Robust scheduling for target tracking with wireless sensor network considering spatial uncertainty

Florian Delavernhe[1], André Rossi[3], Marc Sevaux[4]

[1] Université d'Angers, LERIA, F-49045 Angers, France
florian.delavernhe@univ-angers.fr
[2] Université Paris-Dauphine, LAMSADE, UMR 7243, CNRS, F-75016 Paris, France
andre.rossi@dauphine.psl.eu
[3] Université Bretagne Sud, Lab-STICC, UMR 6285, CNRS, F-56321 Lorient, France
marc.sevaux@univ-ubs.fr

1 Introduction

A wireless sensor network (WSN) is a set of sensors, randomly deployed in an area often hard or dangerous to access and without any infrastructure. Hence, the batteries of the sensors are not refillable which limits the lifetime of the network, *i.e.*, how long it can operate. There are several types of sensors for different applications, and we focus in this work on the target tracking. In such applications, the network aims to monitor a set of moving targets (planes, trains, terrestrial vehicles,...), whose spatial trajectories are estimated. It means that, at instant t in the time horizon, we have an estimation of the position of each target. However, this estimation may not be accurate, and the difficulty of the problem is to cover the targets considering the highest possible deviation from their estimated trajectories. Moreover, in order to preserve energy in the network for future mission, at most one sensor per target should be used at any time. Finally, all the data collected by the sensors have to be transmitted to a base station. The problem is to find a robust schedule to continuously monitor the targets and to transfer the data. This schedule is robust because it has to maximize a spatial stability radius, such that, it stays feasible as long as the targets are not deviated for more than the value of the stability radius from their estimated position. The targets are covered at every instant t as long as, they are located in the disc of radius equals to the stability radius and centered on the estimated position of the target. In this work, we propose (i) a discretization method on the geometric data, (ii) two upper bounds on the value of the stability radius, and (iii) a method that uses the discretized data and the upper bounds to compute a robust schedule.

2 Definition of the problem

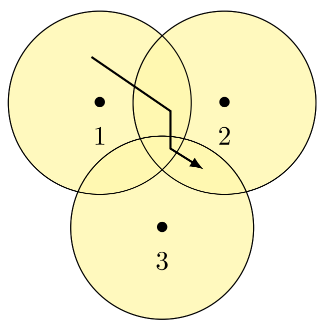
Let J be the set of the n targets that should be monitored. Each target j has an estimated trajectory such that at instant t, the estimated position of j is

$P_j(t)$. For each target, its estimated trajectory is represented using a collection of waypoints. The trajectory is a sequence of segments between the waypoints. *i.e.*, a trajectory is a suite of segments. The network is a set I of m sensors and a base station where the data is sent. A sensor can receive or transmit data only with the base station or another sensor if it is in its neighborhood $N(i)$, *i.e.*, if the distance is less than the communication range R_C. For a target j and a instant t, we define $\rho_j(t, R)$ as the set of all points that are in the disc of radius R and centered on $P_j(t)$. There are three types of energy consumption for a sensor:

- monitoring a target (p^S Watts),
- transmitting data (p^T Watts),
- receiving data (p^R Watts).

3 Discretization

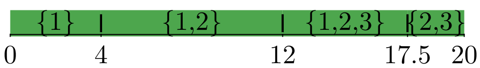
The following figure is an example, where three sensors (1,2 and 3) are deployed to cover a single target (the black arrow) in the horizon of time $H = [0, 20]$:



Discretization is the necessary transformation of the geometric data of the problem into a set of discretized data that can be used for modeling and solving the problem. The aim is to represent the trajectory of each target as a set of time windows with a set of candidate sensors associated to each window, that can monitor the target during the entire time window. Let's call a face f a set of spatial points that are covered by the same set of sensors $S(f)$. Monitoring a target j at time t is therefore monitoring all the faces where j can possibly be. Hence, for a stability radius R, we need to cover all the faces with a non empty intersection with $\rho_j(t, R)$. The intersection of all these faces defines the face to cover (if the intersection is empty, then the target cannot be covered). For example, if a target needs to be covered in the face $\{1\}$ and the face $\{1, 2\}$, the set of candidate sensors is $\{1\} \cap \{1, 2\} = \{1\}$.

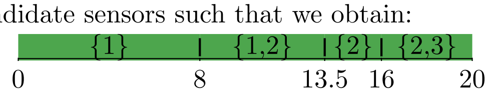
Thus, the trajectory of a target j is represented as a sequence of faces to be covered, associated to the set K_j of time windows. The time windows are delimited by time instants called ticks, such that a tick is either *entering* which means that a new sensor is candidate, or *leaving* when a sensor is no more candidate.

With a $R = 0$, the time windows and candidate sensors over the horizon of time, in our example, are:



When increasing the stability radius, the spatial uncertainty covered is increasing. It delays the instant where a sensor is guaranteed to cover a target and is advancing the moment where a sensor is no more candidate to cover the target. The evolution of a tick depends on the segment of the estimated spatial trajectory where it is located. Therefore, increasing the stability radius corresponds to moving the ticks and is modifying the length of the time windows. This may change the set of candidate sensors when two ticks are equal.

In our example, with a certain value of R, the tick moved and changed the candidate sensors such that we obtain:



Another difficulty is that, for each target j, there are instants t where time window can appear when the stability radius reaches a certain value r. Indeed, it possible that some points in $\rho_j(t, r)$ are no longer covered by a sensor that was initially covering $P_j(t)$ even if the corresponding tick did not move. It means that a new time window is appearing in K_j at t when the stability radius reaches this specific value of R. In our example, a time window is appearing at the time corresponding to the last estimated waypoint if the stability radius is great enough to allow the target to be out of the range of sensor 2.

In order to find all these time windows, we need to look at each intersection of the segments of the estimated trajectory of a same target, *i.e.*, when the target is changing of direction. For each sensor covering an extremity e of a segment, there is a potential time window w. It appears at the time instant the target is estimated to be at e, when the stability radius is equal to the sensing range of the sensor minus the distance between the sensor and e. Indeed, a segment is always leaving the range of a sensor starting by one of its extremity or by the estimated frontier between two faces (initial ticks).

To conclude, an increase of the stability radius is modifying the length of the time windows, adding new time windows, and adding new or changing the sets of candidate sensor. All of this issues are depending on the coordinates of the sensors and the segments of the trajectories.

4 Upper bounds on the stability radius

Two upper bounds were found and implemented for our solving method. The first bound searches the first value of R that creates an empty face to cover. The stability radius cannot exceed this value and it always corresponds to either the intersection between two ticks, or the apparition of a new time window. First, we need to compute, for each extremity, the last time window that will appear. Because each of the time window is corresponding to one sensor no longer candidate for this extremity, the last time window is corresponding to an empty face. These values are easily computed with the distance between the extremity and the sensors. The second part of this bound is, for each segment, to find the lowest value of R that corresponds to an intersection between two ticks such that the intersection point is no longer covered by any other sensor when the intersection occurs. These values are computed using the position of the sensors and the segments' coordinates.

The second bound computes the value of R such that there is not enough energy in a set of candidate sensors to cover the length of the corresponding time window. In that purpose, for each face in the estimated trajectories of the targets, we compute the total sum of the batteries of the sensors in range and the total energy needed. We order these sensors by increasing values of R for which they are no longer in range of any point of the trajectories in the face. Afterwards, we remove the batteries of these sensors, one by one, until the sum of the batteries of the remaining sensors are not enough to cover the face.

5 Solving Process

Because the time windows and the corresponding sets of candidate sensors are depending on the value of the stability radius, a single linear program cannot be solved to maximize R. Therefore, we use a dichotomy method on the values of R which modify these sets. For each value tested by the dichotomy, the following satisfactory linear program is solved:

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

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

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

$$\delta \geq 0 \qquad (4)$$

$$x_{jik} \geq 0 \quad \forall j \in J, k \in K_j, i \in S_j(k) \qquad (5)$$

$$f_{ii'} \geq 0 \quad \forall i \in I, i' \in N(i) \qquad (6)$$

With Δ_k^j the size of the k-th time window of the sensor j and x_{jik} the time sensori i is monitoring j in its k-th time window. Constraints (1) correspond to the limitations of the batteries. Constraints (2) are flow constraints, where the data collected and received by a sensor is transmitted. Constraints (3) sets that the sum of the activities of the sensors in a time window is equal to its length.

References

1. Lersteau, C., Rossi, A., Sevaux, M. (2018). Minimum energy target tracking with coverage guarantee in wireless sensor networks. European Journal of Operational Research, 265(3), 882-894.