experiments

8.1. Description of the protocol

In this last part, we present our experiments, results and analysis. We study the behavior of the solution method and the impact of different parameters like the number of sensors, targets and ranks. Moreover, we evaluate the efficiency of the upper bound introduced in this work compared to the other ones extended from (Lersteau et al. 2016). To this end, we design four experiments on different sets of instances, each of them investigates the impact of the problem from the following:

- Impact of the sensing and communication powers. We vary them in order to compare UB_3 to the two other upper bounds. In this experiment, we study the efficiency of this new upper bound on the stability radius.
- Impact of the sensor density. In the experiment, only the number of sensors is varied.

- Impact of the number of targets. More targets to monitor induces more data, send or receive.
- Impact of the number of ranks and areas.
- Impact of the communication.

Although each experiment has its proper set of instances, they are all generated using the instance generator presented in Section 8.2.

8.2. Dataset

All of our instances are generated with the same algorithm. The generator's inputs are:

- The number of sensors m
- The surface of the rectangular zone $L_1 \times L_2$
- The number of targets n to monitor
- The minimum E_{min} and maximum E_{max} level of energy initially available in the batteries of the sensors
- The powers associated with the different tasks (p^S, p^T and p^R)
- The communication range R_c and sensing range R_s
- The number of priority areas q and ranks r
- The maximum radius of the priority areas R_a

First of all, the sensors are randomly deployed in the zone. Each sensor has a random level of battery picked between the two values given as parameters. Second, the journeys of the targets are drawn also randomly in the zone. Their paths are simple routes made of three segments. Third, the priority areas are randomly deployed and their rank are also randomly chosen, with at least one priority area per rank. Each priority area is a disc whose radius is selected randomly between fifty percent and one hundred percent of the maximum value given as parameters. Finally, the base station is randomly deployed in the zone.

The default parameters of our instances are presented in Table 4.

Parameter	Value	Parameter	Value
Number of sensors m	400	Energy of the sensors $[E_{min}, E_{max}]$	$[350, 400]$
Number of targets n	2	p^S	2.8
Size of the area $L_1 \times L_2$	300×300	p^R	1
Sensing range R_s	35	p^T	1
Communication range R_c	70	Number of areas q	5
Maximum radius of the priority areas R_a	50	Number of ranks r	2

Table 4: Default values of the parameters in the instances generator

8.3. Results and analysis

We present in this section the results and analysis of the different experiments. The software is coded in C++ and all the experiments were run on a computer with Ubuntu 16.04 and Intel Core i7-6700HQ CPU @ 2.60GHz × 8 cores and 16 GBytes of RAM. We use the version 12.7 of IBM CLPEX for solving the linear programs. The CPU times reported are in seconds.

8.3.1. Impact of the sensing and communication powers

In this first experiment, we study the quality of the new upper bound. The objective is to compare it to the previous upper bounds and see if the method benefits from the addition of UB_3. Indeed, the two previous bounds are still valid in our robustness problem and have been extended to consider multiple targets and communication costs. Though, they are both only considering in a face f, the consumption induced by the coverage of the targets inside this face (*i.e.*, sensing the targets inside f and transmitting the data collected). They are not considering the consumption induced by forwarding the data collected in other faces. Hence, the addition of our new upper bound based on such principle seems to be a good opportunity to help the solution process. Indeed, the value of the upper bound is important since it may reduce drastically the number of iterations in the bisection method in P_1, thus the number of linear programs to be solved. Clearly, it is expected to perform better when the communication costs are significant compared to sensing. In this experiment, we are studying the efficiency of the new bound compared to the two extended previous bounds when working on WSN with several significant ratios between the sensing power p^S and the communication powers p^R and p^T.

We test three different sets of powers. In all of them, $p^R = p^T$ and $p^S = 3$. First, we set $p^R = p^T = 0.5$. In the second set, we have $p^R = p^T = p^S = 3$. And in the third set, $p^R = p^T = 5$. We generate a set of fifty instances, and we use the three different powers on each of them. All other parameters are fixed as presented in Table 4, except for the number of sensors and targets fixed to 300 and 4 respectively.

We report the number of times where UB_3 dominates the two other bounds in Table 5.

p^S	$p^R = p^T$	# { $\min(UB_1, UB_2) > UB_3$ }
3	0.5	2 / 50
3	3	20 / 50
3	5	26 / 50

Table 5: Domination of UB_3 over UB_1 and UB_2

Table 5 shows that with a low consumption for communication, UB_3 is dominated. Indeed, it reaches the best value in only 2 instances out of 50. Clearly, UB_1 and UB_2 are less impacted by low communication costs, by contrast with UB_3 which is mostly based on these costs. However, for an average consumption, with $p^R = p^T = p^S = 3$, UB_3 is almost as good as the two other upper bounds combined. In such a case, in 20 instances out of 50, UB_3 is strictly the best upper bound on the stability radius, and is therefore really useful in the bisection method. Finally, we observe that when

power consumption due to communication is much more significant than sensing power consumption, which is the most realistic situation (Anastasi et al. 2009), our new upper bound dominates the other two ones in 26 cases out of 50, and is therefore significantly more efficient. Naturally, this dominance gets stronger when the power consumption due to communication increases.

Though, the two other bounds should still be considered since they are useful in almost half of the instances (note that for the highest communication costs, UB_1 was the best bound in 12 instances, same for UB_2).

To conclude, the new introduced bound, UB_3, has good performances compared to the two other ones and it makes a significant contribution to the approach, in reducing the number of linear programs to solve in the bisection method. Though, it is less useful when low communication costs are considered.

8.3.2. Impact of sensor density

In this second part, we study the impact of sensor density on the method and the values found. For that purpose, we generate a set of fifty feasible instances using the default parameters (Table 4), except for the number of sensors which is fixed to 200 sensors. Afterwards, we add a few sensors to each instance in order to study the impact of sensor density in the network. Each sensor added is generated in the same way as the initial sensors, *i.e.*, they have random positions in the $L_1 \times L_2$ area and a random level of initial battery in $[E_{min}, E_{max}]$. The method is run on the instances when the number of sensors m is in $\{200, 250, 300, 350, 400, 700\}$.

In Table 6, we report the average number of time windows and the number of faces included in a priority area. Table 7 reports the computational effort required by each problem of the solution method with the average number of linear programs solved in P1. Finally, in Table 8 we report the average objectives values. The CPU times are in seconds.

#Sensors	# windows	# faces in priority areas
200	103.48	85.94
250	128.94	133.58
300	154.58	188.26
350	179.94	254.40
400	204.94	324.80
700	357.38	973.34

Table 6: Average number of time windows and faces in priority areas with different numbers of sensors

Table 6 shows that when sensor density increases, the number of time windows increases also. Consequently, each step of the solving method takes more time to solve, with more data to process, more constraints and more variables in the models. Furthermore, there is also a fast increase of the number of faces inside the priority areas. Therefore, the models for P2 and P3 are becoming even more complex with more constraints and more variables.

Each row of table 7 presents the average results over fifty instances. The first column is the number of sensors. The second one is the CPU time (in seconds) spent by

#Sensors	CPU (Discretisation)	CPU (UB)	CPU (P1)	LP solved in P1	CPU (P2)	CPU (P3)	Overall CPU time
200	0.02	0.04	0.28	3.44	1.22	0.49	2.05
250	0.02	0.06	0.16	1.22	2.38	1.14	3.75
300	0.04	0.13	0.41	2.30	4.01	1.86	6.45
350	0.06	0.20	0.35	1.22	6.39	2.62	9.62
400	0.10	0.33	0.53	1.48	8.94	3.63	13.54
700	0.86	2.47	1.03	1.00	59.65	19.52	83.53

Table 7: Average computational effort with different numbers of sensors

the discretisation step, CPU (UB) is the average time (in seconds) for computing the upper bound (6). Column 4, 6 and 7 represent the CPU times for solving P1, P2 and P3 respectively. Column 5 represents the average number of linear programs when solving P1. The last column is the average overall CPU time (s) required to solve an instance.

As expected, the density of the network impacts a lot the solution time. Indeed, the results show that, when the number of sensors increases, the computational effort required for every step and thus the overall CPU time are increasing quickly. It is explained by Table 6, that shows that the number of time windows is increasing. Consequently, the CPU times required by the discretisation and the computation of the upper bound are increasing quickly, since they are mostly dependent on the number of time windows. The solution time of P1 also depends on the number of linear programs to be solved. This is why the CPU time for P1 is not always increasing when the density increases, since there are often less linear programs solved in P1 (because the upper is more often a feasible solution) though they are more complex to solve. The solution processes of P2 and P3 are heavily impacted by the number of faces in the priority areas, hence their running times are also increasing significantly when the sensor density is higher.

#Sensors	Stability radius	T_2	T_1	f_3
200	79.26	40.24	5.84	10816.16
250	102.98	43.74	4.87	10667.43
300	113.56	45.58	3.33	10509.07
350	119.75	47.58	1.56	10342.56
400	124.12	45.30	1.35	10312.58
700	143.20	32.01	1.23	10179.60

Table 8: Evolution of the objectives with the number of sensors

The Table 8 shows that the stability radius is increasing with the number of sensors. The second objective seems to increase at first but with more than 350 it decreases. The following objective, T_1, is impacted negatively by the increase of the number of sensors, with the covering time left in the priority areas rank 1 overall decreasing. Finally, surprisingly, the last objectives (*i.e.*, minimization of the communication consumption) is getting better with more sensors even if a higher stability radius induces more data collected and transferred.

As expected, adding sensors to a network extends the robustness. It adds more opportunities for the network to monitor the targets and to transfer the collected data. Thus, it is less restrained by the battery and by the constraint of having only one sensor monitoring a target at a time. This observation is also explaining the decrease of the last

objective. With more sensors, the network has more opportunities for shorter and more direct routes to the base station. It induces less transfer of data between the sensors and therefore less energy consumption. The objectives on the times guaranteed in the priority areas are affected differently. Indeed, with more sensors, there is more energy in the network and it seems that it gives more opportunity to increase these objectives. However, for each rank ℓ in C, T_ℓ is constrained by the time guaranteed in each face f in $\mathcal{F}(\ell)$, hence T_ℓ is constrained by the less covered face. Adding sensors in the network is not bringing necessarily more sensors in each face already in $\mathcal{F}(\ell)$. Thus, it can still be the same face (*i.e.*, the exact same set of candidate sensors) that restrain the value of T_ℓ after adding sensors to the network. However it still adds more faces to cover in the priority areas (Table 6). Consequently, the network is having less coverage time guaranteed in the priority areas.

8.3.3. Impact of the number of targets

In this third experiment, we analyze the impact of the number of targets. As in the second experiment, all parameters are fixed (see Table 4) except the one we study. The number of targets varies in the set $\{1,\ldots,7\}$. An initial set of feasible instances is generated and solved with 7 targets. Afterwards, we just remove the targets, one by one, to the instances to test on different numbers of targets.

We report in our results the average times for all the instances. Results are summarized in Table 9. The values of the objectives are reported in Table 10. The CPU time reported are in seconds.

#Targets	CPU (Discretisation)	CPU (UB)	CPU (P1)	CPU (P2)	CPU (P3)	Overall CPU time
1	0.10	0.15	0.55	9.09	3.02	12.90
2	0.10	0.28	0.49	9.33	3.22	13.41
3	0.10	0.39	0.50	9.30	3.30	13.57
4	0.10	0.49	0.75	9.75	3.38	14.48
5	0.10	0.61	1.19	10.38	3.60	15.88
6	0.10	0.74	1.84	11.25	3.66	17.59
7	0.10	0.89	5.30	12.13	4.18	22.61

Table 9: Results with different numbers of targets

In this experiment, we observe that P1, P2 and P3 have increasing CPU times resulting in an overall growth of the computational effort. However the results can be analyzed more deeply. First, for the discretisation, only the part of the computation of the routes of the targets is impacted, and not the discretization of the communication or the priority areas. Consequently the time used by this phase is increasing only slowly. Secondly, the most increasing times are for P1 and the computation of the upper bounds as they are totally dependent on the number of targets: more targets imply more time windows and a stability radius more complex to compute. Finally, the time needed for P2 and P3 is increasing slowly since only the part of the model inherited from P1 is impacted.

To conclude on the computational effort, as expected, each part of the solution method needs more time when targets are added.

The values of the objectives in Table 10 show that with more targets, the stability radius is obviously decreasing. Indeed, with more targets the network is more limited,

#Targets	Stability radius	T_2	T_1	f_3
1	207.60	55.84	5.93	5629.14
2	127.61	51.30	4.63	9568.11
3	105.98	48.30	3.25	13606.10
4	98.79	43.40	1.95	18139.19
5	93.77	35.90	1.05	22820.62
6	87.98	26.07	0.67	28297.36
7	72.55	20.31	0.36	33337.56

Table 10: Evolution of the objectives with the number of targets

because it consumes more to monitor the targets. Furthermore, if we increase the stability radius, we increase it for every target. Hence, we consume more energy when increasing the stability radius for problems with an increasing number of targets.

The same observation can be made with the other objectives. The time guaranteed in the priority areas is decreasing and the total energy consumption is also increasing. It shows that the network has still a lot of energy after solving P1 for a few targets. In such cases, the stability radius is bounded by a specific region of the network or by a constraint on the number of sensors activated at the same time. Therefore the total amount of energy consumed (minimized in P3) is limited, and more energy is available to guarantee a higher time in the priority areas. However, with more targets, the network consumes more for the stability radius, so the third objective is increasing and moreover there is less energy for the priority areas. Therefore, with more targets, even with the stability radius decreasing, we consume more energy and can guarantee less time for the priority areas.

8.3.4. Impact of the number of ranks and areas

In this set of experiments, we study the impact of the priority areas by varying the number of areas generated with the number of ranks. As for the experiment on targets, we first generate and solve a set of fifty instances with only two areas of the same rank. Afterwards, we add areas and ranks on the same instances. Therefore, we obtain different versions. For each instance, we have versions with 1 to 7 ranks for respectively 2, 4 to 14 areas with at least one area per rank. Average times are presented in Table 11 and objective values for P2 and P3 (the stability radius is not varying) in Table 12. The CPU times are in seconds.

#Ranks	#Areas	CPU (Discretisation)	CPU (UB)	CPU (P1)	CPU (P2)	CPU (P3)	Overall CPU time
1	2	0.06	0.33	0.55	4.59	3.38	8.90
2	4	0.08	0.36	0.57	10.77	4.17	15.96
3	6	0.12	0.33	0.56	14.82	4.09	19.92
4	8	0.16	0.33	0.56	19.12	4.12	24.29
5	10	0.20	0.32	0.55	23.94	3.95	28.96
6	12	0.25	0.33	0.57	29.39	4.04	34.57
7	14	0.29	0.32	0.57	34.51	4.11	39.80

Table 11: Results with different numbers of ranks

This experiment shows that the numbers of ranks and areas increase the computational effort for the discretisation and for the problem P2. Moreover, as expected,

the computation of the upper bound on the stability radius and problem P1 are not impacted by the areas. We can observe that, with the increase of the number of ranks, the overall CPU time is increasing, up to five times more from one to seven ranks.

#Ranks	T_1	T_2	T_3	T_4	T_5	T_6	T_7	f_3
1	46.93							10058.05
2	7.58	50.56						10158.19
3	2.97	12.61	64.61					10383.87
4	1.44	3.94	24.39	68.00				10425.95
5	0.09	0.91	6.48	25.07	65.57			10197.06
6	0.00	0.00	1.31	6.04	22.87	66.62		10407.70
7	0.00	0.00	0.16	1.22	6.77	26.86	70.71	10480.30

Table 12: Evolution of the objectives with the number of ranks

Table 12 shows that the time guaranteed in an area is heavily depending on the priority assigned to it. Indeed, as an example the areas of rank 1 have a lot of time guaranteed when they are the only priority areas. However, for the same areas, the time is dropping when we add higher ranks. Moreover, with a lot of ranks, there is a really small time guaranteed for the less important ranks. Therefore, we conclude that the time guaranteed in an area is heavily depending of its rank. For the most critical areas, the time guaranteed is huge but will heavily decrease along the priority. For low priority, there is often no time guaranteed. We advise to only use up to three or four ranks.

Secondly, we notice that the energy consumed is globally not varying. Indeed, it seems that the overall time guaranteed in all the priority areas is not considerably rising because it is probably limited by the same region of sensors whatever the number of ranks. Therefore, the state of the network and the energy left after solving P2 does not differ much with more or less ranks. The process of minimizing the energy consumption obtains similar values.

8.3.5. Impact of the communication in the objectives

In this experiment, we study the impact of the communication. *i.e.*, how the mandatory data transfer to the base station has impacted the solution quality and the computational effort. Taking communication into account makes the problem more complex to solve and is restraining the objectives. In order to study such assumptions, we compare the objectives of the solution obtained by the method with and without considering the communication on a set of instances. Several different communication costs are tested to represent different impacts in the network. Fifty instances are generated with the default parameters presented before (Section 8.2). Our method is run multiple times on each of these instances, each time with a different communication power consumption. The values used for p^R and p^T are $\{1, 2, 3\}$, with $p^T = p^R$. The value of p^S is still equals to the default value. Afterwards, for each instance, we run our method without considering the communications costs (*i.e.*, $p^T = p^R = 0$). We report the average value of the objectives in Table 13, for each cost tested.

Afterwards, a second set of feasible instances is generated, with the default parameters except for the number of targets fixed to 1 and no priority area. Hence, on

these instances, we can compare our method and the method developed in previous work (Lersteau et al. 2016). It is recalled that communication is ignored in this article. Several communication costs are tested again for our method, with $p^R = p^T$ and p^R in $\{0, 1, 2, 3\}$. The CPU times (still in second) needed for the computation of UB and P1 are reported in Table 14.

$p^R = p^T$	Stability radius	T_2	T_1	f_3
0	124.60	90.52	40.68	0
1	124.27	52.94	4.58	9559.63
2	122.38	27.79	0.84	19975.08
3	111.92	14.63	0.10	30292.13

Table 13: Impact of communication on the objective values

Problem	$p^R = p^T$	CPU (UB)	CPU (P1)
Our Problem	0	0.17	0.13
	1	0.17	0.44
	2	0.17	1.05
	3	0.17	1.19
Lersteau et al. (2016)	0	0.004	0.08

Table 14: Impact of communication on the solution times

As expected, all the values of the objectives are worsening when increasing the communications costs. Although, for the stability radius, the difference is really small between no communication cost ($p^R = p^T = 0$) and low costs. However, there is a bigger difference when comparing no cost to the most important costs ($p^R = p^T = 3$), with a loss of approximately 10% of the stability radius. The coverage times left in the priority areas, for the two ranks, are also quickly decreasing with higher costs. When communication costs are added, the priority areas of rank 1 are losing almost all their coverage time, and the areas of rank 2 an important part of it. Finally, the overall energy consumed for communication (*i.e.*, f_3) is obviously continuously increasing with the costs.

These results confirm that, as expected, the communication is restraining the objectives. The value of the stability radius is impacted, since more energy are necessary to cover the targets. Though, it does not vary a lot with low communication costs. Indeed, the stability radius is also constrained by the the constraints forcing to only have one sensor activated per target for sensing at any time. Thus, in these cases, adding communication costs is not necessarily reducing the stability radius. That is why the values found for the stability radius are similar in our instances with no communication costs and low costs. Though, the stability radius is decreasing a lot more with high costs. The others objectives are worsening a lot more when the communication costs are added. The energy consumption is increased a lot, and not only the sensors in range of a target are consuming energy. The energy left in the network for the priority areas is a lot smaller, thus restraining these objectives.

To conclude on the objective values, the addition of the communication is certainly

impacting the objectives. However, the loss of robustness is small, especially for low communication costs.

As expected, the CPU times in Table 14 are worst with our problem, since the communication are considered. Both the computation of the upper bound and P1 are longer to compute. For the upper bound, it does not vary with the different tested costs. However, the difference between the time needed for solving P1, with communication and without it, is becoming more and more important with higher costs. It also shows that solving the problem without considering the communication is faster than solving the same problem while considering the communication but without any costs, which, again, is not surprising.

These results were expected, since the problem solved is much more complex, and even with null communication costs, the communication is still computed. The linear program solved in the bisection method in the present work require much more computational effort to solve than the transportation problem in (Lersteau et al. 2016). The upper bound may also be less restraining thus increasing the running time of P1. Finally, the new bound added, is obviously increasing the solution time of UB.

9. Conclusion

In this paper, we extended the original problem, treated in (Lersteau et al. 2016). Their objective was to find an activation schedule to track a target with a WSN. We developed it to make the problem more generic and it now handles cases with multiple targets to be tracked at the same time by the same WSN. Furthermore, we added communications between sensors, with the task to transfer all the data gathered to a base station. The communication is a great generalization since the impact on a WSN is important, and in some applications may be even more consuming that the sensing. These two extensions change the solution process. Indeed, the discretisation is modified to deal with the communication and the targets. Likewise, the optimization of the stability radius (problem P1) is more general. We solved a new model by linear programming instead of a transportation problem. We adapted the previous upper bounds to multiple targets and communication and introduced a new bound, based on energy consumption that is due to communication. The relevance of this new bound has also been checked. Afterwards, we added two objectives in our problem, optimized after P1. The new objectives are the maximization of the time guaranteed inside the priority areas (P2) and the maximization of the overall energy left (P3). We added two steps to the solution method for these objectives, based on new models formulated as linear programs.

We designed different experiments to test and analyze the solution method. The results show that increasing the numbers of sensors, targets, ranks or areas is contributing to the rise of the overall running time. The number of sensors, *i.e.*, the density, is the most significant parameter in terms of CPU time. Finally, we also showed that the new upper bound based on communication, is efficient and is used in the solution process.

To go further, several extensions could be considered. First, what if one targets has a delay or an advance outside the stability radius? How to dynamically change the solution to avoid losing targets? And how to achieve this at low computational cost so

as to meet real time constraints? In order to test various realistic situations, we intend to use CupCarbon simulator (CupCarbon, Mehdi et al. 2014).

References

Mohammed Y Aalsalem, Wazir Zada Khan, Wajeb Gharibi, Muhammad Khurram Khan, and Quratulain Arshad. Wireless sensor networks in oil and gas industry: Recent advances, taxonomy, requirements, and open challenges. *Journal of Network and Computer Applications*, 113:87–97, 2018.

Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications magazine*, 40(8):102–114, 2002.

Abolghasem Alibeiki, Homayun Motameni, and Hosein Mohamadi. A new genetic-based approach for maximizing network lifetime in directional sensor networks with adjustable sensing ranges. *Pervasive and Mobile Computing*, 52:1–12, 2019.

Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella. Energy conservation in wireless sensor networks: A survey. *Ad hoc networks*, 7(3):537–568, 2009.

Luca Benini, Giuliano Castelli, Alberto Macii, Enrico Macii, Massimo Poncino, and Riccardo Scarsi. A discrete-time battery model for high-level power estimation. In *Proceedings of the conference on Design, automation and test in Europe*, pages 35–41. ACM, 2000.

Enrique Campos-Nañez, Alfredo Garcia, and Chenyang Li. A game-theoretic approach to efficient power management in sensor networks. *Operations Research*, 56(3):552–561, 2008.

Mihaela Cardei and Ding-Zhu Du. Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, 11(3):333–340, 2005.

Francesco Carrabs, Raffaele Cerulli, Ciriaco D'Ambrosio, Monica Gentili, and Andrea Raiconi. Maximizing lifetime in wireless sensor networks with multiple sensor families. *Computers & operations research*, 60:121–137, 2015.

Francesco Carrabs, Raffaele Cerulli, Ciriaco D'Ambrosio, and Andrea Raiconi. Extending lifetime through partial coverage and roles allocation in connectivity-constrained sensor networks. *IFAC-PapersOnLine*, 49(12):973–978, 2016.

Francesco Carrabs, Raffaele Cerulli, Ciriaco D'Ambrosio, and Andrea Raiconi. An exact algorithm to extend lifetime through roles allocation in sensor networks with connectivity constraints. *Optimization Letters*, 11(7):1341–1356, 2017.

Fabian Castaño, André Rossi, Marc Sevaux, and Nubia Velasco. A column generation approach to extend lifetime in wireless sensor networks with coverage and connectivity constraints. *Computers & Operations Research*, 52:220–230, 2014.

Fabian Castaño, Éric Bourreau, Nubia Velasco, André Rossi, and Marc Sevaux. Exact approaches for lifetime maximization in connectivity constrained wireless multi-role sensor networks. *European Journal of Operational Research*, 241(1):28–38, 2015.

Fabian Castaño, Éric Bourreau, André Rossi, Marc Sevaux, and Nubia Velasco. Partial target coverage to extend the lifetime in wireless multi-role sensor networks. *Networks*, 68(1): 34–53, 2016. doi: 10.1002/net.21682. URL `https://hal.archives-ouvertes.fr/hal-01328424`.

CupCarbon. Cupcarbon. `http://www.cupcarbon.com/`. Accessed: 14-05-2018.

Mohamed Elhoseny, Alaa Tharwat, Xiaohui Yuan, and Aboul Ella Hassanien. Optimizing k-coverage of mobile wsns. *Expert Systems with Applications*, 92:142–153, 2018.

Malka N Halgamuge, Moshe Zukerman, Kotagiri Ramamohanarao, and Hai L Vu. An estimation of sensor energy consumption. *Progress in Electromagnetics Research*, 12:259–295, 2009.

Matthias Handy, Marc Haase, and Dirk Timmermann. Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In *Mobile and Wireless Communications Network, 2002. 4th International Workshop on*, pages 368–372. IEEE, 2002.

Chi-Fu Huang and Yu-Chee Tseng. The coverage problem in a wireless sensor network. *Mobile Networks and Applications*, 10(4):519–528, 2005.

Yanling Jin, Yongsheng Ding, Kuangrong Hao, and Yaochu Jin. An endocrine-based intelligent distributed cooperative algorithm for target tracking in wireless sensor networks. *Soft computing*, 19(5):1427–1441, 2015.

Hsiang-Tsung Kung and Dario Vlah. Efficient location tracking using sensor networks. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 3, pages 1954–1961. IEEE, 2003.

Charly Lersteau, André Rossi, and Marc Sevaux. Robust scheduling of wireless sensor networks for target tracking under uncertainty. *European Journal of Operational Research*, 252(2): 407–417, 2016.

Charly Lersteau, André Rossi, and Marc Sevaux. Minimum energy target tracking with coverage guarantee in wireless sensor networks. *European Journal of Operational Research*, 265 (3):882–894, 2018.

Yuzhen Liu and Weifa Liang. Approximate coverage in wireless sensor networks. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pages 68–75. IEEE, 2005.

Kamal Mehdi, Massinissa Lounis, Ahcène Bounceur, and Tahar Kechadi. Cupcarbon: A multi-agent and discrete event wireless sensor network design and simulation tool. In *7th International ICST Conference on Simulation Tools and Techniques, Lisbon, Portugal, 17-19 March 2014*, pages 126–131. Institute for Computer Science, Social Informatics and Telecommunications Engineering (ICST), 2014.

Matthew J Miller and Nitin H Vaidya. A mac protocol to reduce sensor network energy consumption using a wakeup radio. *IEEE Transactions on mobile Computing*, 4(3):228–242, 2005.

Kgotlaetsile Mathews Modieginyane, Babedi Betty Letswamotse, Reza Malekian, and Adnan M Abu-Mahfouz. Software defined wireless sensor networks application opportunities for efficient network management: A survey. *Computers & Electrical Engineering*, 66:274–287, 2018.

Shaimaa M Mohamed, Haitham S Hamza, and Iman Aly Saroit. Coverage in mobile wireless sensor networks (M-WSN): A survey. *Computer Communications*, 110:133–150, 2017.

Marjan Naderan, Mehdi Dehghan, Hossein Pedram, and Vesal Hakami. Survey of mobile object tracking protocols in wireless sensor networks: a network–centric perspective. *International Journal of Ad Hoc and Ubiquitous Computing*, 11(1):34–63, 2012.

Dipesh J Patel, Rajan Batta, and Rakesh Nagi. Clustering sensors in wireless ad hoc networks operating in a threat environment. *Operations Research*, 53(3):432–442, 2005.

Priyanka Rawat, Kamal Deep Singh, Hakima Chaouchi, and Jean Marie Bonnin. Wireless sensor networks: a survey on recent developments and potential synergies. *The Journal of supercomputing*, 68(1):1–48, 2014.

Yuri N Sotskov, Vyacheslav S Tanaev, and Frank Werner. Stability radius of an optimal schedule: A survey and recent developments. In *Industrial applications of combinatorial optimization*, pages 72–108. Springer, 1998.

Yuri N Sotskov, Alexandre Dolgui, and Marie-Claude Portmann. Stability analysis of an optimal balance for an assembly line with fixed cycle time. *European Journal of Operational Research*, 168(3):783–797, 2006.

Milica Pejanović Đurišić, Zhilbert Tafa, Goran Dimić, and Veljko Milutinović. A survey of military applications of wireless sensor networks. In *Embedded Computing (MECO), 2012 Mediterranean Conference on*, pages 196–199. IEEE, 2012.

Geoffrey Werner-Allen, Konrad Lorincz, Mario Ruiz, Omar Marcillo, Jeff Johnson, Jonathan Lees, and Matt Welsh. Deploying a wireless sensor network on an active volcano. *IEEE internet computing*, 10(2):18–25, 2006.

Li Da Xu, Eric L Xu, and Ling Li. Industry 4.0: state of the art and future trends. *International Journal of Production Research*, 56(8):2941–2962, 2018.

Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.

Ossama Younis and Sonia Fahmy. Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on mobile computing*, 3(4):366–379, 2004.