Improving Area Coverage of Wireless Sensor Networks Via Controllable Mobile Nodes: a Greedy Approach☆

Massimo Vecchio[a,*], Roberto López-Valcarce[b]

[a] *CREATE-NET, Via alla Cascata 56/D, 38123 Trento - ITALY*
[b] *Departamento de Teoría de la Señal y las Comunicaciones, Universidad de Vigo, 36310 Vigo - SPAIN*

Abstract

Reliable wide–area monitoring with Wireless Sensor Networks (WSNs) remains a problem of interest: simply deploying more nodes to cover wider areas is generally not a viable solution, due to deployment and maintenance costs and the increase in radio interference. One possible solution gaining popularity is based on the use of a reduced number of mobile nodes with controllable trajectories in the monitored field. In this framework, we present a distributed technique for iteratively computing the trajectories of the mobile nodes in a greedy fashion. The static sensor nodes actively assist the mobile nodes in this task by means of a bidding protocol, thus participating towards the goal of maximizing the area coverage of the monitored field. The performance of the proposed technique is evaluated on various simulation scenarios with different number of mobile and static nodes in terms of achieved coverage and mean time to achieve $X\%$ coverage. Comparison with previous state-of-the-art techniques reveals the effectiveness and stability of the proposed method.

Keywords: Wireless sensor networks, collaborative search, mobile collectors.

☆Research supported by the European Regional Development Fund (ERDF) and the Spanish Government (TEC2010-21245-C02-02/TCM DYNACS, CONSOLIDER-INGENIO 2010 CSD2008-00010 COMONSENS), and the Galician Regional Government (CN 2012/260 AtlantTIC).

*Tel: (+39) 0461 408400 Fax: (+39) 0461 421157
Email addresses: massimo.vecchio@create-net.org (Massimo Vecchio), valcarce@gts.uvigo.es. (Roberto López-Valcarce)

1. Introduction

Wireless Sensor Networks (WSNs) are composed by a set of small autonomous systems *(sensor nodes)* which collaborate to jointly perform tasks such as *e.g.*, event, intruder and alarm detection, target classification, field surveillance and patrolling. From a functional point of view, each node is a small device able to collect information from the surrounding environment through one or more sensors, to process this information locally and to communicate it to a data collection center called *sink* or *base station*, using generally node to node multi–hop data propagation [1, 2]. Nodes are typically equipped with a *processing unit* with limited memory and computational power, a *sensing unit* for data acquisition from the surrounding environment and a *communication unit*, usually a radio transceiver [3]. Thus, the most basic concept of WSN is well resumed by a simple equation: CPU + Sensing + Radio = Thousands of potential applications [4].

During the last decade several such potential applications sprang out of this equation, including the possibility of large–scale node deployments for monitoring wide physical areas [5]. However, despite the promising potential, the direct approach of simply spreading hundreds of nodes so as to cover larger areas turns out to be impractical. The first reason is merely economic, since a larger number of nodes results in steeper deployment and maintenance costs. The second reason is more practical: WSNs monitoring large areas are more likely to present *coverage holes* (insufficiently monitored areas where events will go undetected) [6]. Although these holes can be reduced by careful node deployment [7], this also incurs in additional costs and a larger node density, with the associated radio access issues including frequent channel contentions, message collisions, and losses [8]. To overcome these limitations, a trend gaining recent popularity consists of enriching the original description of the classical WSN with a new term, namely the *actuator unit*, enabling the movement of sensor nodes. By introducing some degree of mobility, a WSN can better react to the effects of inaccurate deployments and node failures. Thus, mobility provides fault tolerance, enhances sensing and connectivity coverage, reliability, and energy efficiency, while reducing the cost of a dense/smart deployment. Since the actuator unit comes at an additional cost, the number of mobile units is generally kept low with respect to the total number of nodes comprising the so–called *mixed WSNs*.

As reviewed in [9, 10], within the mixed WSN framework, different types of nodes (*e.g.*, generic nodes, sink nodes or both) can be endowed with mobil-

ity; for instance, in two opposite approaches generic nodes are mobile whereas sink nodes are static, or vice versa. In addition, different degrees of mobility are possible; for example, the review in [10] considers three types of mobile nodes with increasing mobility: *relocatable nodes*, *mobile data collectors* (further cataloged into *mobile sinks* and *mobile relays*), and *mobile peers*. Relocatable nodes only move to change the given topology of the network, assumed yet to be rather dense, generally for improving connectivity and/or coverage [11, 12]. On the other hand, mobile collectors are responsible for visiting the network to collect data generated from source nodes, acting either as destination nodes in the case of mobile sinks or as intermediate nodes in the case of mobile relays. Finally, unlike mobile data collectors (either sinks or relays), mobile peers can be both data source and data relays, while moving within the scenario: hence, when a peer is in the communication range of a base station, it transfers its own data as well as those gathered from other peers. A more detailed classification can be done on the basis of the different mobility patterns [10, 13]. In particular, mobile nodes can follow *uncontrolled* (either *deterministic* or *random*) mobility patterns, or they can actively change their locations by modifying a *controllable* motion trajectory and/or speed. For a complete overview of the state of the art in mixed WSNs, the interested reader is referred to [10] and the references therein.

The main contribution of this paper is a distributed path–planning and coordination technique for the mobile nodes of a mixed WSN, with the goal of improving sensing area coverage. We consider a set of static, non-relocatable sensor nodes randomly deployed in the scenario, and a small fraction of mobile nodes, which can modify and control their trajectories subject to some maneuverability constraints. Thus, according to the classification above, this framework is composed by a static infrastructure of generic nodes, the mobile nodes constitutes the set of mobile sinks with controllable trajectories, and the goal is to exploit mobility to increase the sensing coverage offered by the static infrastructure, by sampling coverage holes. A similar framework has been considered in [14]; however, we argue that, despite its effectiveness, the technique from [14] does not fully exploit the *distributed* processing power offered by the static nodes of the network. Indeed, in [14] the mobile sinks are the only players acting in the stage, and the static infrastructure only passively contributes to the scene by offering its *distributed* sensing unit. With the aim of leveraging the active participation of the static nodes to the design of the mobile nodes' trajectories, we propose a bidding strategy

whose main idea is as follows. Whenever a mobile node has to navigate towards a coverage hole, it sends an auction message which is received by the static sensor nodes within its communication range. Once advertised of the auction, the static sensor nodes independently estimate the extension and the location of the biggest coverage hole around them and directly bid the mobile node. Upon reception of the bids, the mobile node moves towards the best bid (*i.e.*, the widest coverage hole among the received ones). The use of bidding strategies has also been applied to coverage problems in [15], but in a different scenario consisting of a fraction of relocatable nodes. In contrast, in our framework the generic (static) nodes are not relocatable, whereas one or more mobile sinks are available.

The paper is organized as follows. The system model and assumptions are described in Section 2, and in Section 3 the proposed technique is presented. Due to the several active players in the stage, some deterministic countermeasures have to be designed in order to avoid unwanted behavior of the mobile nodes, as described in Section 4. The proposed approach is tested in Section 5 in several simulation scenarios and compared with previous approaches. Finally, conclusions are drawn in Section 6.

2. Model and assumptions

2.1. System model

We model the sensor field as a rectangle $U \subset \mathbb{R}^2$ with size $X \times Y$. A total of $s+m$ sensor nodes are randomly deployed at locations $\boldsymbol{p}_i = (x_i, y_i) \in U$. It is assumed that each node knows its location (by using a suitable localization technique, see *e.g.*, [16]) and advertises it by periodically sending beaconing messages. For tractability, and similarly to [14], U is discretized into square cells of size d (with $d \ll X$ and $d \ll Y$). Hence, the sensor field U can be viewed as a grid $\mathcal{U}$ of size $N_x \times N_y$, where $N_x = \lceil X/d \rceil$ and $N_y = \lceil Y/d \rceil$. After discretization, sensor node i located at $\boldsymbol{p}_i = (x_i, y_i) \in U$ is in the cell $\boldsymbol{u}_i = (\lceil x_i/d \rceil, \lceil y_i/d \rceil) \in \mathcal{U}$.

We assume isotropic sensing and communication models in which the sensing and communication areas are given by circles with radii r_s and r_c, respectively (with $r_s < r_c$, see Figure 1). Thus, two nodes i and j can communicate with each other if and only if $\|\boldsymbol{p}_i - \boldsymbol{p}_j\| \leq r_c$. We note that the proposed approach, to be presented in Section 3, is independent of the specific sensing and communication models: we adopt for both the isotropic model only for the sake of simplicity.

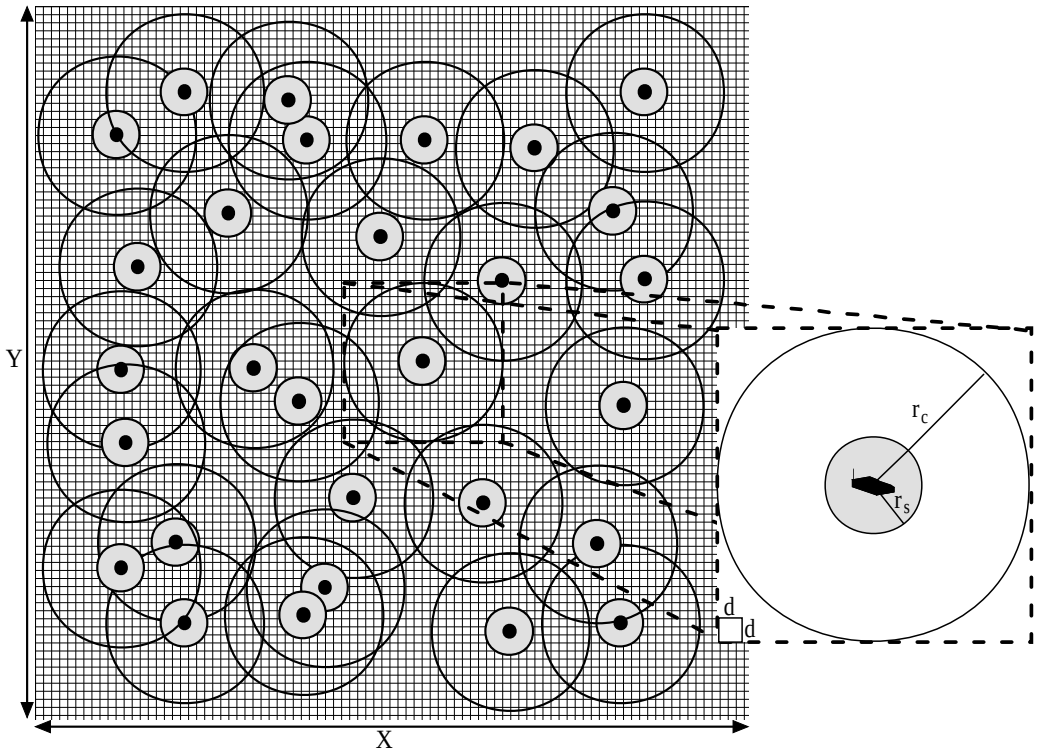


Figure 1: A WSN scenario: $s + m$ nodes are randomly deployed in $U = X \times Y$ and each node is characterized by a sensing and communication radius, r_s and r_c (with $r_s < r_c$), respectively.

The nodes with indices $i = 1, \ldots, s$ constitute the static WSN infrastructure, whereas indices $i = s + 1, \ldots, s + m$ correspond to the m mobile sinks. To be consistent with the motion dynamics of physical vehicles, we assume that each mobile sink can control its trajectory according to some maneuverability constraints, similarly to the model described in [17]. In particular, mobile nodes are able to move at constant speed μ and to turn with a maximum angle ϕ with respect to their current direction. To better describe the path of a mobile sink, let T denote the *movement sampling time*, that is, the time interval between two successive control commands issued to the actuator unit. Based on the maneuverability constraints, assume that at time $t = kT$ a mobile node i is located at $\boldsymbol{p}_i(k) = (x_i(k), y_i(k))$, moving with constant speed μ and angle $\theta(k)$, so that its velocity vector is $(\mu \cos\theta(k), \mu \sin\theta(k))$. Then its location $\boldsymbol{p}_i(k + 1)$ at time $t = (k + 1)T$

will be given by $\boldsymbol{p}_i(k+1) = \boldsymbol{p}_i(k) + \mu T(\cos\theta(k+1), \sin\theta(k+1))$, where $\theta(k+1) \in [\theta(k)-\phi,\, \theta(k)+\phi]$. In order to formulate the path planning problem as an integer programming problem, the continuous arc of candidate positions for $\boldsymbol{p}_i(k+1)$ is discretized into a set of n candidate positions:

$$\mathsf{CP}_i\,(k+1) = \left\{\boldsymbol{p}_i^1\,(k+1)\,, \ldots, \boldsymbol{p}_i^n\,(k+1)\right\}, \tag{1}$$

i.e., the set $\mathsf{CP}_i\,(k+1)$ comprises the cells in $\mathcal{U}$ through which the arc $\{\boldsymbol{p}_i(k) + \mu T(\cos\theta,\, \sin\theta)\,|\,\theta(k) - \phi \le \theta \le \theta(k) + \phi\}$ passes.

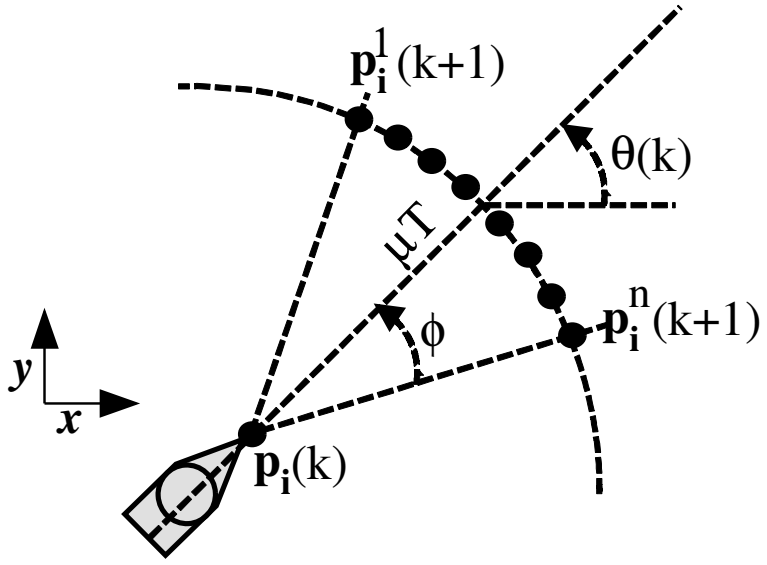


Figure 2: The adopted mobility model: a mobile node moves at constant speed μ and can turn with a maximum angle ϕ with respect to its current direction $\theta(k)$. The continuous arc of candidate positions at time $(k+1)T$ is discretized into a set of n candidate positions.

Depending on its instantaneous location, a mobile node can directly exchange messages with the subset of static nodes within its communication range. Hence, at time kT, the set of neighboring nodes of mobile node i is defined as:

$$\mathsf{N}_i(k) = \{j : \|\boldsymbol{p}_i(k) - \boldsymbol{p}_j\| \le r_c,\ j \le s\}\,. \tag{2}$$

Regarding the sensing area, at time kT a generic node i (either static or mobile) covers a subset $\mathcal{S}_i(k) \in \mathcal{U}$ of cells

$$\mathcal{S}_i(k) = \left\{(p,q) : \left(p - \left\lceil \frac{x_i(k)}{d} \right\rceil\right)^2 + \left(q - \left\lceil \frac{y_i(k)}{d} \right\rceil\right)^2 \le \left\lceil \frac{r_s}{d} \right\rceil^2 \right\}, \tag{3}$$

where p and q are integers with $1 \leq p \leq N_x$ and $1 \leq q \leq N_y$. Note that $\mathcal{S}_i(k)$ depends on k for mobile nodes only. The coverage status of $\mathcal{U}$ at time kT is the union of all individually sensed cells up to that time:

$$\mathcal{S}(k) = \bigcup_{l \leq k} \left\{ \bigcup_{i=1}^{s+m} \mathcal{S}_i(l) \right\}, \tag{4}$$

Clearly, cells within sensing range of one or more static nodes are continuously monitored. On the other hand, cells that are not covered by any static node can only be checked if sensed by a mobile node. Our model implicitly assumes that, once such a cell has been so visited, the corresponding sensing information does not get stale too quickly, in the sense that it can be assumed to remain valid at least until the whole scenario has been covered (at least up to a given percentage) and another coverage cycle starts anew.

2.2. Path planning strategy

The trajectories of mobile nodes are controlled by a number of techniques that will be described in the sequel, and that can be loosely classified as *short–term* and *medium–term* mechanisms. The goal of medium–term actions is to compute the location of target cells in $\mathcal{U}$ towards which mobile nodes should be driven. Once these targets are reached, new ones are computed, and in this way the *long–term* goal of covering $\mathcal{U}$ will be eventually achieved. Medium–term actions will be the object of the next section.

By means of short–term actions, at time kT each mobile node i independently determines its next position $\boldsymbol{p}_i(k+1) \in \mathsf{CP}_i(k+1)$ given its medium–term target cell $\boldsymbol{p}_i^t$ and its current position $\boldsymbol{p}_i(k)$ [14, 17, 18]. To this purpose, an objective function $J : \mathbb{R}^2 \to \mathbb{R}$ is minimized over $\mathsf{CP}_i(k+1)$. We consider the following class of functions:

$$J(\boldsymbol{p}) = \sum_{j=1}^{3} w_j J_j(\boldsymbol{p}), \tag{5}$$

where $\{w_j\}$ are suitable weights; generally speaking, the larger the value of the weight w_j, the larger the influence of the associated cost $J_j(\boldsymbol{p})$ on the overall value of $J(\boldsymbol{p})$.

Without loss of generality, and for the sake of a fair comparison with the technique proposed in [14], we consider three additive cost terms in (5),

namely the distance to target, repulsion from static nodes, and boundary barrier cost functions, as described next. Nevertheless, other terms (as detailed in [19]) could be taken into account to devise more sophisticated objective functions able to meet more specific needs; we invite the interested reader to refer to [19] for a comprehensive description of this framework.

1. *Distance to target.* In order to drive the mobile node toward the target, the first term is taken as the distance to the target, normalized by the communication range:

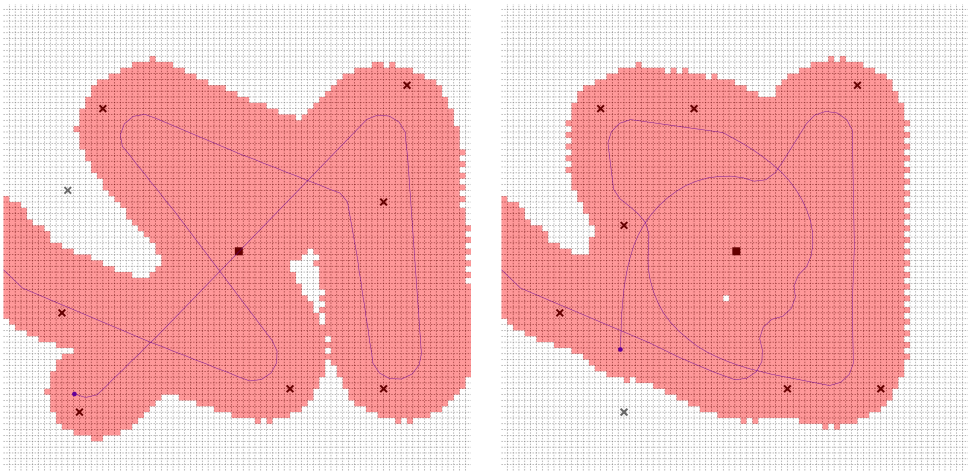
$$J_1(\boldsymbol{p}) = \frac{\|\boldsymbol{p}_i^t - \boldsymbol{p}\|}{r_c}. \tag{6}$$

2. *Repulsion from static nodes.* In order to maximize coverage, mobile nodes should move towards their targets traveling through uncovered zones and avoiding areas already covered by static nodes. The following choice of J_2 accordingly penalizes displacements towards static nodes within communication range at time kT:

$$J_2(\boldsymbol{p}) = \max_{s \in \mathsf{N}_i(k)} \left\{ \exp\left(-\frac{\|\boldsymbol{p}_s - \boldsymbol{p}\|^2}{r_s^2} \right) \right\}, \tag{7}$$

where the distance to static nodes is normalized by the sensing range. In order to better understand the effect of J_2 on the overall value of the objective function $J(\boldsymbol{p})$, Figure 3 considers a simple scenario comprised by one mobile (blue dot) and one static node (black square). The trajectory of the mobile node is marked with a blue line, the successive targets to reach are shown as black crosses (to ease the understanding of the figures, reached targets are not removed from the graph), and covered cells are marked pink. As soon as the mobile node is within communication range of the static node, it starts reaching the targets computed by the latter. As can be noticed, if $w_2 = 0$ (Figure 3(a)) the mobile node passes through the static node's sensing area. On the other hand, when $w_2 > 0$, the corresponding repulsion term prevents this from happening, and as a result, the mobile node steers around the area covered by the static node (note the curves in the mobile trajectory of Figure 3(b)), possibly passing by not-covered-yet areas, hence improving coverage performance. This ultimately motivates the inclusion of a repulsion term J_2 as in (7). Regarding the values of

weights associated to J_1 and J_2 (*i.e.*, w_1 and w_2), after analyzing various scenarios with different numbers of mobile and static nodes, we choose to initialize them with the values suggested in [14]. We must also note that in Section 4 a deterministic countermeasure that temporarily inhibits the effect of the static nodes' repulsion forces will be detailed.



(a) $w_2 = 0$. (b) $w_2 = 0.5$.

Figure 3: A comparison between (a) not including and (b) including a repulsion force in $J(\boldsymbol{p})$.

3. *Boundary barrier.* It makes sense to penalize excursions of the mobile nodes outside the area of interest U. A simple way to do this, and in line with [14], is to introduce a binary function J_3 as follows:

$$J_3(\boldsymbol{p}) = \begin{cases} 0, & \boldsymbol{p} \in U, \\ 1, & \boldsymbol{p} \notin U. \end{cases} \tag{8}$$

The adequate value of the associated weight w_3 of J_3 in (5) ultimately depends on the actual deployment. For example, if zones immediately outside the bounds of the monitored area are steep, or generally harsh (for instance because there is not enough space for the mobile nodes to maneuver), or access to them is forbidden for legal motives, it makes sense to avoid such zones at all by increasing the value of w_3. On the other hand, when out-of-bounds excursions are not a critical issue, they may be allowed to a certain degree in order to ease maneuverability and

avoid larger turnarounds, depending on the mobility constraints shown in Figure 2.

3. Medium–term actions

In this section we present the techniques used for the computation of the target positions towards which the mobile nodes should direct. The three different ingredients will be described in turn: local computation of candidate target positions by static nodes; dynamic interaction between static and mobile nodes based on auctions; and local computation by (and coordination among) mobile nodes.

3.1. Local computation at static nodes

It is clear that candidate target positions should correspond to the approximate location of poorly covered areas. Thus, techniques should be provided that estimate the location as well as the extension of coverage holes in the sensor field. These techniques should be simple in order to allow for execution in battery–powered nodes with limited computational and memory capabilities. Examples of such techniques include those based on Voronoi diagrams [15], virtual potential fields [20], or divide–and–conquer recursive methods [19, 21]. For an extensive taxonomy of coverage algorithms, the reader is referred to [22]. For our purposes, we will adopt the so–called *zoom algorithm* described in [14, 19], since it is distributed, computationally simple, and rather effective; nevertheless, any other distributed scheme could be implemented, as long as it can be handled by the nodes.

The goal of the zoom algorithm is to determine the largest coverage hole within a square region $\mathcal{P} \subseteq \mathcal{U}$. To this end, it partitions $\mathcal{P}$ into four non-overlapping sub-squares (northeast, southeast, southwest and northwest) and computes the number of non-covered cells in each of them. This procedure is repeated on the winning sub-square. The iteration continues until either (i) the corresponding sub-squares contain a single cell each, or (ii) there is a tie among the four sub-squares. In either case, the returned location of the coverage hole is the center of the square at the last step.

For our purposes, not only the location of coverage holes, but also their size, will be of importance. A simple size measure, which will be adopted in the sequel, is the total number of non-covered cells within the initial square region $\mathcal{P}$.

In [14, 19] the zoom algorithm is executed at each time step by each mobile node independently and based on the information about $\mathcal{U}$ locally available at the mobile node itself. On the other hand, in our approach the zoom algorithm is mainly executed by the static sensor nodes within communication range of a mobile node asking for a target location. In this way, the static infrastructure actively participates in the search for uncovered zones of the scenario. In order to estimate a coverage hole of a portion of the sensor field, a static sensor node has to be aware of the coverage status of such portion; in fact, the zoom algorithm aims at finding the biggest coverage hole in $\mathcal{P}$, so that the coverage status of $\mathcal{P}$ should be known by the executing node. As mentioned in Section 2.1, it is assumed that nodes broadcast beaconing messages containing their current location in U. Thus, upon reception of these messages a static node is always aware of the coverage status of the circle $\mathcal{P}_c$ centered at its own location and with radius r_c. Then, we build $\mathcal{P}$ as the square inscribed in $\mathcal{P}_c$. It follows that each static node can execute the zoom algorithm on the square centered at its location and with side equal to $\frac{\sqrt{2}}{2} r_c$.

Finally, as discussed in Section 3.3, mobile nodes will also autonomously execute the zoom algorithm so as to *(i)* estimate the location of the biggest coverage hole of particularly sparse zones of the sensor field not covered by any static sensor node, and *(ii)* escape from already almost-fully covered zones.

3.2. The bidding strategy

We regard a static sensor node is a 2–state–machine, represented in Figure 4 as an UML statechart [23]. There, a state is represented by a box, a transition by a directed arrow (the entry transition leaves from a black point) and the event causing a transition as an italic–capital–letters label.

Figure 4: Statechart of the static sensor node machine.

At time $k = 0$, the states of the static nodes are conventionally set to FREE. Since each mobile node i relies on the static sensor nodes within its communication range to obtain a target location, i publishes an auction in broadcast fashion. Upon the reception of an auction message, each static node $j \in N_i(k)$ whose state is set to FREE executes the zoom algorithm on its corresponding square $\mathcal{P}_j$ and bids the mobile node to cover the location of its biggest coverage hole. After a timeout, the mobile node closes the auction and evaluates the received bids. For the sake of simplicity, we consider the simplest evaluation criterion by which the best bid is the one offering the maximum number of uncovered cells. Nevertheless, other criteria can be combined together in order to obtain more sophisticated evaluations: this research direction is left as future work. The mobile node dispatches a *BEST_BID* event causing the best–bidding node to change its state to MONITOR[1]. This state prevents a static node to bid future auctions; moreover, a monitoring node will continuously check the coverage status of the target location until it is covered by a mobile node; in that case, this event (*TARGET_REACHED*) will be dispatched by the static monitoring node to the mobile node directed towards the reached target, with the following two effects: first, the state of the monitoring node is reset to FREE, and second, the mobile node publishes a new auction.

Figure 5 shows an example of the proposed bidding strategy involving two static sensor nodes (nodes 1 and 3) and two mobile nodes (nodes 2 and 4). At time k, mobile node 2 publishes an auction received by nodes 1 and 3. Being both in FREE state, nodes 1 and 3 independently execute the zoom algorithm on their own squares $\mathcal{P}_1$ and $\mathcal{P}_3$ respectively, and then bid the mobile node in the form (Tj, Nj), where Tj represents the location of the computed target, Nj represents the number of uncovered cells in $\mathcal{P}_j$ and $j = 1, 3$. Upon the reception of those bids, and assuming that $N3 > N1$, mobile node 2 decides to move towards $T3$ dispatching the *BEST_BID* event to node 3. As a consequence, node 3 enters the MONITOR state and keeps monitoring the coverage of location T3. Then, recalling that mobile nodes send beaconing messages containing their current location in U (such messages are labeled as *BEACON* in the figure), whenever node 3 realizes that $T3$ has been covered (by any mobile node in the most general case and

[1]We assume that the time interval between the publishing of an auction and the best bid event is much smaller than T.

by node 4 in this specific example) it dispatches the *TARGET_REACHED* event to mobile node 2 and then resets its own state to FREE. On the mobile node's side, the *TARGET_REACHED* event informs that the target has been covered and consequently it can publish a new auction.

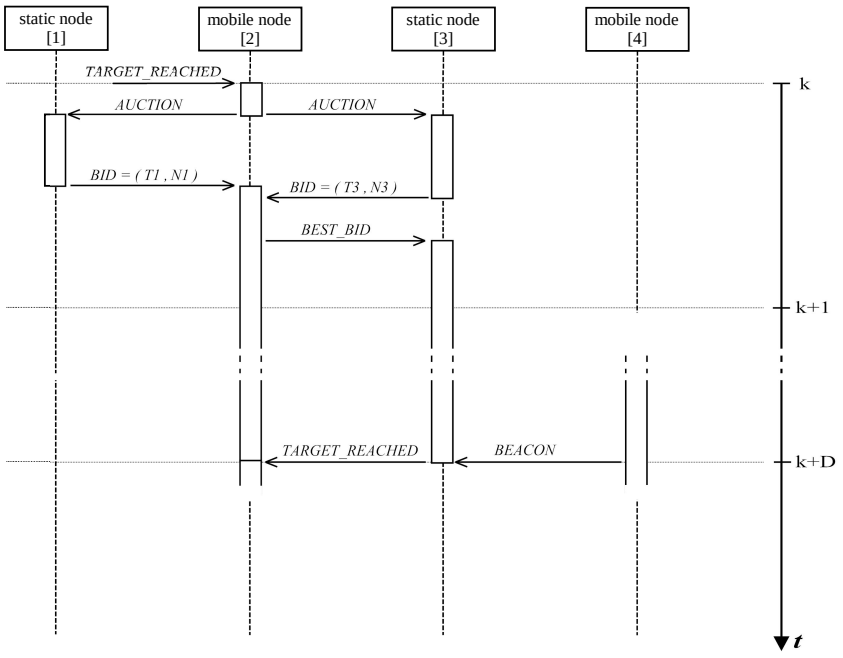


Figure 5: Sequence diagram of an auction involving 2 mobile nodes (nodes 2 and 4) and 2 static nodes (nodes 1 and 3). A bid is represented by a couple (Tj, Nj) representing the location of the target location and the number of uncovered cells computed by node j on the monitored portion $\mathcal{P}_j(k)$ at instant k.

From the above discussion it is clear that, upon publishing a auction, a mobile node receives a number of bids equal to the total number of static nodes within its communication range which are not in monitoring state and whose monitoring area $\mathcal{P}_j$ is not completely covered.

3.3. The coordination algorithm

The described bidding strategy may be unable to assist the mobile nodes traversing particularly sparse or completely covered zones of the scenario. Indeed, when a mobile node publishes an auction while far enough from

all the static sensor nodes, or when such auction is received only by nodes whose monitoring areas have been completely covered already, no static node will bid the mobile. In such situations, a mobile node cannot rely on the static infrastructure to obtain a target location. To let a mobile node react also in these situations, we endow it with an estimate $\hat{\mathcal{S}}(k)$ of the global coverage status $\mathcal{S}(k)$: whenever a mobile node receives no bid within a preset timeout, in order to obtain a target location it autonomously executes the zoom algorithm based on its estimate $\hat{\mathcal{S}}(k)$, and starting on the square with side $2 \cdot r_c$ centered at its current position. The mobile node will move towards the self–computed target location while publishing new auctions at each time step, until bidden.

It must be stressed that the consistency of a mobile node's estimate $\hat{\mathcal{S}}(k)$ with respect to the *actual* value of $\mathcal{S}(k)$ cannot be guaranteed if there are several mobile nodes operating in the scenario, since they may have visited different regions up to that time. Thus, in the proposed framework it is essential that mobile nodes update their corresponding coverage estimates whenever they happen to be within range of one another. This update is performed as follows: first, one of the mobile nodes sends its estimate to the other one, which merges it with its own estimate (the merging consists in a bitwise OR operation); then, the merging node sends the updated estimate back to the sender. Note that estimates $\hat{\mathcal{S}}(k)$ are binary maps which can be efficiently compressed (*e.g.*, [3]), so that the overhead added by these exchanges can be kept at bay.

Finally, when a mobile node is moving in an already completely covered zone, the locally executed zoom algorithm is also unable to assign a target location. In these situations, the mobile node will compute the point of intersection between its current trajectory and the square on which the zoom algorithm is executed. If such position happens to be within U, then it is assigned as the new target; otherwise, the new target is set at a prespecified *rendezvous point,* typically the center of scenario U. This strategy results particularly effective as the coverage of U increases, *i.e.*, as $\mathcal{S}(k)$ approaches $\mathcal{U}$; moreover, by directing the mobile nodes towards a rendezvous point increases their chances of merging their respective coverage estimators.

Figure 6 depicts the mobile node as 2–macro–state machine. At the beginning it is in the DISCOVER macro–state and keeps publishing auctions until at least a bid is received. Upon the reception of the bids, it selects the best one and changes its state to REACH; while in the latter state, it keeps moving towards the assigned target location, until the monitoring node will

dispatch the TARGET_COVERED event. At this point the state is reset to DISCOVER. On the other hand, if a mobile node in DISCOVER state is not bidden within a timeout, it executes the zoom algorithm locally. If the locally executed zoom algorithm fails to assign a target location, then the mobile keeps moving toward a "fake" target along the current direction, until such target lies out of bounds. In that case the target is switched to the rendezvous point (center of scenario), and so on.

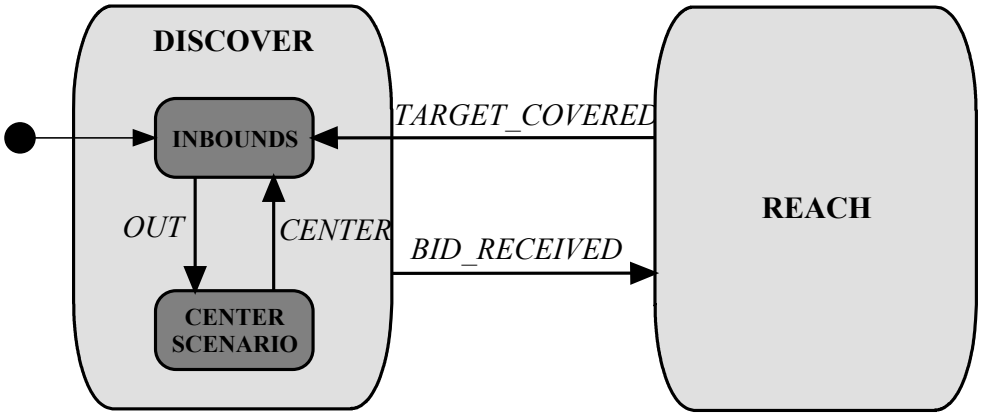


Figure 6: Mobile node's state machine.

4. Deterministic countermeasures

The increased number of active players on stage may result in a unreliable framework unless some deterministic countermeasures are deployed, as described next. Specifically, as we will see in Section 5.2 when comparing our technique with the one proposed by *Lambrou et al.* [14], to cover large percentages (*e.g.*, $> 90\%$) of U within a given maximum number of iterations is not a straightforward task. Moreover, the solution of this problem is not as simple as increasing the number of iterations to let the system evolve autonomously, as most of the times the mobile nodes are simply unable to find the remaining (usually small, sparse and far apart from each other) uncovered areas of U, because they enter loops. Hence, by analysing several simulation runs, we idientified some very specific situations in which, without relying on ad-hoc recovery procedures, some (resp., all) nodes enter loops and the

area coverage minimally increases (resp. do not increase) with the number of iterations, causing starvation (resp. deadlock).

4.1. Fixed target assignment

The first deterministic countermeasure, namely the fixed target assignment, was indeed implicitly developed in the previous section. It refers to the fact that, once a mobile nodes receives a bid from a static node, it changes its internal state to REACH, until the target location is covered by any mobile node. The static node that wins the auction changes its internal state to MONITOR to dispatch the TARGET_REACHED event once the target is covered.

The rationale of this strategy is three–fold. First, it allows a mobile node to follow a smooth trajectory towards the target. Second, it avoids the occurrence of cycles in which a mobile node starts circling around without reaching any target, due to successive target switching. This situation is depicted in Figure 7. There, at instant k the mobile node i in location $\boldsymbol{p}_i(k)$ decides to move towards target $T_i(k) = T_\star$ (winner of the latest auction). If the target assignment were not fixed and the mobile node kept publishing auctions at each instant, it might happen that, at instant $k + D$, the best bid yields a new target $T_i(k + D) = T'_\star \neq T_\star$ before $T_\star$ has been reached. In that case, the mobile node would start turning to reach such new target, possibly ending up, at instant $k + L$, in a location near $\boldsymbol{p}_i(k)$ (it should be remembered that turning is not instantaneous, but subject to maneuverability constraints). Then, if $\mathcal{S}(k) \approx \mathcal{S}(k + L)$ (*i.e.*, the coverage status has not substantially changed within the time interval $[k, k + L]$), then the best bid at $k + L$ would be likely equal to $T_i(k + L) = T_\star$. As a result, mobile node i would keep turning around in a cycle until $\mathcal{S}$ changes due to the movements of other mobile nodes (if they exist). Thus, in scenarios with few mobile nodes, the risk without a fixed target assignment is to have an infinite loop. Third, with a fixed target, mobile nodes have a better control of the path planning strategy; this will be further developed in the following.

4.2. Inhibition of the artificial repulsion force

As introduced in Section 2.2, an artificial repulsion force due to the term $J_2(\boldsymbol{p})$ in (7) is included as an additive term in (5), and its contribution is tuned by acting on the corresponding weight w_2. Basically, the introduction of $J_2(\boldsymbol{p})$ prevents mobile nodes from attempting to cover those cells already covered by the nearest static sensor node. Nevertheless, we have verified by

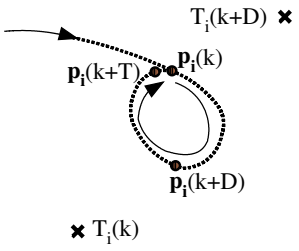
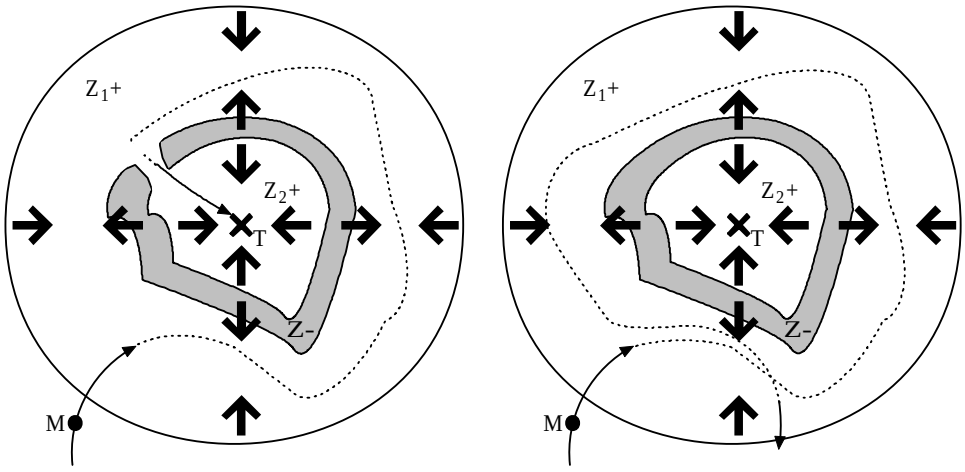


Figure 7: The target switching problem.

simulation that the effect of using (7) with a constant w_2 can be counter–productive from the area coverage point of view.

Indeed, it may happen that a target location assigned to a mobile node is close to a static sensor node (or to a group of them). Thus, as the mobile node approaches the target, the repulsion force exerted by the nearest static node may be strong enough to prevail over the attractive force exerted by the target itself, pushing the mobile node away from the target as a result. Eventually, as the mobile node moves far enough from the source of the artificial repulsion force (and from the target), the dominance of the attractive force due to $J_1(\boldsymbol{p})$ in (6) provokes a new approach to the target (and to the static sensor node). In such circumstance, the mobile node is likely to end up circling the target until finding a "hole" in the virtual barrier due to the static nodes' repulsion forces. With luck, the mobile node eventually finds such hole, and the effect of this phenomenon is just some degradation in the performance of the coverage algorithm; however, it may be possible that the assigned target will never be reached by the mobile node, if the latter is always repelled by the group of static nodes enclosing the target. These situations are depicted in Figure 8. A mobile node M (dot) describes the trajectory marked by the dotted line, with the goal of reaching the assigned target T (cross). Bold arrows represent the directions of the dominant force at the corresponding points; the gray areas $Z-$ represent zones in which the dominant term of (5) is the repulsion one, whereas the white zones Z_1+ and Z_2+ represent zones where the attractive force exerted by the target is the dominant term in (5).

In order to overcome these problems, we exploit the fact that the target location cannot change until accomplishment, and thus a mobile node may interpret an increase in the distance to the target as the result of a dominant effect of the repulsion applied by a static node, with respect to the attraction exerted by the target. Therefore, we let each mobile node temporarily inhibit



(a) A "hole" in the virtual barrier is eventually found.

(b) No "hole" is found: the target is never reached.

Figure 8: The barrier effect of the artificial repulsion force.

the repulsive effect of (7) by setting $w_2 = 0$ when a *repulsion counter,* available at each mobile node and initially set to 1, is equal to 0. In particular, at a given time the counter is decreased by one unit if the current distance to the target has increased with respect to the previous time instant; whenever the counter equals 0, the repulsive force is inhibited for a number of time instants equal to the maximum value the counter started from. After such variable number of time instants, the counter is reset to the previous maximum value, plus one unit. In general, whenever the repulsion force is inhibited for K time instants, either the mobile node will be able to pass through the repulsive zone within those K time instants, or the repulsion force will be restored for $K + 1$ time instants before being inhibited again for $K + 1$ instants. Once the target is reached, the repulsion counter is reset to 1 and the previous distance to target conventionally set to $+\infty$. With this countermeasure, a mobile node is able to pass through arbitrarily wide repulsion zones.

5. Results

In this section we evaluate the effectiveness of the proposed technique and compare it against the approach described in [14]. We do not include

comparisons with the method proposed by *Wang et al.*[15], since all the considered scenarios are too sparse to justify the use of relocatable nodes, while controllable mobile collectors represent the most natural and effective solution. For the sake of fairness, design parameters are set to the same values as those proposed by *Lambrou et al.*[14] whenever possible.

5.1. Simulation setup

We have built different simulation scenarios by randomly placing a variable number of static and mobile nodes in a 300 m $\times$ 300 m square region U, divided into squares of size $d = 1$ m to build $\mathcal{U}$. In particular, we have varied the number of static nodes s in $[0, 200]$ with step $= 10$, and the number of mobile nodes m in $[1, 5]$ with step $= 1$. For each couple (m, s) we have generated 50 random deployments and then executed our algorithm and the one proposed in [14], for a maximum of 8000 iterations (max_iter), averaging the results. In the following we will refer to these random deployments as *runs*. The communication range is fixed to $r_c = a \cdot \sqrt{2}$ m with $a = 32$ m, whereas the sensing range is fixed to $r_s = 8$ m. The mobile maneuverability constraints are set as $\mu T = 2$ m (with $T = 1$ s) and $\phi = \pi/6$, whereas $n = 20$ equispaced candidate positions are considered. Regarding the weights of the different cost functions we have fixed $w_3 = 1$, and initialized $w_1 = w_2 = 0.5$. Finally, the repulsion counter of each mobile node is initialized to 1 (cf. Section 4.2).

5.2. Simulation results

Figure 9 shows the results of the proposed technique for different (m, s) pairs[2], in terms of the mean number of iterations required to reach 95% coverage of $\mathcal{U}$ (*target coverage*); error bars refer to the standard deviation around the mean values. We recall that, while exploring the scenario, the mobile nodes may fail to reach a given target coverage within max_iter iterations. Such runs are marked as "failed" and excluded from the computation of the mean and standard deviation in Figure 9. The number of failed runs for the experiment in Figure 9 is shown in Figure 10. It is seen that as long

[2]Our simulation framework was entirely developed in Matlab 8.1.0.604 (R2013a) for a 32-bit Windows 7 OS, running on a standard laptop machine equipped with 2 Intel(R) Core(TM) i7-2620M CPUs @ 2.70 GHz and 4GB of Ram. Regarding the simulation times of the different runs, they were obviously depending on the actual simulation parameters, the specific algorithm and the iteration at which the target coverage was reached. Anyway we have verified that they never exceeded 1 minute.

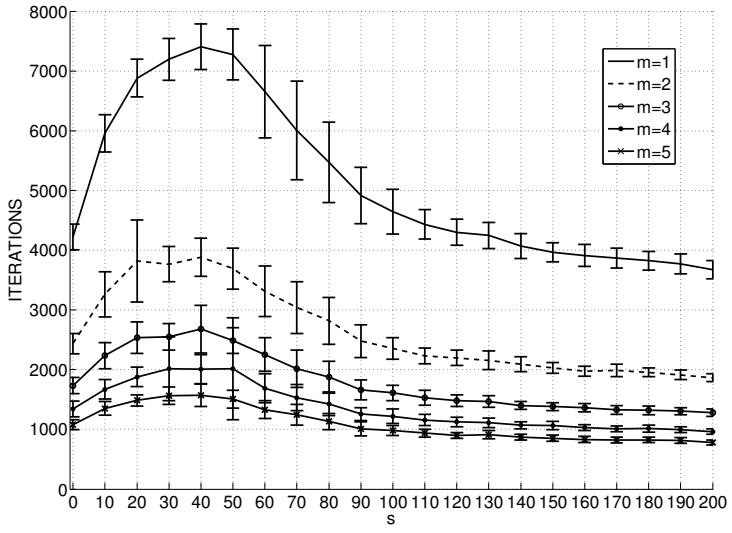


Figure 9: Mean number of iterations to reach the 95% target coverage using the proposed technique, for different (m, s) pairs.

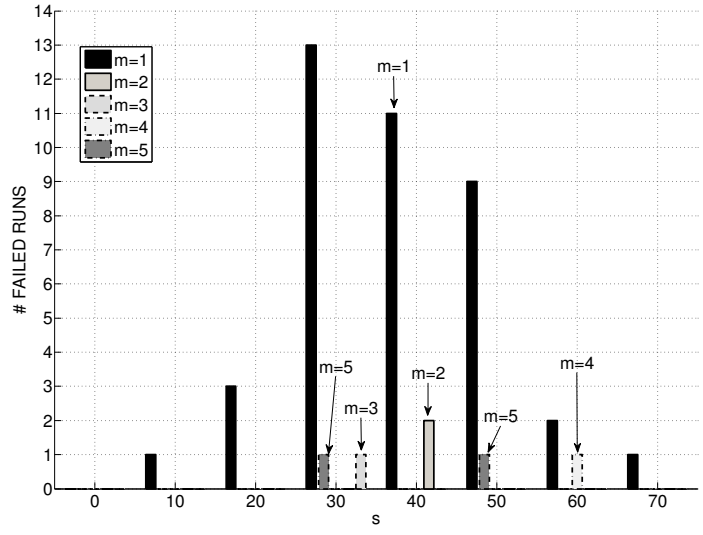


Figure 10: Number of runs (out of 50) for which the 95% target coverage was not reached within 8000 iterations, in the scenario of Figure 9.

as $m > 1$, the proposed technique is almost always able to reach the target coverage in this scenario within the specified time interval. For the case of a single mobile node ($m = 1$), and with the number of static sensors in the range $10 \leq s \leq 60$, the percentage of failed runs is not negligible, indicating some difficulty in reaching the target coverage. This is also reflected in Figure 9: for s in the mentioned range, the number of iterations required is perceivably larger. As the number of mobile nodes increases, this effect is attenuated.

For instance, scenarios $(m, s) = (1, 10)$ seem to require more iterations to reach the target coverage, with respect to scenarios $(1, 0)$. Specifically, when only $m = 1$ mobile node is available, the added cost of deploying s static sensor nodes actually worsens the coverage performance for values of s below some $s_\star \in [130, 140]$ (we will refer to $s_\star$ as the *break–even value*). Briefly, in scenarios $(1, s)$ with $s > s_\star$, the effectiveness smoothly improves as s increases. Yet attenuated, the described behavior is maintained also in scenarios with $m > 1$. The reason for this phenomenon resides in the fact that mobile nodes mainly rely on the static infrastructure to sample the coverage holes; hence, if the number of received bids is not sufficiently high, then the chance to move towards small coverage holes increases, with the corresponding degradation in terms of the time required to achieve the target coverage. By design, whenever a static node bids a mobile node, the latter stops executing its own zoom algorithm and immediately steers to cover the received target location. This happens even if an eventual local execution of the zoom algorithm at the mobile node would result in a better "self-bid" (*i.e.*, a larger coverage hole), and until the monitored area of the bidding node has been fully covered. Hence, a mobile node passing by a static node will exhaustively cover all the uncovered locations within the monitored area of the latter. It follows that, in sparse scenarios with relatively few static nodes, rather than following their own zoom algorithm in a purely greedy fashion, mobile nodes may spend some time before covering potentially small holes, which may slow down the coverage rate.

Table 1 gives the mean number of iterations it_0 (together with standard deviation values $[it_U, it_L]$) required in scenarios $(m, 0)$, and the corresponding intervals $[s_U, s_L]$ for the break–even value $s_\star$. In view of Figure 9 and Table 1, we can assert that in the considered scenario the proposed technique is beneficial whenever more than one mobile node is available and at least 90 static nodes are deployed in the area to be monitored. It is important, however, to realize that this conclusion strongly depends on the target coverage value

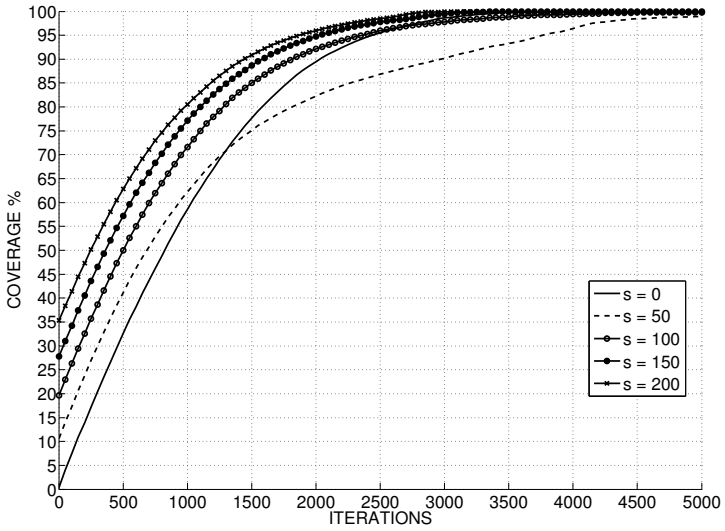
m	it_0	$[it_U, it_L]$	$[s_U, s_L]$
1	4220	$[4247, 4068]$	$[130, 140]$
2	2437	$[2477, 2356]$	$[90, 100]$
3	1734	$[1873, 1659]$	$[90, 100]$
4	1343	$[1428, 1256]$	$[80, 90]$
5	1079	$[1132, 1009]$	$[80, 90]$

Table 1: Break–even intervals for different mobile nodes, using the proposed technique.

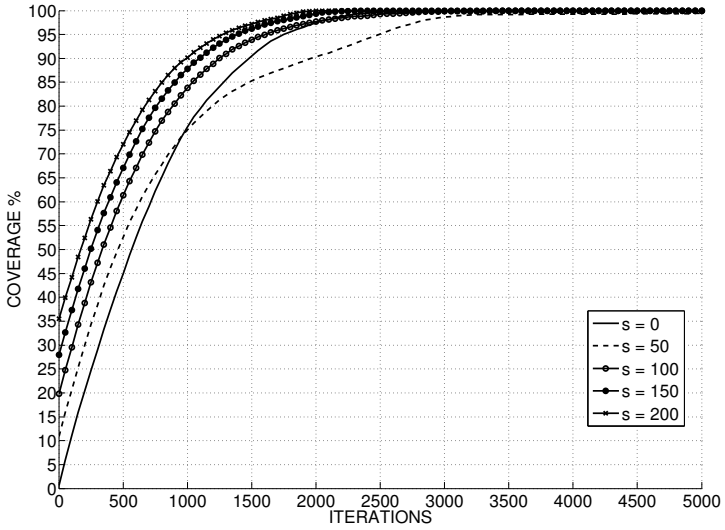
(95% in this example). To illustrate this, Figure 11 shows the mean coverage as a function of time, for $m = 2, 3$ and different number of static nodes in the same scenario. It is seen that, for a given m, the coverage rate (slope of the curves) at time zero is the same regardless of the size of the static infrastructure, with offsets determined by the initial coverage provided by all static nodes; this initial coverage rate is faster for larger number of mobiles m, as could be expected. As iterations progress, the coverage rate starts to drop gradually due to the effects described in the previous paragraph. This is more pronounced for sparse deployments ($s = 50$ in Figure 11). Nevertheless, it should not be hastily concluded that it is beneficial to dispense with the static infrastructure and let a few mobile nodes take care of covering the whole monitoring area, for the following reasons. First, although we have not considered this effect in the simulations, in practice a number of cells in the area may be inaccessible to mobile nodes (due to physical barriers, for instance), which must then rely on carefully deployed static nodes to cover those. Second, a static infrastructure provides an initial coverage advantage: depending on the application, it may well be required that a relatively low target coverage be reached as quickly as possible. For example, in Figure 11, for a target coverage of 50%, even the sparse deployments ($s = 50$) outperform the cases in which only mobiles are used ($s = 0$).

Figure 12 depicts the mean number of iterations versus reached percentage of the total coverage, for different values of m, by fixing the number of static sensor nodes to $s = 100$ and 200, respectively. We can note that, except for the different initial coverage offsets (about 20% with $s = 100$ and 35% with $s = 200$), the proposed technique behaves quite regularly. In terms of convergence speed, a sizable improvement is noticed by going from $m = 1$ to $m = 2$, whereas including additional mobile nodes results in diminishing returns.

In order to compare the proposed technique with that from [14], we con-

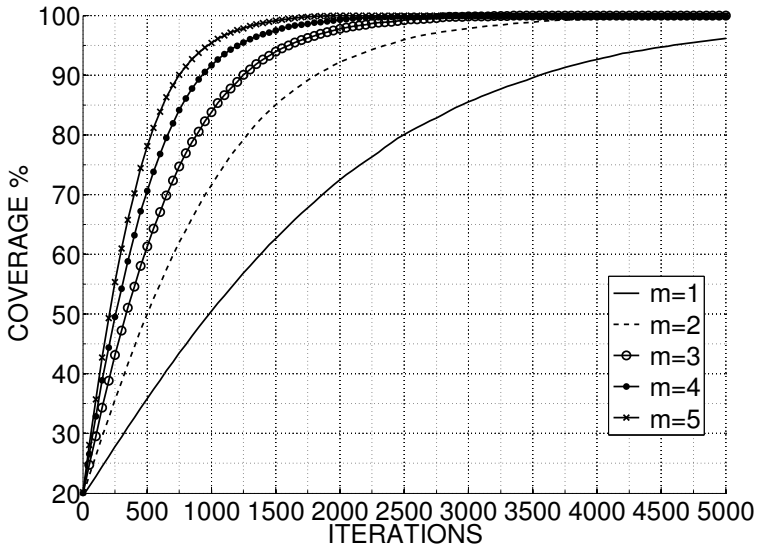


(a) $m = 2$.

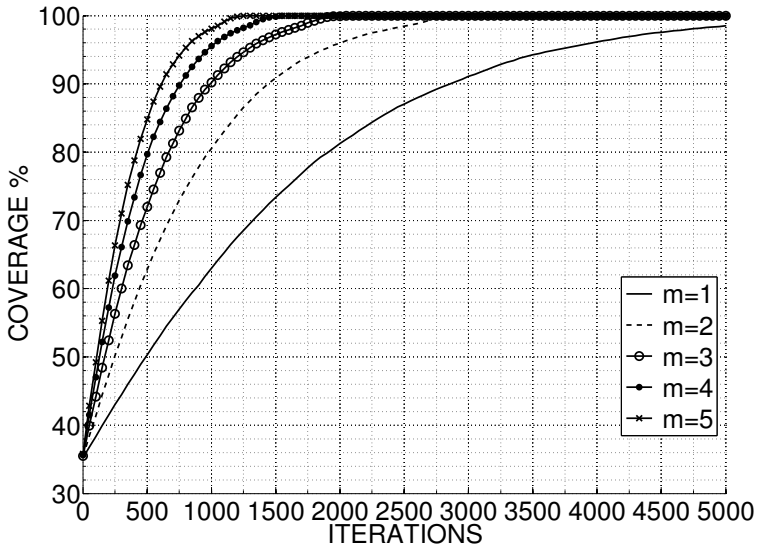


(b) $m = 3$.

Figure 11: Performance of the proposed technique in terms of iterations versus reached percentage of the total coverage for different number of mobile nodes.



(a) $s = 100$.



(b) $s = 200$.

Figure 12: Performance of the proposed technique in terms of iterations versus reached percentage of the total coverage, for different values of m, and for fixed s.

sider the availability of 3 mobile nodes and a 95% target coverage; Figure 13 shows the results of the two approaches for different values of s. It is seen that the approach from [14] is not affected significantly by the value of s, and that it is more effective than our technique for sparse static infrastructures (*i.e.*, $10 \leq s \leq 50$). This, again, is due to the behavior of the proposed approach in sparse settings discussed above. However, in denser settings ($s > 50$) our approach outperforms that from [14]. Note also that the performance of the latter presents a significant dispersion with respect to its average value. To better understand this uneven behaviour, it is instructive to look at the number of failed runs, shown in Figure 14. It is seen that this number remains significant for the approach from [14] even in denser deployments, in contrast with the proposed method. This indicates that the technique from [14] is facing substantial difficulties in order to reach the target coverage. Our method seems to be more stable, as evidenced by the smaller dispersion around the mean value observed in Figure 13.

The behavior of both methods in terms of coverage vs. time is shown in Figure 15, again with $m = 3$ mobile nodes and for two values of s, corresponding to a sparse setting ($s = 30$) and a dense one ($s = 150$). Loosely speaking, we can say that these two settings represent, respectively, the least and most favorable cases for our scheme. In the sparse deployment, it is observed that, after e.g. 1700 iterations, the method from [14] has reached 90% coverage, against 77% of our technique, whereas the 95% target coverage is reached after 2200 and 2550 iterations, respectively. In the dense deployment, and after 1100 iterations, our technique has reached a 90% coverage, against the 85% of the method from [14]; whereas the target coverage is reached after 1400 and 1900 iterations, respectively. Thus, the main advantage of the proposed scheme resides in denser settings: as previously mentioned, in our scheme mobile nodes will likely cover all of the area monitored by any static node they happen to pass by, and this may slow down the coverage rate in sparse scenarios. On the other hand, this behavior avoids leaving small and sparse coverage holes behind the mobile nodes. Indeed, we have verified that finding such small holes is often difficult for the mobile nodes; this is actually evidenced by the smaller dispersion about mean values seen in Figure 13 with respect to the method from [14].

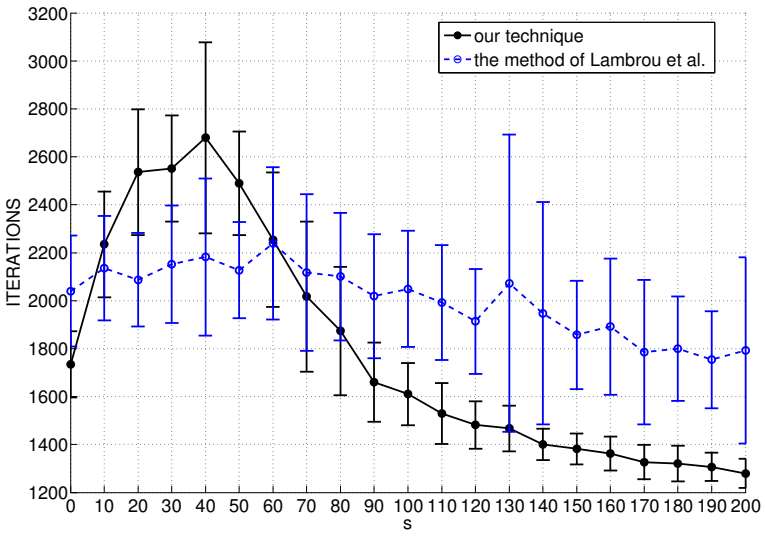


Figure 13: Comparison of the approaches in terms of iterations to reach a 95% target coverage, with $m = 3$ mobile nodes and different values of s.

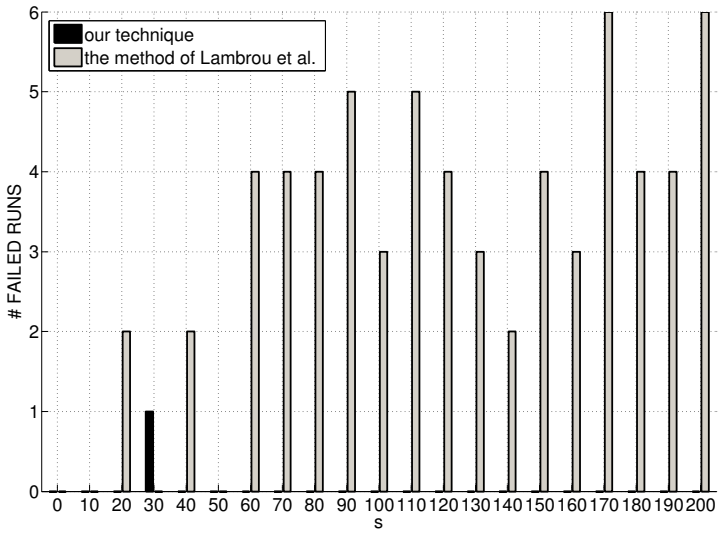


Figure 14: Number of runs (out of 50) for which the 95% target coverage was not reached within 8000 iterations, in the scenario of Figure 13.

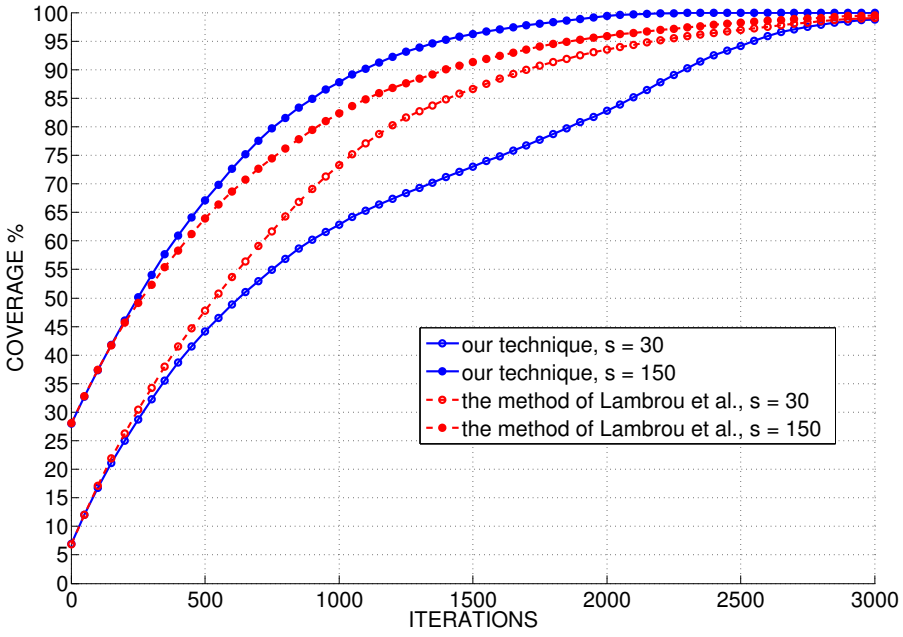


Figure 15: Comparison of the approaches in terms of iterations versus reached percentage of the total coverage, with $m = 3$ and $s = [30, 150]$.

6. Conclusions

We have proposed a technique to increase the sensing area coverage of monitoring WSNs composed by a static infrastructure complemented by mobile sensor nodes. Controllable trajectories of a reduced number of mobile nodes have been exploited in order to improve the coverage rate. The static infrastructure actively participates in the task of estimating coverage holes by means of a bidding mechanism, thus assisting the navigation of the mobile nodes towards such estimated holes, which is directed in a greedy fashion. A number of deterministic countermeasures have been presented in order to stabilize the behavior of the proposed technique. The resulting method enables effective area coverage in scenarios composed by a different number of static and mobile nodes, consistently outperforming a recently proposed technique for the same task in scenarios with a sufficiently dense static infrastructure, which constitute our main focus. Future work will address enhancing the performance of the proposed technique in settings with sparser deployments

by introducing further deterministic countermeasures, as well as dealing with the presence of subregions within the monitored area physically inaccessible to mobile nodes.

[1] T. Abdelzaher, S. Prabh, R. Kiran, On Real-time Capacity Limits of Multihop Wireless Sensor Networks, in: Proceedings of the 25th IEEE International Real-Time Systems Symposium, 2004, pp. 359–370.

[2] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, Directed Diffusion for Wireless Sensor Networking, IEEE/ACM Transactions on Networking 11 (1) (2003) 2–16.

[3] F. Marcelloni, M. Vecchio, An Efficient Lossless Compression Algorithm for Tiny Nodes of Monitoring Wireless Sensor Networks, The Computer Journal 52 (8) (2009) 969–987.

[4] J. Hill, System Architecture for Wireless Sensor Networks, Ph.D. thesis, University of California, Berkeley, aAI3105239 (2003).

[5] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless Sensor Networks: a Survey, Computer Networks 38 (4) (2002) 393–422.

[6] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. B. Srivastava, Coverage Problems in Wireless Ad-hoc Sensor Networks, in: Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001), Vol. 3, 2001, pp. 1380–1387.

[7] M. Younis, K. Akkaya, Strategies and Techniques for Node Placement in Wireless Sensor Networks: a Survey, Ad Hoc Networks 6 (4) (2008) 621–655.

[8] A. A. Somasundara, A. Kansal, D. D. Jea, D. Estrin, M. B. Srivastava, Controllably Mobile Infrastructure for Low Energy Embedded Networks, IEEE Transactions on Mobile Computing 5 (8) (2006) 958–973.

[9] G. Anastasi, M. Conti, M. Di Francesco, A. Passarella, Energy Conservation in Wireless Sensor Networks: A Survey, Ad Hoc Networks 7 (3) (2009) 537–568.

[10] M. Di Francesco, S. Das, G. Anastasi, Data Collection in Wireless Sensor Networks with Mobile Elements: a Survey, ACM Transactions on Sensor Networks 8 (1) (2011) 1–31.

[11] C. Liang, M. He, C. Tsai, Movement Assisted Sensor Deployment in Directional Sensor Networks, in: Proceedings of the 6th International Conference on Mobile Ad-hoc and Sensor Networks, 2010, pp. 226–230.

[12] T. Sung, C. Yang, Voronoi-based Coverage Improvement Approach for Wireless Directional Sensor Networks, Journal of Network and Computer Applications 39 (2014) 202–213.

[13] M. Vecchio, A. Viana, A. Ziviani, R. Friedman, DEEP: Density-based Proactive Data Dissemination Protocol for Wireless Sensor Networks with Uncontrolled Sink Mobility, Computer Communications 33 (8) (2010) 929–939.

[14] T. P. Lambrou, C. G. Panayiotou, S. Felici, B. Beferull, Exploiting Mobility for Efficient Coverage in Sparse Wireless Sensor Networks, Wireless Personal Communications 54 (1) (2010) 187–201.

[15] G. Wang, C. Guohong, P. Berman, T. F. La Porta, Bidding Protocols for Deploying Mobile Sensors, IEEE Transactions on Mobile Computing 6 (5) (2007) 563–576.

[16] G. Mao, B. Fidan, B. D. O. Anderson, Wireless Sensor Network Localization Techniques, Computer Networks 51 (10) (2007) 2529–2553.

[17] M. Polycarpou, Y. Yang, K. M. Passino, A Cooperative Search Framework for Distributed Agents, in: Proceedings of the 2001 IEEE International Symposium on Intelligent Control (ISIC '01), 2001, pp. 1–6.

[18] M. Polycarpou, Y. Yang, Y. Liu, K. Passino, Cooperative Control Design for Uninhabited Air Vehicles, in: S. Butenko, R. Murphey, P. Pardalos (Eds.), Cooperative Control: Models, Applications and Algorithms, Vol. 1 of Cooperative Systems, Springer US, 2003, pp. 283–321.

[19] T. Lambrou, C. Panayiotou, Collaborative Area Monitoring Using Wireless Sensor Networks with Stationary and Mobile Nodes, EURASIP Journal on Advances in Signal Processing 2009 (2009) 1–16.

[20] A. Howard, M. J. Mataric, G. S. Sukhatme, Mobile Sensor Network Deployment Using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem, in: H. Asama, T. Arai, T. Fukuda, T. Hasegawa

(Eds.), Distributed Autonomous Robotic Systems 5, Springer Japan, 2002, pp. 299–308.

[21] L. C. Shiu, C. Y. Lee, C. S. Yang, The Divide-and-conquer Deployment Algorithm Based on Triangles for Wireless Sensor Networks, IEEE Sensors Journal 11 (3) (2011) 781–790.

[22] N. Ahmed, S. S. Kanhere, S. Jha, The Holes Problem in Wireless Sensor Networks: a Survey, ACM SIGMOBILE Mobile Computing and Communications Review 9 (2) (2005) 4–18.

[23] D. Harel, Statecharts: a Visual Formalism for Complex Systems, Science of Computer Programming 8 (3) (1987) 231–274.