Controlled Mobility Sensor Networks for Target Tracking Using Ant Colony Optimization

Farah Mourad, Hicham Chehade, Hichem Snoussi, *Member*, *IEEE*,
Farouk Yalaoui, Lionel Amodeo, and Cédric Richard, *Senior Member*, *IEEE*

Abstract—In mobile sensor networks, it is important to manage the mobility of the nodes in order to improve the performances of the network. This paper addresses the problem of single target tracking in controlled mobility sensor networks. The proposed method consists of estimating the current position of a single target. Estimated positions are then used to predict the following location of the target. Once an area of interest is defined, the proposed approach consists of moving the mobile nodes in order to cover it in an optimal way. It thus defines a strategy for choosing the set of new sensors locations. Each node is then assigned one position within the set in the way to minimize the total traveled distance by the nodes. While the estimation and the prediction phases are performed using the interval theory, relocating nodes employs the ant colony optimization algorithm. Simulations results corroborate the efficiency of the proposed method compared to the target tracking methods considered for networks with static nodes.

Index Terms—Ant colony, controlled mobility, interval analysis, network coverage, state estimation, target tracking.

1 Introduction

Mobile Sensor Networks (MSN) are networks composed of a large number of wireless devices having sensing, processing, communication, and movement capabilities [1], [2], [3]. The main constraint of sensor nodes is their limited energy resources since their batteries are nonrenewable. One important factor is thus to reduce the energy consumption of the sensors in order to increase the lifetime of the network. One can distinguish between two types of mobility in MSN: the uncontrolled (also called passive) mobility, where sensors are moved in an uncontrollable manner, and the controlled mobility, where sensors are moved in response to internal or external commands. Passive mobility makes the use of MSN more challenging since sensor nodes need to be relocated continuously; whereas in controlled mobility, one could take advantage of the mobility of the nodes to improve the accuracy of the sensed data in the network.

MSN have a variety of applications in different fields, such as military and environment monitoring [4], [5], [6]. One interesting application of MSN is target tracking. It consists of estimating instantly the position of a moving target. It is of great importance in surveillance and security especially in military applications. This problem has been mainly considered for networks having static nodes [7], [8], [9]. For instance, in [8], authors present particle filtering methods for target tracking using binary sensors, whereas in [9], a clustering algorithm using the variational filter is proposed. However, when sensors are able to move, it is important to take advantage of their mobility in order to improve the position estimation. This contribution focuses on target tracking in MSN where nodes have a controlled mobility. Different techniques have been proposed to manage the mobility of the nodes [10], [11], [12]. These techniques have mainly focused on upgrading the topology of the network, improving the area coverage or increasing the lifetime of the network, etc. A few methods have been developed for target tracking in MSN [13], [14]. Researchers in [13] have proposed a mobility management scheme based on the Bayesian estimation theory. Nodes are thus able to move to a new position chosen within a set of candidate locations, being at one step away from the current location. The movement decision is made upon whether the new location will improve the tracking quality or not. Note that many assumptions are made in this method such that both the target and the sensor nodes are supposed having constant velocities. The number of candidate locations is limited, as well, in order to reduce the complexity of the method. In a different scenario, researchers in [14] have considered the problem of a mobile target, called mouse, trying to avoid detection by mobile sensor nodes, called cats.

In this paper, we propose a novel strategy for managing sensors mobility, aiming at improving the tracking of a single target. The method consists of four consecutive phases that iterate at each time step as follows:

1. Estimating the current position of the target,
2. Predicting the next-step position of the target using current and previous position estimates,
3. Computing a set of new locations to be taken by the mobile nodes in the way to improve the estimation process,

- *F. Mourad, H. Chehade, H. Snoussi, F. Yalaoui and L. Amodeo are with the Institut Charles Delaunay - STMR (UMR CNRS 6279), Université de Technologie de Troyes, Pôle ROSAS, 12 rue Marie Curie, BP 2060, 10010 Troyes cedex, France. E-mail: {farah.mourad, hicham.chehade, hichem.snoussi, farouk.yalaoui, lionel.amodeo}@utt.fr.*
- *C. Richard is with the Laboratoire Fizeau (UMR CNRS 6525), Université de Nice Sophia-Antipolis, Parc de Valrose, 06108 Nice cedex 02, France. E-mail: cedric.richard@unice.fr.*

Manuscript received 30 Apr. 2010; revised 20 June 2011; accepted 30 June 2011; published online 22 July 2011.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2010-04-0201. Digital Object Identifier no. 10.1109/TMC.2011.154.

4. Assigning each mobile node one new location within the computed set using the ant colony optimization algorithm (ACO).

The estimation phase is performed using interval analysis [15]. The target positions are thus defined as two-dimensional intervals including the real positions. Predicting the next-step position leads to a region of interest to be covered. The relocating phase consists thus of moving the nodes in order to improve the estimation phase while minimizing the energy consumption. The optimization phase is performed using the Ant Colony Optimization algorithm [16], [17]. Being a metaheuristic method, the ACO is able to solve efficiently complex operational research problems. Due to its stochastic nature, it affords the combinatorial explosion in the number of possible solutions. Besides the energy limitation, one important constraint is to maintain the total coverage of the network while moving the sensor nodes. In other words, the whole monitoring area should be covered by sensors in order to be robust to any other intruders. For this reason, we use two types of sensors: static and mobile nodes. While mobile sensors are moved to improve the quality of target tracking, static nodes are uniformly distributed in order to ensure a continuous coverage of the network independently of the movement of the mobile ones.

The rest of the paper is organized as follows: In Sections 2 and 3, we describe the estimation and the prediction phases, respectively. Section 4 introduces the relocation strategy including the ant colony optimization algorithm. Simulation results are given in Section 5. Section 6 concludes the paper.

2 Estimation of the Target Position

Target tracking consists of finding the coordinates of the moving target at each time step. In the following, we first state the problem. We then describe the proposed algorithm.

2.1 Problem Statement

The proposed method consists of collecting connectivity measurements to estimate the target position [18], [19]. A connectivity measurement related to the ith sensor node is given by one-bit information as follows:

$$y_i(t) = \begin{cases} 1, & \text{if sensor } i \text{ detects the target;} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The generation of such measurements depends on the status of the target. In fact, two cases could be encountered:

- **Case 1**—The target is active, and thus, it keeps on communicating with the sensors. In other words, the target broadcasts continuously signals in the network. Assume that all these signals are emitted with the same initial power. According to the Okumura-Hata model [20], [21], the power of a signal decreases monotonically with the increase of the distance traveled by this signal. Let the sensing range be a disk having r as radius and let ρ_r be the power of a target signal corresponding to a traveled distance equal to r. Then, all sensor nodes receiving target signals with powers larger than ρ_r are located at distances to the target less than r. These sensors are assumed to detect the target and thus, they generate connectivity measurements equal to 1. Only the sensors detecting the target communicate their measurements to the fusion center. Connectivity measurements corresponding to other sensors are assumed to be 0.
- **Case 2**—The target is passive and thus noncooperating. In this case, object-detecting systems (such as radar sensors) could be used to detect whether the target is within the sensing range of the sensors. Then, each node detecting the target generates a connectivity measurement equal to 1 and communicates it to the fusion center. Similarly to the first case, connectivity measurements of other sensors are set to 0 at the fusion center.

Let I be the set of indices of sensors detecting the target. Having a circular sensing range of radius r, then for each sensor i, $i \in I$, the distance between the target and the considered sensor is less than r. This leads us to a set of observation equations as follows:

$$(x_1(t) - s_{i,1}(t))^2 + (x_2(t) - s_{i,2}(t))^2 \leq r^2, \quad i \in I, \quad (2)$$

where $\boldsymbol{x}(t) = (x_1(t), x_2(t))$ is the unknown coordinates vector of the target at time t and $\boldsymbol{s}_i(t) = (s_{i,1}(t), s_{i,2}(t))$ is the coordinates vector of the ith sensor at time t. Note that, if the sensing range is not homogenous, its maximal value could be considered in the observation equations. The target falls thus within the overlapping region of all observation disks centered on the sensors detecting it.

2.2 Interval-Based Estimation

In order to solve the estimation problem given in (2), we use the interval analysis. In the following we briefly recall the basic tools of interval analysis. We then introduce the interval-based algorithm.

2.2.1 Interval Analysis

A real interval, denoted $[x]$, is a closed subset of $\mathbb{R}$ given as follows [22]:

$$[x] = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\}, \quad (3)$$

where $\underline{x}$ and $\overline{x}$ are the lower and upper scalar endpoints of the interval, respectively. $[x]$ could also be defined by its center and its width given by $\mathcal{C}([x]) = \frac{\underline{x}+\overline{x}}{2}$ and $\mathcal{W}([x]) = \overline{x} - \underline{x}$, respectively. A multidimensional interval of $\mathbb{R}^n$, also called box, is given by the cartesian product of n real intervals as follows:

$$[\boldsymbol{x}] = [x_1] \times \cdots \times [x_n]. \quad (4)$$

An interval has a dual nature as sets and real numbers. The interval theory takes advantage of this duality to extend all arithmetic and set operations to intervals [15]. Some definitions of interval operators are given in Table 1. Note that all operators could be extended to boxes.

2.2.2 Localization Algorithm

The key idea of the method consists of considering the target position as a two-dimensional box [18], [19]. In other words, the proposed method aims at computing the minimal box that includes all possible solutions of the

TABLE 1
Definitions of Some Operators Applied to Intervals

Operators	Definitions
$+$	$[x]+[y]=[\underline{x}+y,\overline{x}+\overline{y}]$
$-$	$[x]-[y]=[\underline{x}-\overline{y},\overline{x}-\underline{y}]$
$\cap$	$[x]\cap[y]=[\max\{\underline{x},\underline{y}\},\min\{\overline{x},\overline{y}\}]$
$\sqcup$	$[x]\sqcup[y]=[\min\{\underline{x},\underline{y}\},\max\{\overline{x},\overline{y}\}]$

problem. In this way, the target position is a rectangular area including the unknown location of the target and all incertitude over its value.

Resolving the estimation problem using the interval theory consists thus of expressing all observation equations in the interval framework as follows:

$$[[x_1](t)-s_{i,1}(t)]^2+[[x_2](t)-s_{i,2}(t)]^2\subseteq[0,r^2],\quad i\in I,\quad (5)$$

where $[\boldsymbol{x}](t)=[x_1](t)\times[x_2](t)$ is the target boxed position at time t. The problem is then defined as a constraint satisfaction problem. An initial domain, for instance the whole deployment area, is thus contracted in order to obtain the smallest box including the exact scalar solution. The algorithm used to perform the contraction is called the Waltz contractor [15], [23]. It is a forward-backward algorithm that iterates all constraints without any prior order until no contraction is possible. Each equation (5) yields two constraints expressing each coordinate as a function of the other as follows:

$$\begin{cases}[x_1](t)\subseteq[s_{i,1}(t)-\overline{b}_{i,1},s_{i,1}(t)+\overline{b}_{i,1}]\\ [x_2](t)\subseteq[s_{i,2}(t)-\overline{b}_{i,2},s_{i,2}(t)+\overline{b}_{i,2}]\end{cases},\quad i\in I,\qquad (6)$$

where

$$[b_{i,1}]=\sqrt{r^2-[[x_2](t)-s_{i,2}(t)]^2}$$

and $[b_{i,2}]=\sqrt{r^2-[[x_1](t)-s_{i,1}(t)]^2}$. Equations (6) are iterated using the Waltz algorithm in order to reduce the initial domain, denoted $[X_1]\times[X_2]$, as much as possible. The estimation algorithm given at time t is shown in Algorithm 1, where $W([x])=\overline{x}-\underline{x}$ is the width of the real interval $[x]$. Fig. 1 shows an illustration of the estimation phase.

Algorithm 1. Estimation algorithm
Input: Indices of sensors observing the target I;
Output: Target coordinates $[x_1](t)$ and $[x_2](t)$;
Initialization: $[x_1](t)=[X_1]$, $[x_2](t)=[X_2]$,
$\mathcal{A}=\mathcal{W}([x_1](t)).\mathcal{W}([x_2](t))$, $\mathcal{A}_{old}=\mathcal{A}+1$;
while $\mathcal{A}<\mathcal{A}_{old}$ **do**
  $\mathcal{A}_{old}=\mathcal{A}$;
  for $i\in I$ **do**
    $[b_{i,1}](t)=\sqrt{r^2-[[x_2](t)-s_{i,2}(t)]^2}$;
    $[x_1](t)=[x_1](t)\cap[s_{i,1}(t)-\overline{b}_{i,1},s_{i,1}(t)+\overline{b}_{i,1}]$;
    $[b_{i,2}](t)=\sqrt{r^2-[[x_1](t)-s_{i,1}(t)]^2}$;
    $[x_2](t)=[x_2](t)\cap[s_{i,2}(t)-\overline{b}_{i,2},s_{i,2}(t)+\overline{b}_{i,2}]$;
  end
  $\mathcal{A}=\mathcal{W}([x_1](t)).\mathcal{W}([x_2](t))$;
end

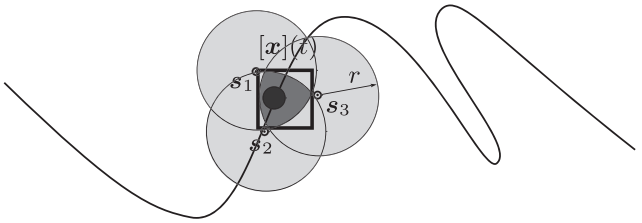

Fig. 1. An illustration of the estimation phase.

3 Prediction of the Target Next-Step Position

Let $x(1),\ldots,x(t)$ be all available estimated positions of the target. Then, a kth order prediction model is given as follows:

$$\widehat{\boldsymbol{x}}(t+1)=\boldsymbol{f}(\boldsymbol{x}(t),\ldots,\boldsymbol{x}(t-k)),\qquad (7)$$

where $\boldsymbol{f}$ is the prediction function and $\widehat{\boldsymbol{x}}(t+1)$ is the predicted position of the target regarding time $t+1$. All available information about the target motion could be used to refine the prediction model. In this paper, we propose a second order prediction model as follows:

$$\widehat{\boldsymbol{x}}(t+1)=\boldsymbol{x}(t)+\Delta t.\boldsymbol{v}(t)+\frac{\Delta t^2}{2}.\gamma(t),\qquad (8)$$

where Δt is the time period falling between two following time-steps and $\boldsymbol{v}(t)$ and $\gamma(t)$ are the respective estimate vectors of the instant velocity and the instant acceleration at time t given by

$$\begin{cases}\boldsymbol{v}(t)=\dfrac{\boldsymbol{x}(t)-\boldsymbol{x}(t-1)}{\Delta t},\\ \gamma(t)=\dfrac{\boldsymbol{v}(t)-\boldsymbol{v}(t-1)}{\Delta t}.\end{cases}\qquad (9)$$

In the interval framework, the prediction model is formulated as follows:

$$[\widehat{\boldsymbol{x}}](t+1)=[\boldsymbol{x}](t)+\Delta t.[\boldsymbol{v}](t)+\frac{\Delta t^2}{2}.[\gamma](t),\qquad (10)$$

where $[\widehat{\boldsymbol{x}}](t+1)$ is the predicted position box of the target, $[\boldsymbol{v}](t)=[[\boldsymbol{x}](t)-[\boldsymbol{x}](t-1)]/\Delta t$ and $[\gamma](t)=[[\boldsymbol{v}](t)-[\boldsymbol{v}](t-1)]/\Delta t$. Using intervals, the prediction phase yields a box including the next-step position of the target.

4 Relocation of the Mobile Sensors

The goal of the method consists of moving the sensors in an energy-aware manner in order to better cover the prediction box, also called area of interest. The relocation should be performed without leaving any uncovered area. In the following, we first address the coverage problem. We then set the new locations that should be taken by the nodes. We finally introduce the positioning of the nodes using the ant colony optimization algorithm.

4.1 Coverage Problem

One main constraint of sensors relocation is to maintain network coverage. Let r be the sensing range of the sensors. Then, each sensor covers an r-disk of the deployment area.

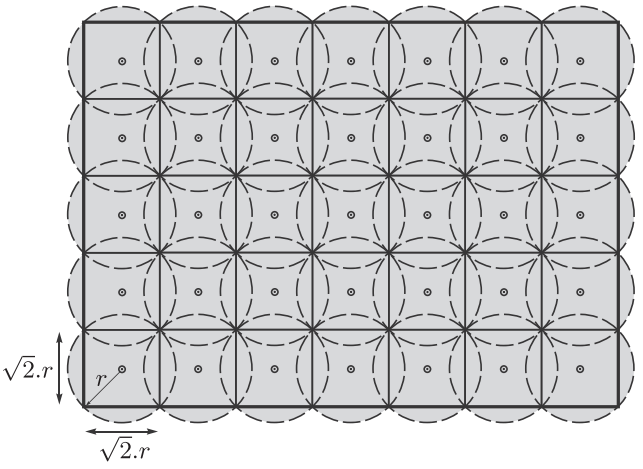

Fig. 2. An illustration of the arrangement of the fixed nodes.

Moving the nodes may yield uncovered regions, which makes the network exposed to intruders. We propose to use hybrid sensor nodes to address this problem: mobile and static nodes. While mobile nodes are moved to improve the tracking, static nodes are used to ensure continuous coverage. In order to have total coverage, the whole deployment area should be filled with the minimal number of sensing disks without leaving any uncovered region. Many algorithms, based on the disk packing theory, have been proposed for solving such problems [24], [25], [26]. These algorithms aim at packing equal disks, in an optimal manner, into a square area. In this paper, we propose a simple technique, using the squares inscribed in the sensing disks having $\sqrt{2}.r$ as side. Putting these squares side by side leads to uniformly arranged static nodes. Let $[\underline{X}_1, \overline{X}_1] \times [\underline{X}_2, \overline{X}_2]$ be the deployment area. The static nodes positions are thus given by the combinations of the following coordinates:

$$\begin{cases} S_{(f,1),p} = \underline{b}_1 + \dfrac{\sqrt{2}}{2}.r + (p-1).\sqrt{2}.r, & 1 \leq p \leq K_1, \\ S_{(f,2),q} = \underline{b}_2 + \dfrac{\sqrt{2}}{2}.r + (q-1).\sqrt{2}.r, & 1 \leq q \leq K_2, \end{cases} \tag{11}$$

where $\underline{b}_1 = \underline{X}_1 - \frac{K_1.\sqrt{2}.r-(\overline{X}_1-\underline{X}_1)}{2}$, $\underline{b}_2 = \underline{X}_2 - \frac{K_2.\sqrt{2}.r-(\overline{X}_2-\underline{X}_2)}{2}$, $K_1 = \frac{\overline{X}_1-\underline{X}_1}{\sqrt{2}.r}$, and $K_2 = \frac{\overline{X}_2-\underline{X}_2}{\sqrt{2}.r}$. If K_1 or K_2 are not integer, one should take the closest upper integer value. Static nodes disks are thus symmetric with respect to the total deployment area. The total number of static nodes required to cover the whole area is equal to $K_f = K_1.K_2$. Fig. 2 shows an illustration of the proposed distribution of the static nodes. Note that, while only mobile nodes are used in the relocation phase, both static and mobile nodes are used in the estimation phase.

4.2 Definition of Sensors New Locations

In this section, we propose a strategy to define a set of locations to be taken by the mobile nodes. The goal of this strategy is to cover an area of interest in the best way. We propose thus to use all mobile nodes closer than a certain distance around the area of interest. Only close nodes are thus considered in order to limit the traveled distance and so to reduce the energy consumption. Let K_m be the number of the considered mobile nodes. Then, the number of positions to be defined is equal to K_m.

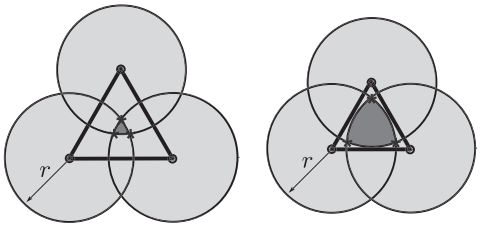

Fig. 3. Dependence of the estimation accuracy to the lengths of the triangle sides.

The proposed method is based on the triangulation principle. Consider only three nodes and a single point to be localized. The triangulation-based idea consists of constructing an equilateral triangle with the sensors. The barycenter of the triangle should fall at the point of interest. Let r be the sensing range of the sensors. Using these sensors in the estimation phase leads to the overlapping area of the sensing r-disks. Note that the overlapping region gets smaller as the triangle sides become larger. In other words, the resolution is related to the distances falling between the nodes. An illustration is given in Fig. 3. Let us denote by sensor triangle, or simply Δ_S, the triangle composed of sensors and by target triangle, or simply Δ_T, the triangle included in the intersection region of the sensing disks. Assume that $\sigma_S = \lambda_S.r$ and $\sigma_T = \lambda_T.r$ are the sides lengths of the triangles Δ_S and Δ_T, then

$$\lambda_T = \frac{\sqrt{3.(4-\lambda_S^2)} - \lambda_S}{2} \tag{12}$$

(see the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2011.154). Knowing that the target is going to fall within a specific box $[\widehat{\boldsymbol{x}}](t+1)$ in the following step leads us to cover at best the considered area of interest. The key idea consists thus of arranging the nodes in the way to fill $[\widehat{\boldsymbol{x}}](t+1)$ with target triangles Δ_T. However, filling the box with independent triangles having each its own sensors leads us either to use large triangles or to increase the number of considered sensors. In fact, in such a case, one is able to define $\frac{K_m}{3}$ independent triangles. While the number of mobile sensors is limited, enlarging the target triangles induces a loss in the accuracy of the estimation.

As a consequence, we propose to use structures of Δ_S where mobile sensors are rigidly linked. All target triangles generated by a given structure are thus fixed one to the other. This approach needs less sensors than the one above for the same number of target triangles. An example of a structure of ten sensors is illustrated in Fig. 4. Such a structure leads to 10 target triangles which would need 30 independent sensors. Note that we show in dark gray the areas covered by at least three sensors, whereas the whole coverage zone of the structure is shown in light gray. Using triangle structures, one is able to cover every single point of the box of interest with at least three sensors. For this reason, σ_S is set to r leading to a σ_T equal to r too. In such a case, all the points of the box are covered in the same manner leading to a guaranteed structure. An illustration is given in Fig. 5a. Let K_1 be the number of the sensors needed

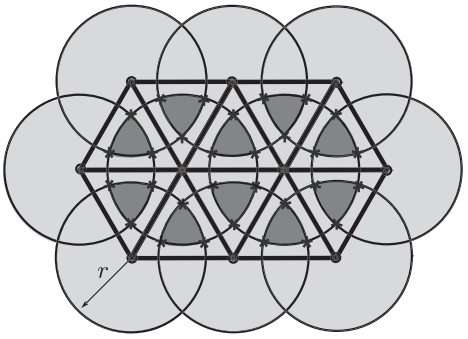

Fig. 4. An example of a structure of 10 mobile sensors.

in the first down row and let K_2 be the number of rows. Having the type of structures of Fig. 5a, rows having odd indices are thus composed of K_1 sensors whereas those of even indices have $K_1 + 1$ sensors. In order to guarantee a coverage of the whole box even for points close to the borders, K_1 and K_2 are set as follows:

$$\begin{cases} K_1 = \text{int}u\left\{\dfrac{\widehat{\mathcal{W}}_1}{\sigma_S}\right\} + 1, \\ K_2 = \text{int}u\left\{\dfrac{\widehat{\mathcal{W}}_2}{\frac{\sqrt{3}}{2}.\sigma_S}\right\} + 1, \end{cases} \tag{13}$$

where $\widehat{\mathcal{W}}_1$ and $\widehat{\mathcal{W}}_2$ are the widths of $[\widehat{x}_1](t+1)$ and $[\widehat{x}_2](t+1)$, respectively, $\frac{\sqrt{3}}{2}.r$ is the height of the sensor triangle and $\text{int}u\{x\}$ is the smallest integer equal or greater than x. In a different manner, one could choose larger sensor triangles leading to more resolution at specific zones but less guarantee in the total prediction area. The target triangles are thus smaller but the box of interest will not be wholly covered by at least three sensors. Note that for $\sigma_S = \sqrt{3}.r$, $\sigma_T = 0$ and thus the target triangle becomes a single point. Then, for more resolution, one should choose σ_S higher than r but less than $\sqrt{3}.r$. An illustration is given in Fig. 5b. Once σ_S is chosen, the number of sensors in a row or a column are given by

$$\begin{cases} K_1 = \text{int}l\left\{\dfrac{\widehat{\mathcal{W}}_1}{\sigma_S}\right\} + 1, \\ K_2 = \text{int}u\left\{\dfrac{\widehat{\mathcal{W}}_2}{\frac{\sqrt{3}}{2}.\sigma_S}\right\} + 1, \end{cases} \tag{14}$$

where $\text{int}l\{x\}$ is the highest integer equal or lower than x. It yields 1 if x is less than 1. Note that, in this approach, we choose the lower integer for K_1 to have all target triangles falling the most probably inside the area of interest.

The total number of sensors needed to define the structure in both cases is thus equal to $K_s = K_1.\frac{K_2+\delta}{2} + (K_1+1).\frac{K_2-\delta}{2}$ where $\delta = 1$ if K_2 is odd and 0 otherwise. The positions of the sensors are then given by the combination of the following coordinates:

$$\begin{cases} S_{(m,1),i} = \begin{cases} \widehat{\underline{b}}_{o,1} + (i-1).\sigma_S, & \text{if } j \text{ is odd } (1 \le i \le K_1), \\ \widehat{\underline{b}}_{e,1} + (i-1).\sigma_S, & \text{if } j \text{ is even } (1 \le i \le K_1+1), \end{cases} \\ S_{(m,2),j} = \widehat{\underline{b}}_2 + (j-1).\dfrac{\sqrt{3}}{2}.\sigma_S, \quad 1 \le j \le K_2, \end{cases} \tag{15}$$

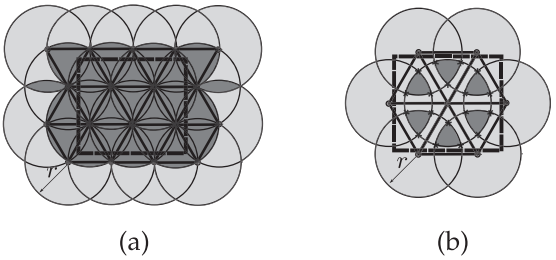

Fig. 5. An example of a guaranteed structure in (a) and a more accurate but less guaranteed structure in (b).

where $\widehat{\underline{b}}_{o,1} = \widehat{\underline{x}}_1 - \frac{K_1.\sigma_S - \widehat{\mathcal{W}}_1}{2}$, $\widehat{\underline{b}}_{e,1} = \widehat{\underline{x}}_1 - \frac{(K_1+1).\sigma_S - \widehat{\mathcal{W}}_1}{2}$, and

$$\widehat{\underline{b}}_2 = \widehat{\underline{x}}_2 - \frac{K_2.\frac{\sqrt{3}}{2}.\sigma_S - \widehat{\mathcal{W}}_2}{2}.$$

The positions we obtain are thus symmetric with respect to the area of interest. Note that in the first approach, if $K_s > K_m$, one could keep the closest K_m positions to the center, whereas when K_s is less than K_m, one could either move only K_s sensors or add $K_m - K_s$ positions uniformly deployed between the K_s ones. For instance, barycenters of the triangles of the structure could be chosen. In the second approach, if $K_s > K_m$, one could also keep the closest K_m positions to the center. When K_s is less than K_m, the barycenters of the triangles of the structure could also be defined as additional positions. If some sensors are remaining, then they could have random positions. Note that according to the plot, less sensors are needed to cover a given box while using the second approach.

4.3 Optimization of Sensors Positioning Using Ant Colony

Having the set of positions that must be taken by the sensors, one should assign each sensor one position within the set while minimizing the traveled distance of the nodes. The problem is thus defined as an optimization algorithm that is solved using the ACO. In the following, we first introduce the ACO. We then apply it to the relocation problem.

4.3.1 Ant Colony Optimization Algorithm

The ACO is a probabilistic method for solving complex computational problems. This algorithm was first developed to solve the traveling salesman problem [16]. It has been applied efficiently afterwards in different fields such as quadratic assignment problems [27], vehicle routing [28], or assembly lines design [29]. The main idea of ACO consists of imitating the behavior of real ants in their way to find the shortest path to get food sources. A path is thus generated according to two elements: a chemical substance called pheromone and the visibility of the ant. Let $f(x_1, \ldots, x_n)$ be a function of n variables whose values are taken from a specific set S. Optimizing f consists of finding the n-permutation of $(x_1, \ldots, x_n)$ over all possible permutations that optimizes the function f. In such problems, the function f is called objective function or fitness function, whereas $x_1, \ldots, x_n$ are called the decision variables. Let m be the cardinal of S, then the number of all possible n-permutations is equal to $\frac{m!}{(m-n)!}$ with $m!$ being the factorial of m. Evaluating all solutions requires too much computational time especially for large-size problems. In

such cases, using optimization algorithms such as ACO becomes crucial to reduce the time of computation.

Starting with an initial solution, ACO moves toward optimal solutions using an efficient memory-based search technique. The generation of solutions basically employs two parameters: visibility and pheromone. These parameters correspond to a priori and a posteriori information about the solutions, respectively. While visibility remains unchanged, pheromone is modified during the optimization process according to solutions evaluation. Technically, the ACO algorithm considers a fixed number of ants K_a, each of which generating one solution at every iteration. Solutions are thus encoded by assigning each decision variable, one after the other, a specific value of S. Let $S = \{a_1, \ldots, a_m\}$. An ant k, $k \in \{1, \ldots, K_a\}$, having assigned the variable x_i a value a_j, will assign the variable x_{i+1} the value a_l chosen as follows:

$$l = \begin{cases} \text{argmax}_{u \in J_k(i)} \{\tau^{\alpha}_{(i),j,u}.\eta^{\beta}_{(i),j,u}\}, & \text{if } q \leq q_0, \\ l^*, & \text{otherwise,} \end{cases} \quad (16)$$

where:

- $\tau_{(i),j,u}$: The amount of pheromone of setting $x_{i+1} = a_u$ knowing that $x_i = a_j$,
- $\eta_{(i),j,u}$: The ant visibility or desirability to choose the value a_u for x_{i+1} going from a_j,
- α and β: Parameters defining the relative importance of pheromone versus visibility,
- q: A random number generated at each iteration between 0 and 1,
- q_0: A parameter ($0 < q_0 < 1$) defining the relative importance of exploitation against exploration,
- $J_k(i)$: The indices set of values of S not yet assigned by ant k,
- l^*: A random variable computed according to a specific distribution P^*.

Note that the first variable is assigned random values of S. The probability distribution P^* is defined as follows:

$$P^*(u) = \begin{cases} \frac{\tau^{\alpha}_{(i),j,u}.\eta^{\beta}_{(i),j,u}}{\sum_{v \in J_k(i)} \tau^{\alpha}_{(i),j,v}.\eta^{\beta}_{(i),j,v}}, & \text{if } u \in J_k(i), \\ 0, & \text{otherwise.} \end{cases} \quad (17)$$

Using the pseudorandom rule, the next state is either deterministic depending of η and τ ($q \leq q_0$) or random, chosen with the probability distribution P^* ($q > q_0$). The method provides thus a direct way to balance between exploration of new states and exploitation of a priori and accumulated knowledge.

At the end of each iteration, i.e., when all ants have completed a solution, the pheromone of the entire system evaporates. The pheromone corresponding to all solutions is thus updated locally as follows:

$$\tau_{(i),j,l} = (1 - \rho).\tau_{(i),j,l} + \rho.\tau_0, \quad (18)$$

where $1 \leq i \leq K_m - 1$, $\{j, l\} \in \textit{ant solutions}$, τ_0 is the initial pheromone amount, and ρ is a parameter falling between 0 and 1. Solutions are then evaluated, each of them yielding a value of the fitness function. Only the best solution is used to update its corresponding pheromone as follows:

$$\tau_{(i),j,l} = (1 - \rho).\tau_{(i),j,l} + \rho.\Delta\tau, \quad (19)$$

where $\{j, l\} \in \textit{the best solution}$ and $\Delta\tau$ is a function of the best solution. This mechanism is called the global update.

4.3.2 Sensors Relocation

The relocation problem consists of minimizing the total distance traveled by the nodes while moving to their new positions. The fitness function is thus equal to the sum of the distances traveled by the moving nodes, whereas the decision variables are the sensors coordinates. These variables take their values within the set of the positions already defined. Let $\boldsymbol{s}_1(t), \ldots, \boldsymbol{s}_{K_m}(t)$ be the sensors coordinates at time t. Then, $\boldsymbol{s}_1(t+1), \ldots, \boldsymbol{s}_{K_m}(t+1)$ are their new coordinates. The fitness function is thus computed as follows:

$$f(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{K_m}) = \sum_{i=1}^{K_m} \|\boldsymbol{s}_i(t), \boldsymbol{s}_i(t+1)\|, \quad (20)$$

where $\|\bullet, \bullet\|$ is the euclidian distance operator. Let $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{K_m}$ be the set of positions. Then, the number of all possible solutions is equal to $K_m!$.

Resolving the problem using ACO consists of iterating the set of (16), (17), (18), (19) until a maximal number of iterations $MaxIter$ is performed. Defining all parameters is required in order to do it. The visibility is thus defined according to the traveled distance by the nodes as follows:

$$\eta_{(i),j,l} = \frac{1}{\|\boldsymbol{s}_i(t), \boldsymbol{a}_j\| + \|\boldsymbol{s}_{i+1}(t), \boldsymbol{a}_l\|}, \quad (21)$$

where $\eta_{(i),j,l}$ is the ant visibility to set $\boldsymbol{s}_{i+1}(t+1) = \boldsymbol{a}_l$ after setting $\boldsymbol{s}_i(t+1) = \boldsymbol{a}_j$. The visibility thus increases when the traveled distance decreases. While the visibility varies from a state to the other, the initial value of pheromone is chosen constant for all the states and $\Delta\tau$ is set to $\frac{1}{f_{min}}$ where f_{min} is the minimal total distance corresponding to the best solution. α, β, ρ, q_0, τ_0, K_a, and $MaxIter$ are chosen according to several tests. After all iterations are performed, the sensors are assigned the K_m-permutation of positions corresponding to the minimal value of the fitness function over all iterations. The relocation method for a given time-step t is illustrated in Algorithm 2 where V_k is the vector of indices of the positions assigned to the sensors by ant k, $\setminus$ is the set difference operator, $\text{random}(A)$ yields a uniformly distributed random number of the set A and $\text{random}_{P^*}(A)$ yields a random number of A computed according to the specific distribution P^*.

Algorithm 2. Relocation algorithm
Input: Sensors coordinates $\boldsymbol{s}_1(t), \ldots, \boldsymbol{s}_{K_m}(t)$ and available positions $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{K_m}$;
Output: Sensors new coordinates $\boldsymbol{s}_1(t+1), \ldots, \boldsymbol{s}_{K_m}(t+1)$;
Initialization:
for $1 \leq i \leq K_m - 1$ **do**
  for $1 \leq j \leq K_m$ **do**
    for $1 \leq l \leq K_m$ **do**
      $\eta_{(i),j,l} = \frac{1}{\|\boldsymbol{s}_i(t), \boldsymbol{a}_j\| + \|\boldsymbol{s}_{i+1}(t), \boldsymbol{a}_l\|}$;
      $\tau_{(i),j,l} = \tau_0$;
    end
  end
end

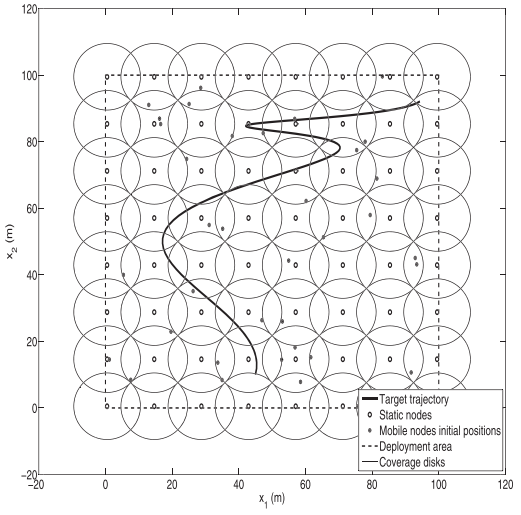

Fig. 6. An illustration of the target trajectory with uniformly deployed static nodes and initial positions of mobile sensors.

while $iter \leq MaxIter$ **do**
  $q = \text{random}([0,1])$;
  for $1 \leq k \leq K_a$ **do**
    $V_k(1) = \text{random}(\{1,\ldots,K_m\})$;
    $J_k(1) = \{1,\ldots,K_m\} \setminus V_k(1)$;
    for $1 \leq i \leq K_m - 1$ **do**
      if $q \leq q_0$ **then**
        $V_k(i+1) = \text{argmax}_{u \in J_k(i)}\{\tau^{\alpha}_{(i),V_k(i),u}.\eta^{\beta}_{(i),V_k(i),u}\}$;
        $J_k(i+1) = J_k(i) \setminus V_k(i+1)$;
        $\tau_{(i),V_k(i),V_k(i+1)} = (1-\rho).\tau_{(i),V_k(i),V_k(i+1)} + \rho.\tau_0$;
      end
      if $q > q_0$ **then**
        for $u \in J_k(i)$ **do**
          $P^*(u) = \frac{\tau^{\alpha}_{(i),V_k(i),u}.\eta^{\beta}_{(i),V_k(i),u}}{\sum_{v \in J_k(i)} \tau^{\alpha}_{(i),V_k(i),v}.\eta^{\beta}_{(i),V_k(i),v}}$;
        end
        for $u \notin J_k(i)$ **do**
          $P^*(u) = 0$;
        $V_k(i+1) = \text{random}_{P^*}(\{1,\ldots,K_m\})$;
        $J_k(i+1) = J_k(i) \setminus V_k(i+1)$;
        $\tau_{(i),V_k(i),V_k(i+1)} = (1-\rho).\tau_{(i),V_k(i),V_k(i+1)} + \rho.\tau_0$;
      end
    end
  end
  for $1 \leq k \leq K_a$ **do**
    $Ob(k) = \sum_{i=1}^{K_m} \|\boldsymbol{s}_i(t), \boldsymbol{a}_{V_k(i)}\|$;
  end
  $k^* = \text{argmin}_{1 \leq k \leq K_a} Ob$;
  $C_{iter} = V_{k^*}$, $MinOb(iter) = Ob(k^*)$;
  $\Delta\tau = \frac{1}{Ob(k^*)}$;
  for $1 \leq i \leq K_m - 1$ **do**
    $\tau_{(i),V_{k^*}(i),V_{k^*}(i+1)} = (1-\rho).\tau_{(i),V_{k^*}(i),V_{k^*}(i+1)} + \rho.\Delta\tau$;
  end
end
$iter^* = \text{argmin}_{1 \leq iter \leq MaxIter} MinOb$;
for $1 \leq i \leq K_m$ **do**
  $\boldsymbol{s}_i(t+1) = C_{iter^*}(i)$;
end

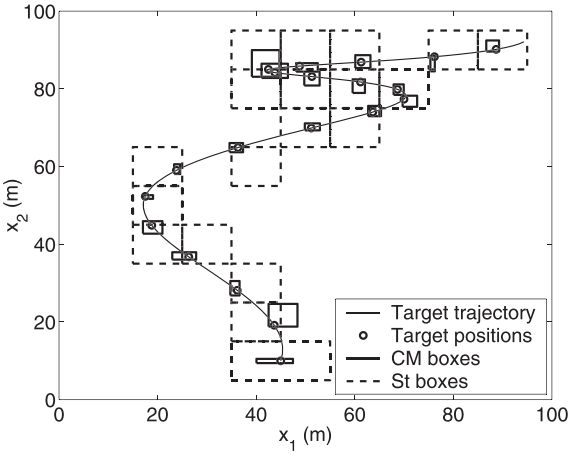

Fig. 7. An illustration of the estimated boxes obtained with both our method (CM) and the static method.

5 Simulations

In order to evaluate the effectiveness of the proposed method, we suppose a target moving in a $[0, 100\text{ m}] \times [0, 100\text{ m}]$ deployment area. The sensing range of sensors is set to 10 m. The number of required static nodes is thus equal to $K_1.K_2$ where $K_1 = K_2 = \text{int}u\{100/10.\sqrt{2}\} = 8$. A 100-steps target trajectory is illustrated in Fig. 6. It shows static nodes as well. It is obvious that every single point of the area is covered by at least one static node. Note that static nodes are not required to have the same sensing range as for mobile nodes. The plot shows as well the initial positions of the mobile sensors. In the following, we first compare the proposed method to an interval-based method developed for static sensor networks. We then compare the guaranteed relocation-based approach of our method to the accuracy-based one. We evaluate afterwards the sensitivity of the proposed method to the sensing range of the mobile sensors. We then compare the estimation technique based on intervals to a Monte-Carlo-based technique. We evaluate afterwards the second-order prediction model. We finally illustrate the effectiveness of the ant colony optimization algorithm. Note that all simulations are performed on an Intel(R) Core(TM)2 CPU (2.40 GHz, 1.00 GB RAM) using MATLAB 7.9.

5.1 Comparison of the Proposed Method to an Interval-Based Method for Static Sensor Networks

In this section, we compare our method to a target tracking method developed for static sensor networks. We thus propose an interval-based method performing a similar estimation as our method. The sensors are deployed uniformly for the static method whereas for our method, the mobile nodes are initially deployed in a random manner. Hundred sensors are used for both methods. In particular, 64 static sensors and 36 mobile ones are considered for our method. Note that all mobile nodes are used at each time step. This leads to $36! = 3.7199.10^{41}$ possible solutions. Fig. 7 shows the estimated boxes obtained with both methods. We are using large sensor triangles for sensors relocation is this example with $\sigma_S = 10.\sqrt{2}$ m, $\tau_0 = 10$, $\alpha = 0.7$, $\beta = 0.3$, $\rho = 0.6$, $q_0 = 0.75$,

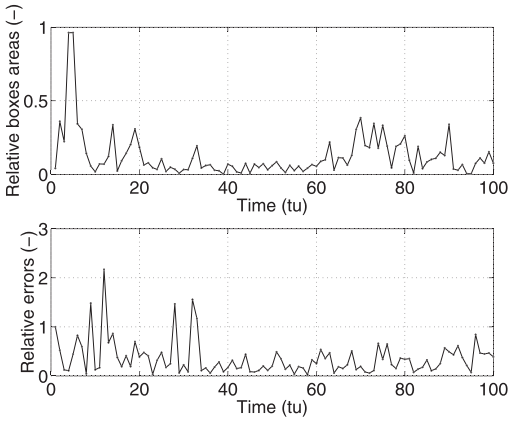

Fig. 8. An illustration of the relative boxes areas (in the top plot) and the relative estimation errors (in the bottom plot).

and $K_a = 10$. Using these values of parameters, the optimal solution is obtained with at most 10 iterations. The average computational time is equal to 0.3125 s per time step for our method. This time is mostly required for the optimization phase since the static method takes around 0.000645 s per time step. The average traveled distance is equal to 7.4539 m per time step for a mobile sensor, whereas it is equal to 2.9806 m for the target. This difference is related to the accuracy of the prediction model. In other words, when the predicted box is large, more mobile sensors, flying larger distances, are needed to ensure accurate estimation of target positions.

Let tu be the time unit and assume that the estimation error at a given time step is equal to the distance between the real position of the target and the center of the estimated box. Then, Fig. 8 shows the ratios of boxes areas and estimation errors obtained with our method (CM) over those obtained with the static method (St). The average ratio of boxes areas is equal to 0.1249 whereas the average ratio of estimation errors is equal to 0.3598. It is thus obvious that moving nodes improves significantly the target tracking.

Fig. 9 shows the variation of the computation time per time step with respect to the total number of sensors. It also shows the variation of the relative boxes area and the relative estimation error. We thus vary the total number of sensors

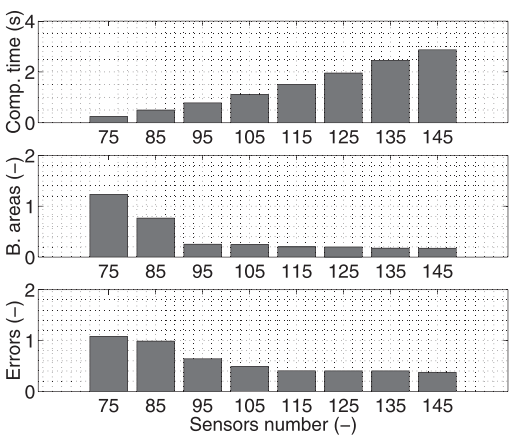

Fig. 9. An illustration of the variation of the computation time, the relative boxes areas, and the relative estimation errors with respect to the total number of sensors.

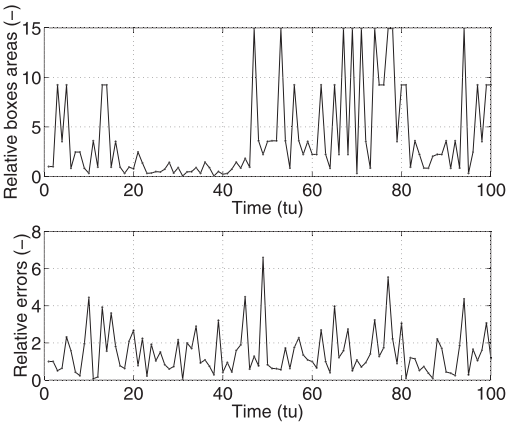

Fig. 10. An illustration of the ratios curves of the boxes areas (in the top plot) and the estimation errors (in the bottom plot) obtained with ACM over GCM.

from 75 to 145. Note that 64 of all sensors are static for our method. Compared to the static method, the performance of our method increases with the increase of the number of mobile nodes at the cost of the computational time.

5.2 Comparison of the Guarantee-Based Approach to the Accuracy-Based One for Sensors Relocation

In this section, we show the impact of the distances between the sensors on the accuracy of estimation. For this reason, we consider 80 sensors, 64 of them being static. We thus compare our method with $\sigma_S = 10$ m (GCM) to a second version of it with $\sigma_S = 10.\sqrt{2}$ m (ACM). Fig. 10 shows the ratios curves of the boxes areas and the estimation errors obtained with ACM over those obtained with GCM. Note that we use the same number of nodes K_s for both methods. K_s is defined at each time step as the minimal number of nodes needed to cover the predicted box in a guaranteed way. The plot shows that ACM boxes are smaller than GCM boxes at some time steps and vice versa at others. Figs. 11 and 12 illustrate the relocated mobile sensors corresponding to $t = 31tu$ obtained with ACM and GCM, respectively. The

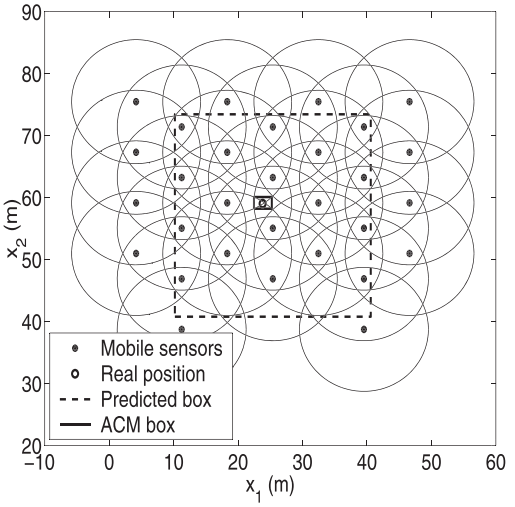

Fig. 11. An illustration of the relocated sensors obtained with ACM at time $t = 31$ tu.

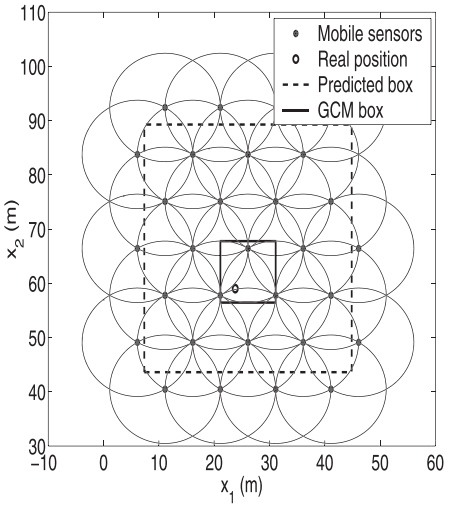

Fig. 12. An illustration of the relocated sensors obtained with GCM at time $t = 31$ tu.

predicted and estimated boxes are also given. The plot shows that the real position falls within a target triangle covered by at least three sensors using ACM, whereas it falls within a larger target triangle with GCM. This leads to a smaller box with ACM yielding more accuracy than GCM. In a different manner, Figs. 13 and 14 illustrate the relocated sensors corresponding to $t = 53tu$ obtained with ACM and GCM, respectively. In this example, the exact position falls outside the accurate target triangles with ACM leading to a larger estimation box. Note that only mobile nodes are involved in the estimation phase in this paragraph in order to illustrate in a better way the impact of sensors triangles size. Adding static sensors may lead to smaller estimation boxes than the one obtained with only mobile nodes. The average ratios of areas and errors with ACM over GCM are equal to 3.5765 and 1.5132, respectively. It is thus obvious that GCM works better than ACM for this example. In fact, with few mobile sensors, the guaranteed method with $\sigma_S = r$ (GCM) covers in a better way the prediction box since with ACM, the area covered by less than three sensors is too large. Having a large number of mobile sensors, one is able to use the ACM version with larger sensor triangles. The method consists thus of covering the whole area with small

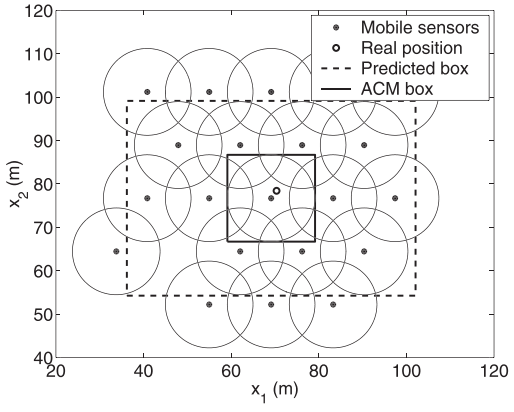

Fig. 13. An illustration of the relocated sensors obtained with ACM at time $t = 53$ tu.

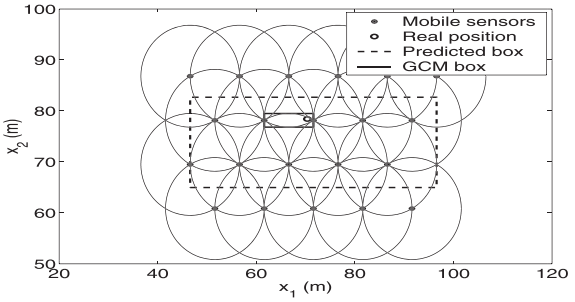

Fig. 14. An illustration of the relocated sensors obtained with GCM at time $t = 53$ tu.

target triangles which refines the mesh leading to more accuracy in the estimation.

5.3 Sensitivity of the Proposed Method to the Sensing Range Value

In this section, we illustrate the dependence of the performances of our method to the sensing range of mobile nodes. For this reason, we assume that fixed sensors are having a fixed sensing range equal to 10 m and that mobile sensors are having a sensing range varying from 1 to 15 m. Given the same surveillance area of Fig. 6 with a network composed of 100 sensors, 64 fixed sensors are needed to ensure the continuous coverage, and thus 36 mobile sensors remain for following the target. The structures used in the definition of sensor locations are guaranteed, and thus the lengths of the triangle sides are equal to the sensing range of mobile nodes. Note that the sensors that are able to move at each time step are located at a distance less than 30 m to the borders of the predicted box. Fig. 15 shows from top to bottom: the average traveled distance per time step and per mobile node (Movement in m), the average estimation error per time step (Errors in m), the average area of estimated boxes (E.B. areas in m^2) and the average area of predicted boxes (P.B. areas in m^2) as functions of the sensing range. It is obvious that the proposed method is more accurate with smaller sensing ranges. The average computing time and the average number of mobile sensors moved per time step vary

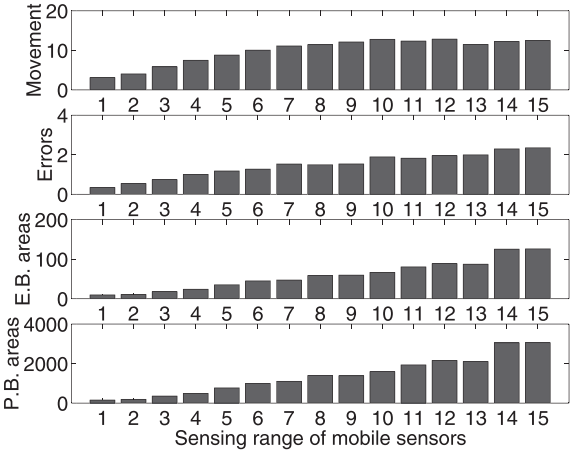

Fig. 15. An illustration of the average traveled distance (Movement in m), the average estimation error (Errors in m), the average area of estimated boxes (E.B. areas in m^2) and the average area of predicted boxes (P.B. areas in m^2) as functions of the sensing range.

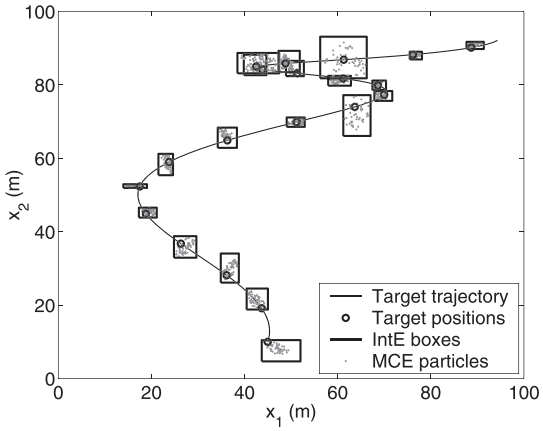

Fig. 16. An illustration of the estimated boxes with IntE and the estimated particles with MCE.

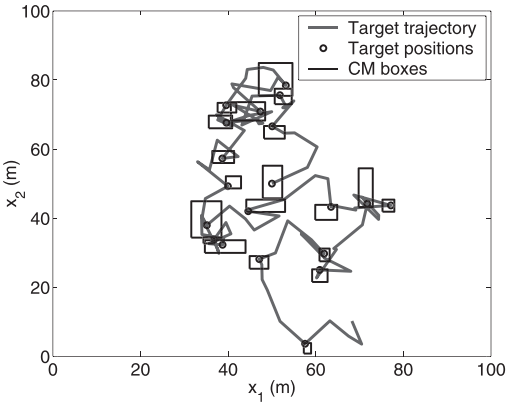

Fig. 17. An illustration of a nonpredictable target trajectory with the estimated boxes.

a little with different sensing range values. Their values are around $0.3850\,\mathrm{s}$ and 30 nodes, respectively. Note that, as expected, with larger sensing ranges, larger estimated boxes are obtained, leading to less accuracy and thus more estimation errors. Moreover, having more incertitude on the estimated positions leads to larger predicted boxes, forcing mobile sensors to travel larger distances. For these reasons, one can say that the proposed method performs better with mobile sensors having small sensing ranges.

5.4 Comparison of the Interval-Based Estimation Technique (IntE) to a Monte-Carlo-Based Method

In this section, we illustrate the performances of the Interval-based Estimation technique. We thus compare it to a Monte-Carlo-based Estimation method (MCE) [30], inspired by the works of Hu and Evans in [31] and Baggio and Langendoen in [32]. In other words, MCE is an iterative method, aiming at generating a fixed number of particles N_p, in the way to estimate the posterior distribution of the unknown position. Practically, the MCE method iterates two phases: the generation phase and the correction phase. The first phase consists of generating N_p random points within a specific rectangular area, called initial domain; whereas the second phase consists of filtering these points, by only keeping those who satisfy all observation constraints. The initial domain could be the whole surveillance area. In order to reduce the computation time of MCE, we propose to use at each time step a smaller domain, defined by the overlapping region of the squares covering the sensing disks of sensors detecting the target. Both phases are iterated until N_p particles are kept in the memory. In order to compare IntE to MCE, we first run the whole controlled mobility method over 100 time steps, with $\sigma_S = r = 10\,\mathrm{m}$. In this way, we can obtain the path of 100 positions that follows each mobile node. Then, MCE and IntE are run using the same computed paths of mobile nodes. Fig. 16 shows the boxes obtained with IntE and the particles obtained with MCE, with N_p set to 50. The computing times are equal to $0.003922\,\mathrm{s}$ and $0.016657\,\mathrm{s}$ with IntE and MCE, respectively. Let the estimation error be the average distance between the centers of the boxes and the exact positions for IntE and the average distance between the barycenters of the particles and the exact positions for MCE. Then, the estimation errors are equal to $1.4602\,\mathrm{m}$ for IntE and $1.8735\,\mathrm{m}$ for MSE. Moreover, it is worth noting that with the MSE method, N_p particles are kept in the memory, whereas only one box is computed in IntE at each time step. In consequence, the interval-based method is more efficient than the Monte-Carlo-based method in terms of computing time, memory consumption, and accuracy.

5.5 Evaluation of the Prediction Model

In this section, we study the sensitivity of the method to the prediction model performances. Being a second-order model, the prediction model combines three consecutive estimated positions in order to define a prediction box. Compared to the estimated boxes, the prediction box is large since it accumulates all incertitude given by the previous estimates. The proposed model assumes that the acceleration of the target is constant. When the target has abrupt changes in direction, the prediction box may not cover the real position. In order to illustrate such a case, we suppose a target moving using a random walk mobility model [33]. The target is thus having an nonpredictable movement having at each time period a random velocity varying between 0 and $10\,\mathrm{m.tu}^{-1}$ and a random direction varying between 0 and $2.\pi$. The target trajectory and the estimated boxes are shown in Fig. 17. While the prediction model generates wrong predicted boxes at 35 time steps, estimation remains mostly accurate. This is mainly due to the static nodes covering the whole deployment area. Fig. 18 shows in black the predicted box, the estimated box, and the real position of the target given at time step $t = 18\,\mathrm{tu}$. It also shows in gray the predicted box, the estimated box and the real position corresponding to time $t = 19\,\mathrm{tu}$. It thus demonstrates that even if the prediction model is not working at a time step, it could perform correct prediction at the following step.

5.6 Effectiveness of the Ant Colony Optimization

In this section, we illustrate the effectiveness of the ant colony optimization algorithm. For this reason, we compare it to the exact method where all possible permutations are evaluated. We thus consider a network composed of 64 static sensors and 10 mobile nodes. All mobile nodes are

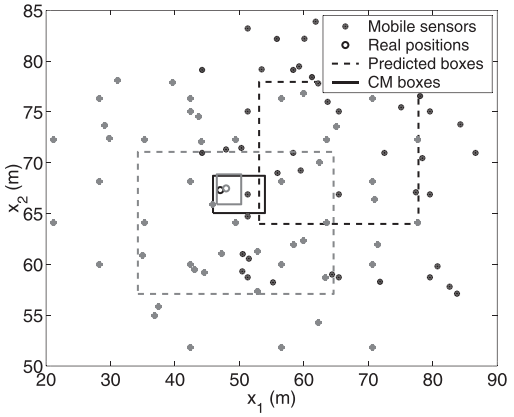

Fig. 18. An illustration of the predicted boxes, the estimated boxes and the real positions of the target given at time $t = 18\,\mathrm{tu}$ (in black) and $t = 19\,\mathrm{tu}$ (in gray).

relocated at each time step. The total number of possible solutions is thus equal to $10! = 3{,}628{,}800$. Having a set of 10 positions, the exact method consists of generating all 10-permutations of solutions where each mobile sensor is assigned one position of the set. It then computes the total traveled distance of the nodes. The permutation yielding the minimal movement is finally chosen. We consider a target moving over 100 time steps. At each time step, new positions of sensors are defined and then both ACO algorithm and the exact method are applied. Fig. 19 shows the target trajectory and the estimated boxes obtained with 10 relocated mobile nodes. The average time required by the ACO-based approach to perform one time-step computation is equal to $0.1821\,\mathrm{s}$. Fig. 20 shows the relative decrease of the computational time obtained with ACO with respect to the computational time obtained with the exact method in the top plot. This plot shows that the computation time is increased for about 98 percent at all time steps. The bottom plot shows the relative increase of the traveled distance obtained with ACO with respect to the exact method. The average decrease of computation time is equal to 98.62 percent whereas the average increase of the traveled distance is given by 3.39 percent. Note that the computation time of the exact method highly increases with the increase of the number of mobile nodes. While the energy consumption is slightly increased, the ACO ensures

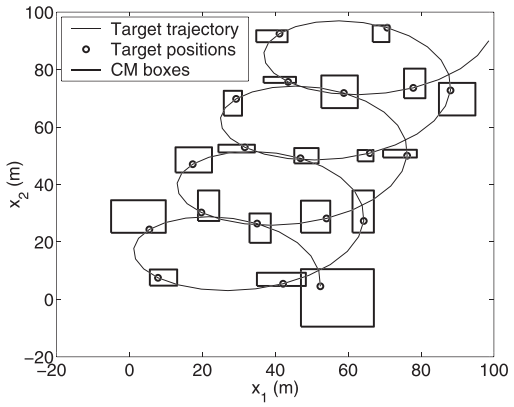

Fig. 19. An illustration of the target trajectory and the estimated boxes obtained with either the ACO-based approach or the exact method-based one.

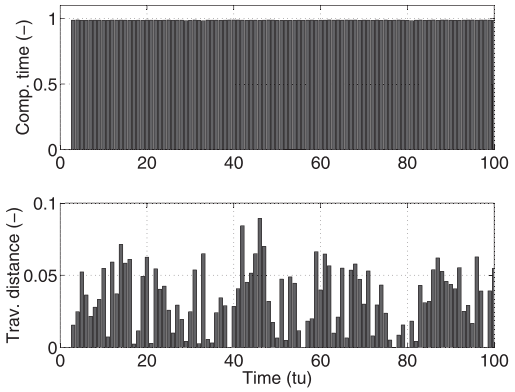

Fig. 20. An illustration of the relative decrease of the computation time with ACO with respect to the exact method (in the top plot) and the relative increase of the traveled distance with ACO with respect to the exact method (in the bottom plot).

a substantial gain in terms of computation time. It is thus crucial to use the ACO algorithm especially when the number of mobile nodes increases.

6 Conclusion

In this paper, we proposed an original method for target tracking in controlled mobility sensor networks. Having a moving target at each time step, the method consists of estimating the current position of the target and then predicting its following position using a second-order prediction model. A relocation of sensors is then performed in order to optimize the target localization for the following time step. A set of positions is thus defined using a triangulation-based method. Each sensor is then assigned one position of the set using an ant colony optimization algorithm. While the relocation phase uses a metaheuristic-based approach, estimation and prediction phases employ interval analysis where target positions are boxes including the real value. The proposed approach uses a hybrid sensor network composed of both static and mobile nodes. While mobile nodes are used for optimizing the target tracking, static nodes ensure the total coverage of the network. Simulation results illustrate the efficiency of the proposed method compared to algorithms developed for static sensor networks. Future works will handle the problem in a distributed manner where decisions are locally made. One is also able to extend the method to a multitarget tracking problem.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, pp. 393-422, 2002.

[2] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G.S. Sukhatme, "Robomote: Enabling Mobility in Sensor Networks," *Proc. IEEE/ACM Fourth Int'l Conf. Information Processing in Sensor Networks (IPSN-SPOTS)*, pp. 404-409, 2005.

[3] D. Zeinalipour-Yazti and P.K. Chrysanthis, "Mobile Sensor Network Data Management," *Encyclopedia of Database Systems*, pp. 1755-1759, Springer, 2009.

[4] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, and D. Rubenstein, "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with Zebranet," *Proc. 10th Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLPS '02)*, 2002.

[5] *Mobile, Wireless, and Sensor Networks: Technology, Applications, and Future Directions,* R. Shorey, ed. John Wiley & Sons, 2006.
[6] G. Song, Y. Zhou, F. Ding, and A. Song, "A Mobile Sensor Network System for Monitoring of Unfriendly Environments," *Sensors,* vol. 8, pp. 7259-7274, Nov. 2008.
[7] S. Aeron, V. Saligrama, and D.A. Castañón, "Efficient Sensor Management Policies for Distributed Target Tracking in Multihop Sensor Networks," *IEEE Trans. Signal Processing,* vol. 56, no. 6, pp. 2562-2574, June 2008.
[8] P.M. Djurić, M. Vemula, and M.F. Bugallo, "Target Tracking by Particle Filtering in Binary Sensor Networks," *IEEE Trans. Signal Processing,* vol. 56, no. 6, pp. 2229-2238, June 2008.
[9] J. Teng, H. Snoussi, and C. Richard, "Decentralized Variational Filtering for Target Tracking in Binary Sensor Networks," *IEEE Trans. Mobile Computing,* vol. 9, no. 10, pp. 1465-1477, Oct. 2010.
[10] A. Kansal, M. Rahimi, W.J. Kaiser, M.B. Srivastava, G.J. Pottie, and D. Estrin, "Controlled Mobility for Sustainable Wireless Networks," *Proc. IEEE CS First Ann. Conf. Sensor and Ad Hoc Comm. and Networks (SECON),* pp. 1-6, Oct. 2004.
[11] S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli, and Z.M. Wang, "Controlled Sink Mobility for Prolonging Wireless Sensor Networks Lifetime," *Wireless Networks,* vol. 14, no. 6, pp. 831-858, 2008.
[12] T.P. Lambrou, C.G. Panayiotou, S. Felici, and B. Beferull, "Exploiting Mobility for Efficient Coverage in Sparse Wireless Sensor Networks," *Wireless Personal Comm.,* vol. 54, no. 1, pp. 187-201, Apr. 2009.
[13] Y. Zou and K. Chakrabarty, "Distributed Mobility Management for Target Tracking in Mobile Sensor Networks," *IEEE Trans. Mobile Computing,* vol. 6, no. 8, pp. 872-887, Aug. 2007.
[14] J.-C. Chin, Y. Dong, W.-K. Hon, C.Y.T. Ma, and D.K.Y. Yau, "Detection of Intelligent Mobile Target in a Mobile Sensor Network," *IEEE/ACM Trans. Networking,* vol. 18, no. 1, pp. 41-52, Feb. 2010.
[15] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis.* Springer, 2001.
[16] M. Dorigo and L. Gambardella, "Ant Colonies for the Traveling Salesman Problem," *Biosystems,* vol. 43, pp. 73-81, 1997.
[17] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard, *Métaheuristiques pour lóptimisation difficile.* Eyrolles, 2005.
[18] F. Mourad, H. Snoussi, F. Abdallah, and C. Richard, "Guaranteed Boxed Localization in MANETs by Interval Analysis and Constraints Propagation Techniques," *Proc. IEEE GlobeCom,* 2008.
[19] F. Mourad, H. Snoussi, F. Abdallah, and C. Richard, "Anchor-Based Localization via Interval Analysis for Mobile Ad-Hoc Sensor Networks," *IEEE Trans. Signal Processing,* vol. 57, no. 8, pp. 3226-3239, Aug. 2009.
[20] A. Medeisis and A. Kajackas, "On the Use of the Universal Okumura-Hata Propagation Prediction Model in Rural Areas," *Proc. Vehicular Technology Conf.,* vol. 3, 2000.
[21] Z. Nadir, N. Elfadhil, and F. Touati, "Pathloss Determination Using Okumura-Hata Model and Spline Interpolation for Missing Data for Oman," *Proc. World Congress Eng.,* vol. 1, July 2008.
[22] R.E. Moore, *Methods and Applications of Interval Analysis.* SIAM, 1979.
[23] D.L. Waltz, "Generating Semantic Descriptions from Drawings of Scenes with Shadows," technical report, Massachusetts Inst. of Technology, 1972.
[24] M. Goldberg, "The Packing of Equal Circles in a Square," *Math. Magazine,* vol. 43, no. 1, pp. 24-30, http://www.jstor.org/stable/2688107, 1970.
[25] M. Locatelli and U. Raber, "Packing Equal Circles in a Square: A Deterministic Global Optimization Approach," *Discrete Applied Math.,* vol. 122, nos. 1-3, pp. 139-166, 2002.
[26] M.L.B. Addis and F. Schoen, "Disk Packing in a Square: A New Global Optimization Approach," *INFORMS J. Computing,* vol. 20, no. 4, pp. 516-524, 2008.
[27] Y. Hani, L. Amodeo, F. Yalaoui, and H. Chen, "Ant Colony Optimization for Solving an Industrial Layout Problem," *European J. Operational Research,* vol. 183, pp. 633-642, 2007.
[28] A. Donati, R. Montemanni, N. Casagrande, A. Rizzoli, and L. Gambardella, "Time Dependent Vehicle Routing Problem with a Multi Ant Colony System," *European J. Operational Research,* vol. 185, pp. 1174-1191, 2008.
[29] H. Chehade, L. Amodeo, and F. Yalaoui, "A New Efficient Hybrid Method for Selecting Machines and Sizing Buffers in Assembly Lines," *J. Operations and Logistics,* vol. 3, pp. 1-22, 2009.
[30] A. Doucet, S. Godsill, and C. Andrieu, "On Sequential Montecarlo Sampling Methods for Bayesian Filtering," *Statistics and Computing,* vol. 10, pp. 197-208, 2000.
[31] L. Hu and D. Evans, "Localization for Mobile Sensor Networks," *Proc. ACM MobiCom,* 2004.
[32] A. Baggio and K. Langendoen, "Monte-Carlo Localization for Mobile Wireless Sensor Networks," *Proc. Second Int'l Conf. Mobile Ad Hoc and Sensor Networks (MSN '06),* 2006.
[33] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Comm. and Mobile Computing,* Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, pp. 483-502, 2002.

Farah Mourad received the diploma degree in electrical engineering from the Lebanese University, Faculty of Engineering, Tripoli, in 2006, the master's degree in systems optimization and safety from the University of Technology of Troyes (UTT), France, in 2007, and the PhD degree in systems optimization and safety from UTT in 2010. Her research concerns the localization of mobile sensors and the tracking of targets in mobile sensor networks.

Hicham Chehade received the engineering degree in industrial systems engineering and the master's degree in systems optimization and safety in 2005, followed by the PhD degree in systems optimization and safety from the University of Technology of Troyes (UTT), France, in 2009. He is currently an assistant professor in the Industrial Systems Optimization Laboratory (ICD-LOSI, STMR UMR CNRS 6279) at UTT. His research fields concern mainly the optimization of production, assembly, and robotic lines design. He also has other research topics such as scheduling problems, operations research, modeling, and simulation.

Hichem Snoussi received the diploma degree in electrical engineering from the Ecole Supérieure d'Electricité (Supelec), Gif-sur-Yvette, France, in 2000 and the DEA and PhD degrees in signal processing from the University of Paris-Sud, Orsay, France, in 2000 and 2003, respectively. He received the HdR from the University of Technology of Compiègne in 2009. Between 2003 and 2004, he was a postdoctoral researcher at IRCCyN, Institut de Recherches en Communications et Cybernétiques de Nantes. He has spent short periods as a visiting scientist at the Brain Science Institute, RIKEN, Japan, and the Olin Neuropsychiatry Research Center at the Institute of Living, United States. Between 2005 and 2010, he has been an associate professor at the University of Technology of Troyes (UTT). Since September 2010, he has been appointed as a full professor position at UTT. He is in charge of the regional research program S3 (System Security and Safety) of CPER 2007-2013 and the CapSec platform (wireless embedded sensors for security). He is the principal investigator of an ANR-Blanc project (mv-EMD), a CRCA project (new partnership and new technologies), and a GDR-ISIS young researcher project. He is a partner of many ANR projects, GIS, and strategic UTT programs. He obtained the national doctoral and research supervising award PEDR 2008-2012. He is a member of the IEEE.

Farouk Yalaoui received the engineering degree in industrial engineering from the Polytechnics School of Algiers in 1995, the master's degree in industrial system engineering from the Polytechnics Institute of Lorraine, Nancy, France, in 1997, the PhD degree in production management from the Troyes University of Technology (UTT) in 2000, and followed by a Habilitation à diriger les recherches (Dr. Hab) from Compiegne University of Technology in 2006. He is currently a full professor at UTT, France, where he is a head of the master's programme and the head of production management and Filed 2 of the OSI-ICD team (FRE CNRS, UTT). His research focuses on scheduling problems, system design, operations research, modeling, analysis and optimization of logistic and production systems, reliability and maintenance optimization, and optimization problems in general. He is the author or coauthor of more than 180 contributions, publications, or communications with one book, seven book chapters, and 38 articles in journals such as *IIE Transactions*, *European Journal of Operational Research*, *International Journal of Production Economics*, *IEEE Transactions on Reliability*, *Reliability Engineering and System Safety*, *Computer & Operations Research*, and *Journal of Intelligent Manufacturing*. He has also published more than 110 papers in conference proceedings and is member of several international conference committees and International journal boards.

Lionel Amodeo received the PhD degree in automatic and production management from the University of Technology of Belfort-Montbeliard, Belfort, France, in 1999. Since 2000, he has been with the Technology University of Troyes, France, where he recently became a full professor in the Industrial Systems Optimization Department. He is currently the head of the engineer degree in industrial systems. His research interests include logistic and production systems optimization, scheduling, and facility layout problems. He has published one book, seven book chapters, and more than 140 papers in technical journals and conference proceedings. He was a finalist of the Kayamori best paper award of the 2002 IEEE International Conference on Robotics and Automation, Washington, May 2002. He has organized and chaired more than 10 sessions/tracks for many conferences including IEEE and has served as a program committee member for several international conferences including those sponsored by the IEEE.

Cédric Richard received the Dipl-Ing and the MS degrees in 1994 and the PhD degree in 1998 from the University of Technology of Compiègne, France, all in electrical and computer engineering. From 1999 to 2003, he was an associate professor at the University of Technology of Troyes (UTT), France. From 2003 to 2009, he was a full professor at the Institut Charles Delaunay (CNRS FRE 2848) at UTT, and the supervisor of a group consisting of 60 researchers and PhD students. In winter 2009 and autumn 2010, he was a visiting researcher with the Department of Electrical Engineering, Federal University of Santa Catarina (UFSC), Florianòpolis, Brazil. He has been a junior member of the Institut Universitaire de France since October 2010. Since September 2009, he has been a full professor at Fizeau Laboratory (CNRS UMR 6525, Observatoire de la Côte d'Azur), University of Nice Sophia-Antipolis, France. His current research interests include statistical signal processing and machine learning. He is the author of more than 100 papers. He was the general chair of the XXIth francophone conference GRETSI on Signal and Image Processing that was held in Troyes, France, in 2007, and of the IEEE International Workshop on Statistical Signal Processing held in Nice in 2011. Since 2005, he has been a member of the board of the federative CNRS research group ISIS on Information, Signal, Images, and Vision. He has served as an associate editor of the *IEEE Transactions on Signal Processing* since 2006 and of the *EURASIP Signal Processing Magazine* since 2009. In 2009, he was nominated as liaison local officer for EURASIP and member of the Signal Processing Theory and Methods (SPTM) Technical Committee of the IEEE Signal Processing Society. Paul Honeine and he received the Best Paper Award for "solving the pre-image problem in kernel machines: a direct method" at the 2009 IEEE Workshop on Machine Learning for Signal Processing (IEEE MLSP 2009). He is a member of the GRETSI association board and the EURASIP Society, and a senior member of the IEEE.