Autonomous Mobile Sensor Placement in Complex Environments

NOVELLA BARTOLINI, TIZIANA CALAMONERI, and STEFANO CIAVARELLA,
Sapienza University of Rome, Italy
THOMAS LA PORTA, Pennsylvania State University, USA
SIMONE SILVESTRI, Missouri University of Science and Technology, USA

7

In this article, we address the problem of autonomously deploying mobile sensors in an unknown complex environment. In such a scenario, mobile sensors may encounter obstacles or environmental sources of noise, so that movement and sensing capabilities can be significantly altered and become anisotropic. Any reduction of device capabilities cannot be known prior to their actual deployment, nor can it be predicted. We propose a new algorithm for autonomous sensor movements and positioning, called DOMINO (DeplOyment of MobIle Networks with Obstacles). Unlike traditional approaches, DOMINO explicitly addresses these issues by realizing a grid-based deployment throughout the Area of Interest (AoI) and subsequently refining it to cover the target area more precisely in the regions where devices experience reduced sensing. We demonstrate the capability of DOMINO to entirely cover the AoI in a finite time. We also give bounds on the number of sensors necessary to cover an AoI with asperities. Simulations show that DOMINO provides a fast deployment with precise movements and no oscillations, with moderate energy consumption. Furthermore, DOMINO provides better performance than previous solutions in all the operative settings.

Categories and Subject Descriptors: C.2.1,C.2.2 [**Computer-Communication Networks**]: Network Architecture and Design, Network Protocols

General Terms: Mobile sensor networks, Algorithms and protocols

Additional Key Words and Phrases: Wireless mobile sensor networks, autonomous coordination, communication and movement obstacles, obstacle sensing, path planning

ACM Reference Format:
Novella Bartolini, Tiziana Calamoneri, Stefano Ciavarella, Thomas La Porta, and Simone Silvestri. 2017. Autonomous mobile sensor placement in complex environments. ACM Trans. Auton. Adapt. Syst. 12, 2, Article 7 (May 2017), 28 pages.
DOI: http://dx.doi.org/10.1145/3050439

1. INTRODUCTION

Mobile sensors have enormous potential for monitoring fields that are inaccessible, unfamiliar, or even hostile. They could be used for environmental monitoring to track the dispersion of pollutants, gas plumes, or fires. They could also be used for public

This work is partially supported by NATO - North Atlantic Treaty Organization, under the SPS grant G4936 "Hybrid Sensor Network for Emergency Critical Scenarios." The work of T. Calamoneri is partially supported by the Italian Ministry of Education and University, PRIN project "AMANDA: Algorithmics for MAssive and Networked DAta."
Authors' addresses: N. Bartolini, S. Ciavarella, and T. Calamoneri, Computer Science Department, Sapienza University of Rome, Via Salaria 113, 00198 Rome, Italy; emails: {novella, ciavarella, calamo}@di.uniroma1.it; T. L. Porta, Computer Science and Engineering Department, Pennsylvania State University, 16801, Pennsylvania, USA; email: tlp@cse.psu.edu; S. Silvestri, Computer Science Department, Missouri University of Science and Technology, 500 West 15th Street, Rolla, MO 65409; email: silvestris@mst.edu.

DOI: http://dx.doi.org/10.1145/3050439

safety, for example, to monitor the release of harmful agents as a result of an accident. In such scenarios, sensors must be deployed from a distance, for example, sent from a safe location or dropped from an aircraft, and position themselves to provide the required sensing coverage. A sensor deployment algorithm is necessary to automate the positioning activity.

One of the major difficulties in turning mobile sensor networks into an everyday reality is that existing deployment algorithms have been designed to work only in a flat and smooth Area of Interest (AoI) or they assume detailed knowledge of the field prior to the deployment. Nevertheless, in real contexts, knowledge of the AoI may not be sufficiently accurate. Obstacles such as walls and trees may preclude device movement and sensing, while noise sources could reduce device sensing capabilities, such as in the case of acoustic sensors in a noisy area or cameras in fog. Therefore, a deployment algorithm should adapt sensor positions to the terrain and obstacles, and perform a denser deployment when required.

In this article, we provide a new algorithm, called DOMINO (DeplOyment of MobIle Networks with Obstacles), that addresses the problem of deploying mobile sensor fields containing unknown obstacles, terrain asperities, and noise sources. DOMINO is able to work with heterogeneous networks and successfully deploys sensors in both indoor and outdoor environments.

In order to evaluate the performance of DOMINO and its ability to work under the described scenario, we compare it with PDND (Parallel and Distributed Network Dynamics) [Ma et al. 2008], which is one of the most complete solutions to the deployment of mobile sensors. PDND is based on a virtual force model. It has excellent performance and proven termination. As the original version of PDND assumes smoothness of the field of interest, we extend it with additional rules to include management of obstacles and position-dependent device anisotropies. Performance comparisons are then carried out by means of extensive simulations.

The main contributions of this article are the following:

—We propose DOMINO, the first algorithm to make mobile sensors self-deploy in unknown environments with obstacles and noisy areas.
—We formally prove that DOMINO always terminates in a finite time, and it provides full coverage when sufficient sensors are available.
—We extend the algorithm PDND to make it work under the same operative scenarios addressed by DOMINO.
—We perform extensive simulations that show that DOMINO works efficiently even in complex scenarios with obstacles and asperities; furthermore, we experimentally compare DOMINO with both the original PDND and its extended variant and we show that DOMINO significantly outperforms PDND in all key performance metrics.

2. RELATED WORK

One possible approach to the deployment of sensors over a critical field that is inaccessible to humans is to use mobile robots preloaded with static sensors. An example of this approach is proposed in Li et al. [2014], where a number of mobile robots are scattered randomly in an unknown bounded two-dimensional field of interest, where terrain and boundary are not known prior to the deployment. Each robot drops the static sensors at the vertices of a virtual grid and adopts a backtracking algorithm to guarantee full coverage of the area of interest. A similar approach is proposed in Ines et al. [2014], who propose the use of a team of robots to relocate the least number of sensors to guarantee full coverage and avoid physical obstacles in an unknown region. While these approaches address a similar problem to ours, they rely on the use of medium-size robots to carry static sensors. Small-size mobile sensors are more appropriate if the environment is hostile. Furthermore, in a critical environment, having

multiple static sensor nodes carried by a single mobile device would create vulnerable points of failures for a large part of the devices to be deployed.

For this reason, we address the problem of deploying a team of small-size, low-cost mobile sensors, where each of them moves on its own and communicates with the others to expand coverage and to regulate positioning conflicts.

Some works on robotics provide approaches to the problem of autonomous coordination of mobile robots. The work presented in Dieudonné et al. [2008] considers a team of mobile sensing robots and provides a coordination protocol that makes them assume a circular formation, starting from any arbitrary positioning. Gilbert et al. [2009] study the problem of autonomous coordination of mobile robots over a field of interest, even in the presence of changes in the underlying ad hoc network, and in the team of robots. The objective is to let robots arrange themselves to form a particular pattern, such as a uniform distribution along a curve or a bird flock formation. Area sensing coverage is not the objective of the formation, and for this reason problems such as sensing anisotropies possibly due to obstacles are not addressed. The algorithm proposed in Guo et al. [2012] provides a multirobot pattern formation and a boundary coverage approach that also works in the presence of moving obstacles. The approach is inspired by biological morphogenesis. It aims at making the autonomous robots form specific patterns, such as bird flocks, or boundary coverage.

Unlike DOMINO, the approaches in Dieudonné et al. [2008], Gilbert et al. [2009], and Guo et al. [2012] do not address the problem of area coverage and the related problems such as sensing and communication heterogeneity and anisotropy in the presence of obstacles.

A previous solution to mobile sensor deployment is the virtual force approach (VFA). It models the interactions among sensors as a combination of attractive and repulsive forces. Drawbacks of VFA include complex tuning of several parameters and sensor oscillatory behavior. Possible improvements to decrease the oscillations include the introduction of dissipative forces [Howard et al. 2002; Garetto et al. 2007] or the definition of arbitrary thresholds as stopping conditions [Chen et al. 2007; Heo and Varshney 2005]. The PDND algorithm [Ma et al. 2008] described in Section 6 is one of the best-performing VFA-based solutions and has a guaranteed termination. Bartolini et al. [2014a] consider for the first time the vulnerabilities of deployments based on VFA, proposing a new protocol that provides total coverage also in the presence of a malicious attacker. However, they assume that the area of deployment is obstacle-free, and sensing and communications are isotropic.

Other approaches are inspired by fluid models, such as in Pac et al. [2006], or by gas models, such as in Kerr et al. [2004]. Similar to the VFA, these works suffer from oscillatory behavior.

Among the works on mobile sensor deployment, many adopt computational geometric structures such as Voronoi diagrams [Wang et al. 2006] and Delaunay triangulation [Ma and Yang 2007] in order to identify responsibility regions that are used to guide sensor movements. These geometric structures can be successfully adopted only under the assumptions of obstacle-free AoI and a Boolean sensing model. Moreover, Bartolini et al. [2014b] study the vulnerabilities of the Voronoi-based deployment algorithm and propose related countermeasures. Although some works consider heterogeneous sensors [Bartolini et al. 2011b] and nonuniform sensing [Fagiolini et al. 2008; Gusrialdi et al. 2009], they still assume predictable heterogeneity and anisotropy of the sensing capabilities and do not address the scenarios considered in this article.

A completely different approach is based on the construction of regular patterns over the AoI. This technique is motivated by the studies on static sensor deployment [Bar-Noy et al. 2009; Johnson et al. 2009b]. According to these approaches, mobile sensors coordinate themselves in order to form a regular pattern such as a hexagonal

grid [Bartolini et al. 2011a; Falcon et al. 2011] or a stripe-based pattern [Tan et al. 2009a]. Although these approaches may obtain optimal patterns, depending on the ratio between r_{tx} and r_s, they cannot be applied to the complex scenarios considered in this article. Indeed, as discussed in Section 3, the presence of obstacles and asperities makes it difficult to calculate an optimal deployment. Furthermore, regular patterns would not guarantee full coverage and connectivity due to anisotropic sensing and communications. Notice that DOMINO falls in this category. Nevertheless, the regular pattern provided by DOMINO is autonomously adapted and distorted by the mobile sensors in order to adapt the network configuration to the environment, and additional sensors are placed in order to achieve full coverage in noisy areas.

Finally, there are miscellaneous other approaches that do not fall into any of the previous categories, which make use of different techniques such as a hybrid-pattern-based virtual force approach [Tariq et al. 2010], system theory applied to biological systems [Martinez et al. 2007], and analytic modeling of animal behavior [Ozturk et al. 2011; Tua et al. 2012]. Nevertheless, such solutions cannot be applied to the realistic scenarios considered in this article as they also assume isotropic device capabilities, an obstacle-free environment, and device homogeneity.

3. MOBILE SENSOR DEPLOYMENT IN UNKNOWN AND IRREGULAR FIELDS

Obstacles and terrain asperities are typically present in an AoI and alter device movement and sensing capabilities, causing position-dependent anisotropies. This new scenario brings up the need to solve new problems related to device deployment: sensors need to adjust their positions in order to cover the terrain even when obstacles are present, and to perform a denser deployment around areas that are subject to noise sources; possible errors of the positioning system may occur, which may be worsened by the presence of terrain asperities and obstacles; furthermore, a path planning algorithm is necessary to let sensors efficiently circumnavigate obstacles and determine the reachability of a desired position. To the best of our knowledge, DOMINO is the first algorithm that addresses all of these problems and is able to work in such a realistic and complex scenario.

We consider N mobile sensors deployed in indoor and outdoor operative settings. When sensors are deployed in an indoor scenario, we assume they are sent from a safe region and therefore initially form a connected component. Therefore, any sensor can determine its relative location inside the AoI by cooperating with neighbor sensors [Patwari et al. 2005; Bulusu et al. 2000]. Nevertheless, this assumption is not valid if sensors are to be deployed outdoors, where they can be randomly dropped from an aircraft. Therefore, in this latter scenario, we consider nodes endowed with low-cost GPS.[1] Notice that DOMINO can work without GPS even in an outdoor scenario, if sensors execute a preliminary rendezvous technique that can be designed by taking inspiration from the work of Tan et al. [2009a], to which we refer for details. In this article, we only consider the cases of indoor deployment and outdoor with GPS.

When working in an obstacle- and noise-free AoI, a sensor s_i has a *sensing radius* $r_s^{(i)}$ and *communication radius* $r_{tx}^{(i)}$, with $i = 1, \ldots, N$. By contrast, when deployed in an AoI containing obstacles and noise sources, the sensing capabilities can be reduced and become anisotropic. We assume that every sensor can detect obstacles in its sensing range, for example, by using ultrasonic waves [Ohya et al. 1996]. By using this technique, devices are able to estimate their sensing coverage extension with a certain

[1]Low-cost GPSs currently available provide accuracy in the orders of a few decimeters [Beran et al. 2007] and have a cost of around $ 200 per unit [Memsic datasheet 2017].

precision. Furthermore, sensing devices can be heterogeneous. In such a case, we define $r_s^{Max} = \max_{i=1,\ldots N} r_s^{(i)}$ and $r_{tx}^{min} = \min_{i=1,\ldots N} r_{tx}^{(i)}$.

We anticipate that DOMINO positions at least one sensor in each accessible tile of a regular square grid drawn over the AoI. We consider a grid where each squared tile has side $\sqrt{2} \cdot r_s^{Max}$. This setting ensures that the only sensors that can cover an entire tile are the ones working with maximum sensing range. The algorithm works independently of the grid size. Nevertheless, a smaller grid would possibly create unnecessary coverage redundancy. By contrast, as we detail in Section 4, when one sensor alone is not sufficient to entirely cover its responsibility region, DOMINO positions other additional sensors where necessary.

We assume that noise and communication obstacles do not preclude the communication between any two sensors positioned in adjacent tiles. This assumption guarantees connectivity of the final deployment provided that the AoI is not partitioned in disjoint portions. In order to meet this assumption, the transmission range of a sensor should never be reduced below the value of $\sqrt{10} \cdot r_s^{Max}$, which is the maximum distance between any two positions in the rectangular area formed by two adjacent tiles (corresponding to the diagonal of such a rectangular area with sides $\sqrt{2} \cdot r_s^{Max}$ and $2 \cdot \sqrt{2} \cdot r_s^{Max}$).

Notice that this assumption can be removed so as to consider both the case of power attenuation and the case of obstacles that shield communications, by allowing the use of auxiliary sensors for connectivity purposes around obstacles or through asperities when needed according to a technique described in the the work of Tan et al. [2009a].

Finally, we assume that the communication protocol implementing DOMINO is based on an underlying communication protocol stack that ensures error-free communications by means of retransmissions.

4. THE DOMINO ALGORITHM

4.1. Target Deployment of DOMINO

When DOMINO is executed over a smooth and flat AoI and with homogeneous sensors, the target deployment is a squared grid of sensors where each tile is assigned to one sensor only. In such a case, it is possible to size the squared grid so that each sensor completely covers its tile and is able to communicate with the four adjacent neighbor sensors in the grid.

Nevertheless, if DOMINO is performed in a more realistic environment, it can be impossible to obtain such a regular grid deployment because some grid positions may be unreachable and a single sensor may be insufficient to cover the whole tile due to obstacles and asperities.

We hereby use the term *skeleton grid* to refer to the regular squared grid that DOMINO would create in the case of a flat and smooth AoI. We assume that the coordinates of the skeleton grid points are known to all sensors being deployed. We underline that knowledge of the skeleton grid coordinates does not require knowledge of obstacles, asperities, and in general the reachability of these locations in the AoI. Indeed, the skeleton grid is used to guide the movements of the sensors in the area, and the actual final deployment may be substantially different from the tile centers of the skeleton grid.

The size of the grid is not a primary issue for the definition of the algorithm, but it can affect the coverage redundancy and the number of required sensors. We propose to set the length of the grid side to $l_s^{Max} = \sqrt{2} r_s^{Max}$. The wiggle room technique [Johnson et al. 2009b] can be used to accommodate small errors in sensor positioning by slightly reducing the grid side. Notice that by setting this side length, a single sensor may not always be sufficient to cover the entire tile. In fact, in the regions in which the AoI is not flat and smooth, and sensors have reduced sensing capabilities or sensing radius

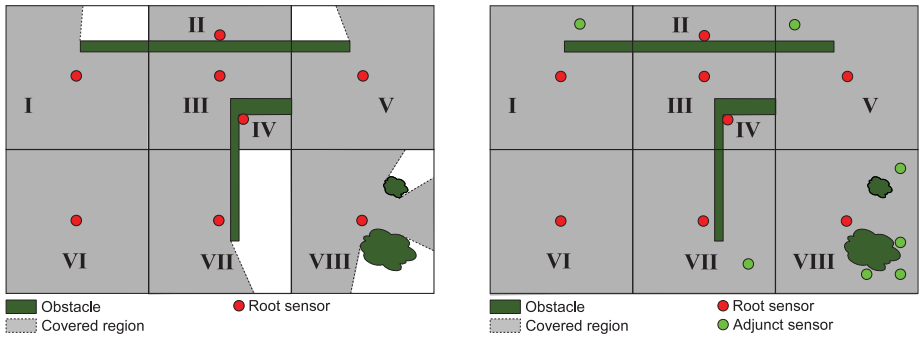


Fig. 1. Example of AoI partitioning (a), and example of refined deployment with adjunct sensors (b).

lower than the Maximum value r_s^{Max}, the tiles of the skeleton grid will require more than one sensor to be covered.

A *partitioning obstacle* is an obstacle that partitions a grid tile in several disconnected portions, such that a sensor must traverse at least one other tile in order to move from one side of the obstacle to the other. DOMINO partitions the AoI into *responsibility regions*. A responsibility region is the area of a tile that is delimited by its borders and by any partitioning obstacle crossing the tile. Notice that a tile that does not contain any partitioning obstacle constitutes a single responsibility region.

The target deployment of DOMINO consists of having at least one sensor, hereafter called the *root* sensor, in any responsibility region. All sensors lying in the responsibility region of a root sensor act as its *slaves*. If a root sensor does not entirely cover its responsibility region, it turns some of its slaves into *adjunct sensors* and deploys them in key positions in the same tile, according to a recursive *regridding* mechanism. The grid formed by root sensors is called the *top-level* grid. Two responsibility regions belonging to two different tiles of the top-level grid are called *adjacent* whenever they have a common boundary portion that can be traversed by a moving sensor. Such traversable boundary portions are called *corridors*. DOMINO requires every root sensor to be in communication with all its adjunct sensors and with the root sensors belonging to adjacent responsibility regions (*adjacent root sensors*).

An example of AoI partitioned into responsibility regions is shown in Figure 1(a), where the AoI is tessellated into six squared tiles and eight responsibility regions. The responsibility regions II, III, and IV belong to the same squared tile and are disconnected by two partitioning obstacles. The responsibility region III is adjacent to regions I, V, and VII, but it is not adjacent to regions II and IV. Similarly, region V is adjacent to regions II, III, IV, and VIII. The root sensors of the responsibility regions numbered I, V, VII, and VIII do not cover their regions completely, because of the presence of sensing obstacles; therefore, DOMINO places adjunct sensors to complete the coverage of their regions, as shown in Figure 1(b).

4.2. Outline of the Algorithm DOMINO

In this paragraph, we outline the basic actions provided by the algorithm DOMINO. We will give a more detailed description of each action in Section 4.3, where we address every potential problem that each sensor may have to face when performing each algorithm action, such as how to solve positioning conflict among several sensors or where to go when a desired location is inaccessible due to the presence of obstacles.

The algorithm actions are executed by each sensor in a distributed and interleaved manner. No synchronization among sensors is necessary.

Every sensor that has not yet been involved in other algorithm activities by other sensors tries to position itself (*Snap*) in the closest skeleton grid point and starts the grid formation from there. After a successful snap action, the sensor becomes the root of its own responsibility region. In order to extend the grid formation, a root sensor checks whether its adjacent responsibility regions are already under the control of other root sensors. It then sends a slave to snap into any ungoverned region. In general, this results in the creation of several aligned grid portions, which eventually grow and merge into a unique grid. *At the end of a Snap action, new root sensors are deployed to cover new responsibility regions; hence, global coverage is increased.*

If a root sensor determines that its responsibility region is not completely covered, it runs a local grid refinement (*Grid Refinement* activity) in order to place some of its slaves in key positions for increasing coverage within its tile. After such an activity, the involved slaves take the role of adjunct sensors. Adjunct sensors that become redundant after the placement of other adjunct sensors are released and made available again for other purposes. *At the end of the Grid Refinement activity, the coverage of a tile is increased.*

If a root needing to complete the Snap or the Grid Refinement activities has no slaves available, it invites slaves from other regions by means of a *Pull* action. Such an action is auxiliary to the execution of either a grid refinement or a snap action and is never executed alone. For this reason, *at the end of a Pull action, an increase of coverage is always observed (although the actual increase in coverage may be due to the placement of a sensor other than the one that received the invitation).*

Some slaves in a given responsibility region can be redundant because their region is completely covered and all adjacent root sensors have been positioned. In such a case, the root sensor pushes its redundant slaves toward regions with lower density, if any, according to the so-called *Push* activity. *At the end of a Push activity, the slave density is uniformed among adjacent tiles.*

All sensor movements provided by DOMINO follow the path planning algorithm called Area Limited Bug 2 (*AL-Bug2*) described in Section 4.4.

4.3. The DOMINO Activities in Detail

In this section, we describe in detail the activities of DOMINO. Figure 2 shows a flowchart of these activities.[2]

4.3.1. Snap. The Snap action is performed by sensors in order to expand the top-level grid. Each sensor that has not yet been involved in other algorithm activities positions itself in the closest point of the skeleton grid and becomes a root sensor of the corresponding responsibility region. In particular, when performing a snap activity, sensor t tries to reach its target skeleton grid position. If such a position cannot be reached due to the presence of obstacles, t performs an exploration of the part of the obstacle it can reach without exiting the tile, and then it stops in the point of the tile at minimum distance from the tile center. If there are multiple positions at such a minimum distance, an additional condition, such as being the northern position or any other arbitrarily chosen orientation, is used to ensure the uniqueness of the snap position. This is necessary because two or more sensors may try to snap in the same region R, causing a *positioning conflict* that can be solved only if the snap position is unique, according to a timestamp criterion.

[2]The figure focuses on the actions performed by a sensor becoming the root of a responsibility region. Actions performed by slave and adjunct sensors are omitted for ease of representation.

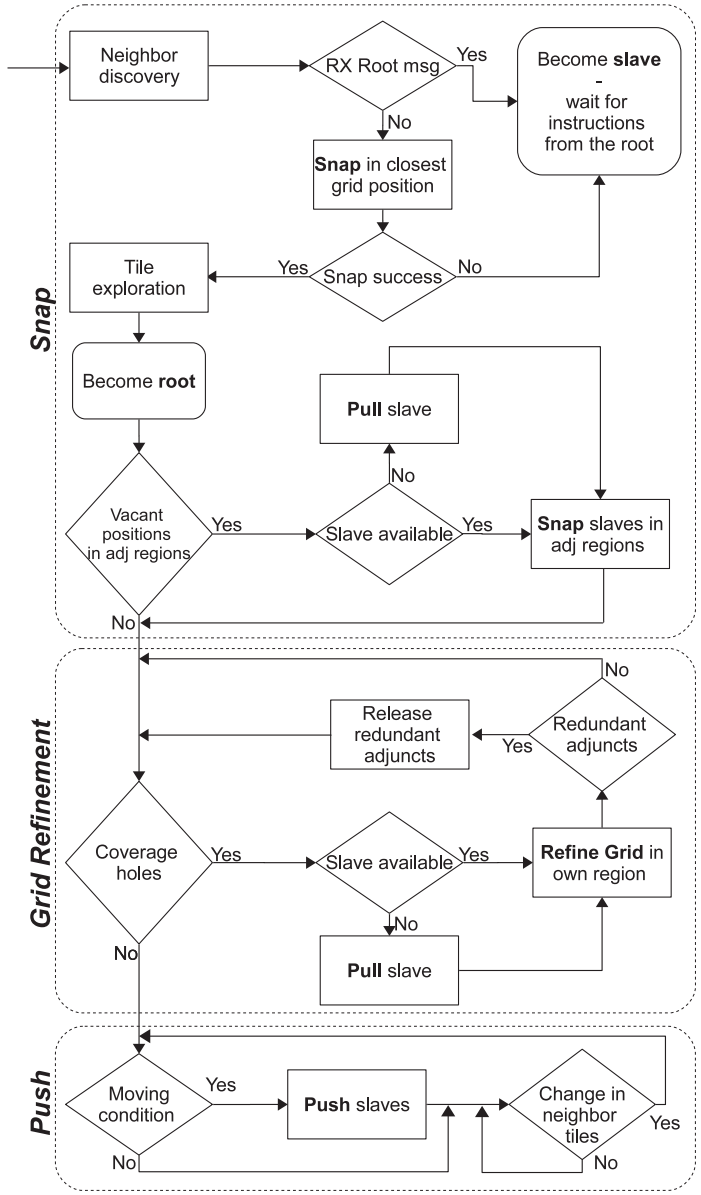


Fig. 2. Flow chart of DOMINO activities.

Let t be the sensor that won the positioning conflict. Before acting as the root, sensor t needs to determine the border of its responsibility region. Notice that for this purpose, DOMINO requires that t explores every unknown part of the border of its responsibility region. After this exploration, t assumes its snap position and, from there, it first determines the presence of coverage holes and then broadcasts a message by which it declares itself as the root of the region. This message also contains the coordinates of its position and of the border of its responsibility region. The sensors located within the region reply to this message with their position and switch their status to slave of the

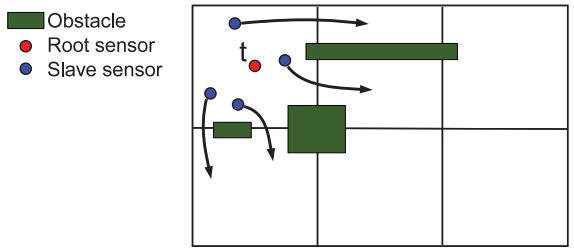


Fig. 3. An example of the Snap activity.

newcomer root sensor. The adjacent root sensors also reply to this message to notify t of their presence. The root sensor t tries to expand the deployment by sending slaves in ungoverned adjacent regions to make them snap. When t determines the presence of an adjacent region R without any root sensor governing it and has available slaves, it sends one of its slaves, s, to snap in the ungoverned adjacent region, with the purpose to make it become its root. If there are ungoverned adjacent regions but there are no slaves available, t resorts to the pull activity to invite new slaves, as described in Section 4.3.4. Notice that the presence of obstacles requires the root to send an available slave, s, for each corridor detected on the border of the responsibility region of t. Consider the scenario illustrated in Figure 3. Since t has complete knowledge only of its tile, it is not able to determine whether two or more corridors lead to disconnected responsibility regions or to a single one. Potential positioning conflicts are solved considering timestamps.

At the end of this activity, the former slave sensor s, now a root, has complete knowledge of the number and positions of its slaves, the identity and position of its adjacent sensors, the border of its responsibility region, and the corridors connecting to its adjacent responsibility regions. Furthermore, it is in communication with node t, which is the root sensor that sent s to snap.

As a consequence of the snap activity, each sensor becomes either a root or a slave. This activity results in the creation of several connected components forming separate grid portions that expand through the AoI. In the hypothesis that enough sensors are available to cover the AoI entirely, and that all the points that need to be covered in the AoI are reachable by the moving sensors, the subsequent activities provided by DOMINO ensure that these grid portions eventually expand enough to come close to each other and merge in a single connected grid.

4.3.2. Grid Refinement. If a root sensor t of a responsibility region R detects coverage holes inside R, it performs the *Grid Refinement* activity. Sensor t refines the grid by placing additional adjunct sensors in new locations within R corresponding to the node positions of a quad-tree structure as follows.

The root sensor t virtually divides the squared tile containing the region R into four squared *subregions* and evaluates the coverage of the subregions that intersect R. If a coverage hole is detected in a subregion, an adjunct sensor (either a slave of t or a sensor invited by t through the *Pull* activity) is positioned in the center of the subregion (or in the closest position to the center and to the north direction, or to any other arbitrarily chosen direction if the center is not reachable).

Every adjunct sensor recursively analyzes the coverage of its subregion and, if necessary, repeats the process by dividing it into the other four subregions and positioning additional adjunct sensors.

The recursive grid refinement of the subregions is performed in a breadth-first search fashion, that is, level by level, and sequentially to avoid placing unnecessary

sensors if an adjunct sensor of a subregion is able to also cover other adjacent subregions.

The process terminates as soon as either the considered responsibility region is completely covered or all the available sensors have been definitively positioned. Additionally, in order to avoid the creation of a too-deep quad-tree, we limit the level of regridding steps to a Maximum fixed level $k_{\texttt{Max}}$ at the expense of a small loss in coverage.[3]

Some adjunct sensors, which have been placed at a given instant to fill a coverage hole, may become unnecessary at a later time, due to the subsequent placement of other additional adjunct sensors in the tile. In this situation, the unnecessary adjunct sensors alert the root sensor. The root sensor t selects, among the redundant adjunct sensors, the ones that can be *released* from the quad-tree and can be made slaves again. This procedure, corresponding to the "release redundant adjuncts" block of Figure 2, ensures that no adjunct sensor in R is redundant at the end of the grid refinement activity performed by t.

In order to avoid possible stale situations that may occur as a consequence of the regridding operations, we provide the following *cascaded release process*.

If an adjunct sensor u needs other adjunct sensors in its subtile, it notifies the root sensor t of this requirement. As it is likely that the placement of adjunct sensors makes u and some of its ancestors in the quad-tree unnecessary for coverage, the placement of the adjunct sensors in the subtile of u occurs according to the following process: the adjunct u moves itself to one of the candidate positions for new adjunct sensors in its subtile. This movement is notified to the lowest ancestor of u, which will release itself and take the previous position of sensor u only if it is necessary for coverage. This process could recursively proceed along the path from u to the root t. If while performing such a cascaded release process the root is notified that some released ancestor of u is still necessary for coverage, it replaces it by selecting a new slave if available or issuing a pull action if necessary. Notice that the root of a responsibility region is never released even in the case of redundancy.

The Snap and Grid Refinement activities govern the expansion of the top-level grid and its refinement, respectively. We introduce a timeout Δ, before which no regridding is allowed. Δ needs to be tuned according to the network coverage goals: before the expiration of the timeout, a quicker expansion takes priority over grid refinement, and vice versa after its expiration.

4.3.3. Push. Once it has completed the coverage of its responsibility region, a root sensor t may still have some available slave sensors that can be useful somewhere else in the AoI. In this case, it proactively pushes the extra slaves toward adjacent regions with fewer slaves. Before performing a push activity, a root sensor exchanges a message with the adjacent root sensors concerning the respective current number of slaves.

Among the adjacent regions, t selects the destination region of the push action as the one that has the lowest number of slaves; among the slaves that can be pushed, t selects the one that is closest to the destination region.

In order to avoid endless cyclic movements of slaves, the push activity from the region of the root sensor t to the region of the root sensors s is allowed only if the following *moving condition* is verified:

$$\{S(t) > (S(s) + 1)\} \vee \{S(t) = (S(s) + 1) \wedge id(t) > id(s)\}, \tag{1}$$

[3]Notice that this limit is needed to have a theoretically guaranteed termination and also to account for the inherent size of devices, which makes it impossible to have an infinite number of regridding steps. A value of $k_{\texttt{Max}}$=2 was sufficient in all our experiments to have both termination and complete coverage.

where $S(\cdot)$ is the number of slaves of the sensor in argument and $id(\cdot)$ is the unique identity code of the sensor radio device, namely, its MAC address, or any other serial number that can be used to uniquely identify each sensor device. The moving condition is verified if t has at least two more slave sensors than s, or if t has exactly one more slave sensor and its id is greater than the id of s. This condition prevents ping-pong effects that may occur when the difference in the number of slaves is only unitary. The only slave movement allowed in such a case is in the direction of decreasing id.

4.3.4. Pull. The *Pull* activity provides a reactive method to attract sensors where needed. Notice that this is an auxiliary activity and is never executed alone. When a root sensor s needs new slaves to expand the grid through a Snap or to refine the deployment through a Grid Refinement, it issues a Pull request for a given number $k \geq 1$ of new slave sensors. The sensor s sends an invitation message, which is flooded from one responsibility region to the next adjacent one. This message flooding has a limited horizon h, initially set to 1, and increases sequentially until the necessary slaves are found.

If a root sensor t having some available slaves receives this message, it answers to the invitation by sending a slave proposal message to s. The slave proposal message contains information regarding the number of slaves available in t's region. The proposed slaves are then considered as reserved by t until they are either accepted or refused by s. Slaves are never proposed to multiple root sensors performing a pull action. If the proposals received by s contain at least k sensors, s sends an acknowledgment message to the roots from which it wants the k slaves to come. All other root sensors having reserved slaves for s wait for an acknowledgment from s until the expiration of a timeout; after the expiration of this timeout, they cancel the slave reservation.

If, on the contrary, the proposals received by s contain fewer than k sensors, s acknowledges all the proposals, but it continues flooding the invitation with an increased horizon $h + 1$ in order to find the sensors that are still necessary. This mechanism continues by increasing h several times until s finds the k needed sensors or h reaches the maximum network diameter.

Notice that the flooding of invitations is performed only through adjacent root sensors. As a result, if the root sensor s accepts a slave from a root sensor t, there is a traversable path from the responsibility region of t to the one of s, and vice versa.

It must be noted that some slave sensors may be engaged in other activities and may not be immediately available for a pull action, but they may become available at a later time. For example, an adjunct sensor being in an intermediate phase of regridding of a given responsibility region may not be available at the moment of the pull invitation, but may become available after a release. Additionally, a slave sensor may be temporarily reserved for another pull action if its root has already sent a slave proposal message but has not received an acknowledgment by the corresponding puller sensor. In both these cases, the root sensor that is receiving the new pull invitation stores such an invitation message in a dedicated queue. At the end of the Grid Refinement, if an adjunct sensor is released or a reserved slave sensor becomes newly available, and there are pull requests in the invitation queue, the root notifies the inviter of the new slave availability.

Furthermore, notice that a pull invitation may be launched while there is no full network connectivity yet. In this case, the message may only reach root sensors at the frontier of the connected component containing the puller root. These root sensors may be unable to propagate the invitation message because they do not have any adjacent root sensors to which they could forward the message. If such a premature pull activity occurs, the frontier root sensors store the invitation message in a forward queue and forward it as soon as they have new adjacent roots as potential recipients. At the same

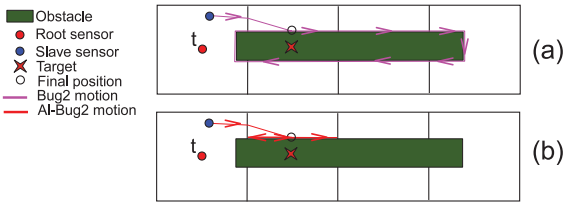


Fig. 4. Example of motion planning according to Bug2 (a) and AL-Bug2 (b).

time, they notify the inviter sensor so that it may reissue the pull invitation if still necessary. While root sensors execute a pull invitation, they must also store invitation messages received by other roots. This is necessary because a pulled sensor may reach the responsibility region of a puller root later than required, that is, when the vacant position has already been covered by another sensor.[4]

The described pull activity ensures that, if enough sensors are available to entirely cover the AoI at the required accuracy, each sensor performing the *Pull* activity will eventually receive all the sensors it needs.

4.4. Path Planning Under DOMINO: AL-Bug2

Mobile sensors, like any mobile robot, need a path planning algorithm to surround obstacles when heading to a destination. We designed the algorithm AL-Bug2 (Area Limited Bug2), a modified version of the Bug2 algorithm [Lumelsky and Stepanov 1987], which takes account of the specific needs of DOMINO.

According to Bug2, the robot heads toward the target along the segment connecting the two points. When the robot encounters an obstacle (*hitpoint*), it goes along the border of the obstacle (following a preferential direction, e.g., right-hand) until it can continue on its previous path from a point that is closer to the target than the hitpoint. In order to allow the robot to exit from maze-shaped obstacles, each time the robot reaches the same hitpoint, it changes the preferential direction with respect to the previous one (from right- to left-hand or the opposite).

While the original Bug2 is meant to guide the movement of a single robot from a source to a precise destination point, DOMINO requires the mobile sensors to move either to (1) a specific point (during a snap) or (2) a generic point of a responsibility region (during a pull or a push activity). Moreover, when the sensor has a specific destination point and it is obstructed by an obstacle, it may decide to change its destination depending on the shape of the responsibility region.

The algorithm Bug2 is designed to let robots circumnavigate obstacles until they reach a clear path to their destination or they realize that the destination is unreachable. For instance, consider Figure 4(a), where a slave sensor is sent to snap to an adjacent tile by the root sensor t. The target position is the center of the tile, but it is obstructed by the presence of an obstacle. Following the basic Bug2 algorithm, the sensor would traverse many tiles before being able to conclude that the desired position is unreachable and that it needs to stop in the closest point. This long path is necessary for Bug2 as it assumes that there is only one sensor and this sensor has to

[4]This may occur if two root sensors a and b compete for the same unique slave z. Let us assume that a wins the competition and z starts moving toward a. Assume also that in the meantime, the hole for which a issued the pull is filled by a third root sensor c. In the time that z reaches a, the pull action of b may terminate, having reached the maximum horizon, after contacting all possible roots. In order to let the slave z reach the region of the root sensor b, a must store the invitation message received by b and notify b of the availability of z as soon as its need of slaves is fulfilled by the third root sensor c.

reach a precise destination. Bug 2 moves the sensor on the opposite side of the obstacle to determine if the desired destination is reachable surrounding the obstacle.

Nevertheless, DOMINO does not require a sensor to completely circumnavigate large obstacles. When an obstacle is encountered during a snap action, the sensor determines the reachability of its destination within the target tile by moving along the obstacle until it reaches the tile border. If the destination is unreachable without exiting the tile, it means that the obstacle cuts the tile in two or more responsibility regions. Hence, once the sensor determines the shape of its responsibility region, it finds a new snap position within it and the destination point of the path planning algorithm is updated.

For this reason, in the same scenario of Figure 4(a), the new proposed path planning algorithm AL-Bug2 in (b) prevents sensors from uselessly circumnavigating large obstacles in the wrong direction, by making it more adherent to DOMINO requirements.

Furthermore, we recall that DOMINO may require a sensor to move to any point of an adjacent responsibility region. This may occur during a push or a pull action. In this case, the sensor is sent to the middle point of the corridor between the two tiles and stops as soon as it passes the border.

By contrast, when a sensor is pulled from a source tile to a destination tile, the two tiles may not be adjacent. Nevertheless, we recall that the flooding of pull invitations is performed only through adjacent tiles, so the pulled sensor can be informed of the entire list of tiles it has to traverse by analyzing the route followed by the received invitation message. Therefore, the movement can be decomposed in a list of shorter movements between adjacent tiles. Once the pulled sensor reaches a new tile, it updates the destination considering the next tile on the list.

In order to regulate the movement between adjacent responsibility regions, AL-Bug2 works as follows: when a sensor reaches the destination tile, if (1) the destination point is any point in the new tile, it simply stops there; instead, if (2) the destination point is a specific point in the new tile, the sensor considers its borders as new virtual obstacles and heads toward the destination treating real and virtual obstacles alike.

Let us consider a sensor s moving from region R_1 to an adjacent region R_2. The sensor s initially heads toward the corridor connecting the two regions, following the Bug2 algorithm. During this movement, s considers all the borders of R_1, except the corridor to R_2, as virtual obstacles. As a result, s can leave R_1 only through the corridor. When s enters R_2, it considers all the borders of R_2 as virtual obstacles and heads toward the target using Bug2, updating its destination when necessary.

5. ALGORITHM PROPERTIES

5.1. Bounds on the Number of Necessary Sensors

Let N_{DOMINO} be the number of sensors necessary to cover the AoI. N_{DOMINO} can be computed as the sum of several contributions: $N_{DOMINO} = N_{flat} + N_{asp} + N_{obs}$, where N_{flat} is the number of sensors needed to construct the skeleton grid, whereas N_{asp} and N_{obs} are the sensors placed according to the quad-tree refinement. N_{asp} is the number of additional sensors required to completely cover asperities or to fill the coverage gaps due to heterogeneity of sensing devices and N_{obs} is the number of additional sensors required to cover obstacles.

We give an upper bound N^{UB}_{DOMINO} on N_{DOMINO} by considering separate upper bounds on the number of sensors needed to create the skeleton grid (N^{UB}_{flat}), cover obstacles (N^{UB}_{obs}), and refine the grid (N^{UB}_{asp}).

Since N_{obs} highly depends on the size, mutual position, and orientation of obstacles over the AoI, it is impossible to give a general upper bound N^{UB}_{obs}.

In the following, we give some bounds on N_{flat} and N_{asp}. In the discussion, we assume that sensors have homogeneous sensing capabilities with sensing radius r_s. The result

can be trivially extended to the case of heterogeneous sensors by considering each of them equal to the one with lowest capabilities.

5.1.1. Bound on $N_{\mathtt{flat}}$. In order to compute N_{flat}, we consider a flat and smooth AoI. In this case, every root sensor lies in the center of a tile of the skeleton grid. The number of sensors composing the skeleton grid depends on the position and orientation of such a grid, and there is a boundary effect that must be considered.

In order to calculate an upper bound N_{flat}^{UB} on N_{flat}, we consider an AoI extended with a surrounding strip of width $\sqrt{2} \cdot l_s$, which is the diameter of a tile. We refer to such an extended AoI as $AoI^{+\sqrt{2}l_s}$. We use the notation $||AoI||$ to denote the size of the area of the region AoI. We use a similar notation $||AoI^{+\sqrt{2}l_s}||$ for the area of the region $AoI^{+\sqrt{2}l_s}$. We have

$$\lfloor ||AoI||/{l_s}^2 \rfloor \leq N_{flat} \leq N_{flat}^{UB} = \lceil ||AoI^{+\sqrt{2}l_s}||/{l_s}^2 \rceil.$$

5.1.2. Bound on $N_{\mathtt{asp}}$. We assume that the asperities can be modeled as n uniformly noisy areas: $Z_1, \ldots, Z_n$. Each asperity Z_i is characterized by a withering factor α_i, $0 < \alpha_i < 1$, which represents the reduction in the sensing range due to the asperity Z_i. In particular, the minimum sensing radius inside Z_i is $\alpha_i \cdot r_s$.

Let k_i^{last} be the number of recursive regridding steps that are necessary for a root sensor s to cover the intersection between the noisy area Z_i and its responsibility region. Sensor s recursively regrids its region until adjunct sensors are placed at a sufficient density to achieve full coverage. At each iteration of the regridding process, the diameter of the subtiles is half of the one of the previous step. As a result, at the jth iteration, the tile semidiagonal is given by $r_s/2^j$. The regridding stops at the iteration k_i^{last} such that $r_s/2^{k_i^{last}} \leq \alpha_i r_s$. Easy calculations lead to $k_i^{last} = \lceil -log_2 \alpha_i \rceil$. Coverage of the tile will be completed when the last level of adjunct sensors is deployed (at the k_i^{last}-th step of regridding). It must be noted that in the case described previously, where an asperity is the only reason for a root sensor to be incapable of covering its responsibility region, the last layer of regridding is sufficient to guarantee complete coverage. When such a layer is deployed, all the adjunct sensors placed at the previous regridding steps are redundant and are therefore released and made available for other activities. As a result, the number of adjunct sensors that remain in the tile after the release activity is $4^{k_i^{last}}$ and a tile inside an asperity Z_i is covered by means of a squared grid deployment with side $l_{Z_i} = \sqrt{2}\frac{r_s}{2^{k_i^{last}+1}}$.

The number of root sensors whose tile intersects Z_i can be upper bounded by using the method of the extended area we used for bounding N_{flat}. Each of these root sensors places at most $4^{k_i^{last}}$ additional sensors in its tile. As a result, the number of additional adjunct sensors $N_{asp}(Z_i)$ for the asperity Z_i is given by

$$N_{asp}(Z_i) \leq 4^{k_i^{last}} \left\lceil \frac{||Z_i^{+\sqrt{2}l_{Z_i}}||}{{l_s}^2} \right\rceil = N_{asp}^{UB}(Z_i) = 4^{k_i^{last}} T_i,$$

where T_i is the maximum number of responsibility regions intersecting the noisy area Z_i.

An upper bound N_{asp}^{UB} on N_{asp} is given by the sum of the contributions of all n noisy areas:

$$N_{asp} = \sum_{i=1}^{n} N_{asp}(Z_i) \leq \sum_{i=1}^{n} N_{asp}^{UB}(Z_i) = N_{asp}^{UB}.$$

5.2. Correctness and Termination of DOMINO

THEOREM 5.1 (TERMINATION). *The algorithm DOMINO terminates in a finite time.*

PROOF. Let $L = \{\ell_1, \ell_2, \ldots, \ell_{|L|}\}$ be the set of responsibility regions. We denote the status of the network with an $(|L| + 1)$-dimensional vector $\mathbf{s} =< \gamma, n_1, n_2, \ldots, n_{|L|} >$, where:

—γ denotes the uncovered portion of the AoI. Here coverage is calculated by considering the only coverage realized by root or adjunct sensors within their region (we do not consider as covered the portions of a tile that are covered by sensors located in adjacent tiles or by slaves).
—n_i represents the number of slaves in the ith responsibility region.

The execution of every algorithm activity leads to a change in the network status as follows:

—The Snap of a new root sensor in a responsibility region leads to a decrease of γ. Indeed, even when multiple Snap commands are issued toward the same responsibility region, a new region is covered, no matter which of the competing sensors actually becomes root.
—The Grid Refinement activity causes a decrease of γ in a finite time, after a limited number (up to $k_{\texttt{Max}}$ for the ith responsibility region) of recursive steps of regridding and a final release of redundant adjunct sensors.
—The Pull activity is always executed in a joint manner with a Snap or a Grid Refinement; therefore, it also terminates and results in a decrease of γ. As the Pull activity involves the forwarding of an invitation message through the network, its execution time is limited by the network size and by the depth of possible regridding, which may cause a delay in the release of the necessary sensor. In any of the previous cases, the status change occurs after a finite time.
—The Push activity is the only activity that leads to a change in the network status without an immediate decrease of γ. A push, in fact, implies a change in a pair of elements (n_i, n_j) of the status vector, as there is a movement of slaves from the ith to the jth responsibility region. Such a state change occurs in a finite time.

All the other algorithm activities (including intermediate forwarding of Pull messages and border/obstacle exploration) are only a complement to the execution of one of the activities listed earlier.

We define the function $f : \mathbb{R} \times \mathbb{N}^{|L|} \to \mathbb{R} \times \mathbb{N}^2$ as follows:

$$f(\mathbf{s}) = \left(\gamma, \sum_{i=1}^{|L|} n_i^2, \sum_{i=1}^{|L|} n_i \cdot id(\ell_i) \right).$$

We denote with $\succ$ the lexicographic order on the elements of the codomain of the function $f(\cdot)$ defined previously.

We now show that the function $f(\cdot)$ is monotonically decreasing at every change in the status of the network.

With the exception of the Push, every action (no matter if accompanied by a corresponding Pull) leads to an increment in coverage of the AoI, implying a decrease in γ, which also means a decrease in the value of the function $f(\cdot)$.

Let us consider the status change caused by a Push action. Let us consider a generic state change from s to s' that occurs when the root sensor s_x pushes a slave to the region of root sensor s_y. We have that $n_i = n_i' \quad \forall i \neq x, y$, $n_x' = n_x - 1$, and $n_y' = n_y + 1$. As the transfer of the slave has been performed according to the Moving Condition

(Equation (1)), either $n_x > n_y + 1$ or $(n_x = n_y + 1) \wedge (id(x) > id(y))$. In the first case, $n_x > n_y + 1$ implies that $\sum_{i=1}^{|L|} n_i^2 > \sum_{i=1}^{|L|} n'^2_i$. In the second case, since $n_x = n_y + 1$ and $id(x) > id(y)$, we have $\sum_{i=1}^{|L|} n_i^2 = \sum_{i=1}^{|L|} n'^2_i$ but $\sum_{i=1}^{|L|} n_i \cdot id(\ell_i) > \sum_{i=1}^{|L|} id(\ell_i) n'_i$.

Therefore, in both cases, $f(\mathbf{s}) \succ f(\mathbf{s}')$.

Notice that $f(\cdot)$ decreases in the first component γ only a finite number of times as a consequence of Snap actions (whose number is bounded above by $|L|$) or of Grid Refinement actions (whose number is bounded above by $|L| \cdot k_{\texttt{Max}}$) or of any other action composition that involves either a Snap or a regridding action. Furthermore, f decreases in the second and third components of discrete quantities (integer values) at any push action. As all these components are bounded below by 0 (f is lower bounded by $(0, 0, 0)$), the number of state changes is finite, and therefore DOMINO terminates in a finite time. □

We say that the AoI is *connected* if given any two points of the area that are not obstructed by obstacles, there is a traversable path connecting the two points.

THEOREM 5.2 (COVERAGE). *If at least N^{UB}_{DOMINO} sensors are available, and the AoI is connected, DOMINO performs a complete coverage of the AoI.*

PROOF. Let us consider an AoI with several obstacles and noisy zones.

The hypothesis of having N^{UB}_{DOMINO} sensors implies that the number of available sensors is at least equal to the number of root sensors necessary to govern each responsibility region, taking account of the obstacles over the AoI, plus the number of adjunct sensors necessary to cover the noisy zones at the necessary depth level.[5]

In order to prove that DOMINO is able to properly use such sensors, providing a complete coverage, recall that if a root sensor s detects a coverage hole in its own responsibility region or discovers a vacant root position in an adjacent tile, it either uses its available slaves to fill the hole or generates a Pull action, with the purpose to use a pulled sensor either for a new Grid Refinement or for a Snap action, respectively.

We need to prove that if no other root sensors govern the coverage of the mentioned hole, the forwarding of the Pull message sent by s eventually reaches a root sensor that has an available slave. This slave will move toward s and will be used by s either for a Snap or for a regrid action according to the coverage needs, thus filling the detected coverage hole.

We address the proof by splitting the previous assertion into two parts: Part (a) concerning the existence of a slave sensor to be pulled, and Part (b) concerning the reachability of such a sensor by means of a Pull invitation.

Part (a). Existence of a sensor that can eventually cover the hole: as long as there is a coverage hole, by hypothesis, there is a sensor in some region of the AoI in one of the following conditions: (a.1) it is an undeployed slave; (a.2) it is a pulled sensor, which will eventually become an unnecessary slave; or (a.3) it is an adjunct sensor, which will eventually be released.

In case (a.1), the Pull request will eventually reach this sensor, unless it is reserved by another root for a Pull or a Grid Refinement activity, leading to either case (a.2) or (a.3).

In case (a.2), even if the only available slaves in the network were already reserved due to Pull actions by other sensors, either these actions lead to an increase in coverage, freeing other pulled or adjunct sensors, or a pulled sensor moves toward its inviter root and eventually becomes an unnecessary slave. As the inviter root must have stored any

[5]We are implicitly assuming that coverage can be obtained in each responsibility region by means of a number of regridding actions that is lower than or equal to the maximum level $k_{\texttt{Max}}$ allowed by DOMINO to ensure termination.

Pull message received during its own Pull activity in its invitation queue, it notifies s of the new slave availability, so s can receive it if it is still necessary.

In case (a.3), thanks to the Cascaded Release mechanism, the quad-tree refinement activity does not create stale situations, as adjunct sensors at a given level k are always released in favor of the placement of other adjunct sensors at level $k+i$ with $i > 0$ and new sensors for level k are called back only if necessary for coverage. Once the regridding has led to the completion of the last level of refinement, all the Pull activities issued for the previous levels are interrupted and the received Pull invitations are honored if necessary as discussed for case (a.2).

Part (b). Reachability of t through a Pull action: Let t be the slave sensor that root sensor s needs to complete coverage, whose existence is proved in Part (a) of this proof. We now prove that, under the assumption that the AoI is connected, t is reachable by s through a Pull invitation. In case (a.1), t is an undeployed slave. Its root is eventually reached by a request from s thanks to the propagation mechanism of the Pull action and t is sent to s. If it is a sensor in situation (a.2) or (a.3), its root z is also reached by the Pull invitation and it stores the invitation message. In both cases, t will become available at a later time. At that time, z will notify s of the new sensor availability and subsequently will send t to s, which will use it to fill the coverage hole with either a Regrid or a Snap action.

We conclude the proof by observing that, while the root sensor s is still performing the Pull action to obtain the required slave, the coverage hole may be filled by newly arrived slaves (as a consequence of Push or Pull activities involving other root sensors). In such a case, the pulled sensor reaches s and becomes its slave, available to fulfill Pull requests received in the meantime or to participate in future algorithm activities. □

6. A VIRTUAL FORCE APPROACH FOR UNKNOWN FIELDS

As discussed in Section 3, there are no prior works dealing with the applicative scenario we consider in this article. For a thorough review of existing approaches, we refer the reader to Section 2.

We modified one of the best-performing virtual force-based algorithms in the literature, that is, the Parallel and Distributed Network Dynamics (PDND) algorithm [Ma et al. 2008]. PDND is one of the most complete solutions based on VFA currently available. In particular, unlike several previous proposals, it is formally proven that, under PDND, the sensors stop moving in a finite time without position oscillations that are typical of many VFA-based solutions. Furthermore, the algorithm shows very good performance in terms of coverage and uniformity of the final sensor distribution. For this reason, we extend PDND to deal with both position-dependent sensing capabilities and obstacles.

PDND is a round-based algorithm and sensors are synchronized. At each round, sensors broadcast their position and collect the information sent by neighbors. Then each sensor moves according to the virtual force acting on itself. The virtual force acting on a sensor is the vectorial sum of the forces exerted by its neighbors. In particular, given two sensors s_i and s_j at a distance d, the force exerted by s_i on s_j is modeled as a piecewise linear continuous function: it is repulsive if $d < r^*$, and attractive if $r^* < d < r_f$, where r^* and r_f are manually tuned parameters. The setting of these parameters is critical for the algorithm performance and is dependent on the value of the sensing radius r_s. In the case of sensors with homogeneous sensing radii, in order to obtain a deployment that is close to a hexagonal pattern with minimum overlap and no coverage holes, it is $r^* = \sqrt{3}r_s$ and $r_f = 2r_s$.

We introduce two major modifications to PDND in order to let it work with position-dependent sensing ranges and in the presence of obstacles. We set $r^* = \sqrt{3}(r_i + r_j)/2$,

where r_i and r_j are the position-dependent sensing radii of s_i and s_j, respectively. In the case of homogeneous sensors, this setting corresponds to a regular hexagonal pattern. Notice that PDND has a guaranteed termination only if the virtual force function is Lipschitz continuous. The presence of asperities implies a discontinuity in the force function. In order to ensure the termination of PDND in the presence of asperities, we introduce a centralized oscillation control method as in Heo and Varshney [2005]. In particular, we consider a sensor to be in an *oscillatory state* if the global movement of the previous m rounds is less than ϵ_m. We halt the algorithm execution as soon as all sensors are in an oscillatory state. We highlight that, although impractical, this centralized oscillation control constitutes an advantage for PDND; thus, our comparisons are fair. Furthermore, we apply such a centralized termination method only in scenarios characterized by the presence of asperities.

In order to let sensors deploy in the presence of obstacles, we adopt the same method proposed in Ma et al. [2008] to let sensors move according to the force component that is parallel to the obstacle borders. Moreover, in order to confine sensor movements within the AoI, we consider the borders of the AoI in the same way as obstacles.

7. EXPERIMENTAL RESULTS

In this section, we introduce some performance comparisons between DOMINO and PDND. In order to compare the performance of the two algorithms, we developed a simulator on the basis of the wireless module of the OPNET simulation environment [Riverbed - OPNET Technologies].[6]

In the experiments, we use the following parameter setting: homogeneous communication radius $r_{tx} = 15$m, homogeneous device sensing radius (locally altered by ground asperities in a position-dependent manner) $r_s = 5$m, sensor speed $v = 1$m/s, AoI size $80 \times 80\text{m}^2$, tile side of the skeleton grid $l_s = \sqrt{2}r_s$. Furthermore, we enable a delay in the regridding activity, as described in Section 4.3.2, by setting the parameter $\Delta = 1000$sec. We use the setting proposed in Ma et al. [2008] for PDND: round length $t_{round} = 1$sec and minimum movement threshold $l_{thr} = 0.1$m. In scenarios characterized by the presence of asperities, PDND does not ensure convergence; thus, we adopt the centralized termination method described in Section 6 by setting $m = 20$ and $\epsilon_m = 0.01$m. We use the following sensor energy consumption model: a sensor consumes energy as a consequence of movements (for traversing the desired distance and start/stop actions [Sibley et al. 2002]) and communications (sending, listening to, and receiving messages). We express the energy consumption of these actions in terms of energy units (eu). The reception of one message corresponds to $c_{rc} = 1$eu, a single transmission costs $c_{tx} = 1.2$eu, a 1-meter movement costs $c_{mv} = 340$eu, and a single start/stop action costs $c_{ss} = 340$eu [Anastasi et al. 2004].

We consider indoor and outdoor scenarios with and without asperities in order to study the performance of the two algorithms, and we consider an initial placement of the sensors in a small squared safe area A_{init} at the bottom left of the AoI, with size $10\text{m} \times 10\text{m}$.

Finally, we study the sensitivity of the two algorithms with respect to possible inaccuracies in estimation of obstacle location and sensing ranges.

7.1. Benchmark Scenario Without Asperities and Obstacles

Before showing more complex results, we first show that the modifications we made to PDND do not compromise the performance of the original algorithm in any aspect, but are only meant to extend its applicability to scenarios with ground asperities.

[6]The entire simulator code of both DOMINO and PDND can be downloaded from http://web.mst.edu/~silvestris/domino/.

Table I. Simulation Parameters

Parameter	Value	Description
r_{tx}	15m	Transmission radius (homogeneous)
r_s	5m	Sensing radius (homogeneous)
v	1m/sec	Device speed
l_s	$\sqrt{2}r_s$	Side of skeleton grid
AoI	$80 \times 80\text{m}^2$	Size of the AoI
A_{init}	$5 \times 5\text{m}^2$	Safe area where sensors are deployed initially
c_{rc}	1eu	Energy expenditure for receiving 1 message
c_{tx}	1.2eu	Energy expenditure for transmitting 1 message
c_{mv}	340eu	Energy cost of a movement of 1m
c_{ss}	340eu	Energy cost of a stop and start action
t_{round}	1sec	Duration of a single round (PDND)
l_{thr}	0.1m	Minimum movement threshold (PDND)
Δ	1,000sec	Regridding delay (DOMINO)
m	20	Maximum number of oscillatory movements (PDND)
ϵ_m	0.01m	Distance threshold for oscillation control (PDND)

We considered a smooth area $80 \times 80\text{m}^2$, with no obstacles or ground asperities. In such a scenario, the minimum number of sensors that guarantees coverage of the area according to the analysis of Section 5.1.1 is $N_{flat} = [80 + 2 \cdot (2r_s)]/(2 \cdot r_s^2) = 200$ sensors. When the number of sensors is 200 or higher, all the algorithms achieve complete coverage, namely, 100% of the AoI. Figure 6 shows the execution of both the original and the modified version of PDND by varying the number of sensors. In the interest of fairness, we ran the same experiments for DOMINO.

The figure shows that the original version of PDND, named "PDND-orig," performs exactly the same as the modified version, referred to as "PDND-mod." In fact, in a scenario without obstacles and ground asperities, the two variants of PDND perform the same movements and message transmissions. It can be seen from Figure 6(a) that PDND needs considerably more time than DOMINO to achieve the same coverage. Although PDND provides the same coverage with a slightly lower average movement distance, which is shown in Figure 6(b), it also requires a number of stop and restart actions that is orders of magnitude higher than DOMINO, as shown in Figure 6(c). This implies that both versions of PDND consume more than 10 times the energy consumed by DOMINO, as shown in Figure 6(d).

This analysis shows that the changes we provided to PDND extend its applicability to a wider range of operative scenarios, without harming its performance in the settings for which it is specifically designed. For this reason, in the following experiments, we use only the modified variant of PDND, which is hereafter called PDND for shortness of notation.

7.2. Experimental Scenario 1: Indoor

In this indoor scenario, the AoI contains walls, doors, and corridors. Sensors are initially deployed in a safe location, which is a room at the bottom-left corner of the area (see Figure 5(a)). Black dots are sensors, while gray circles are their sensing ranges. Figure 5(b) and Figure 5(c) show the final deployment achieved under DOMINO and PDND, respectively, when 300 sensors are initially deployed as shown in Figure 5(a). DOMINO successfully constructs the skeleton grid, which is refined by performing the regridding activity when root sensors do not entirely cover their responsibility regions. The Pull activity enables the flow of sensors toward uncovered areas even in such a complex environment. On the contrary, under PDND, sensors remain trapped in the first traversed rooms (they are stuck at high density along the walls). This is due to the fact that PDND aims at covering the AoI in an indirect manner, thanks to repulsive

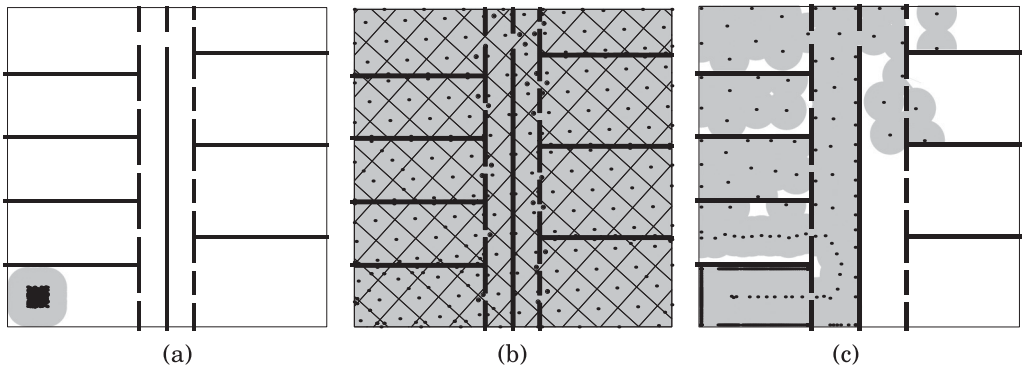


Fig. 5. Scenario 1: Indoor: an example of initial deployment with 300 sensors (a); final deployment under DOMINO (b) and PDND (c).

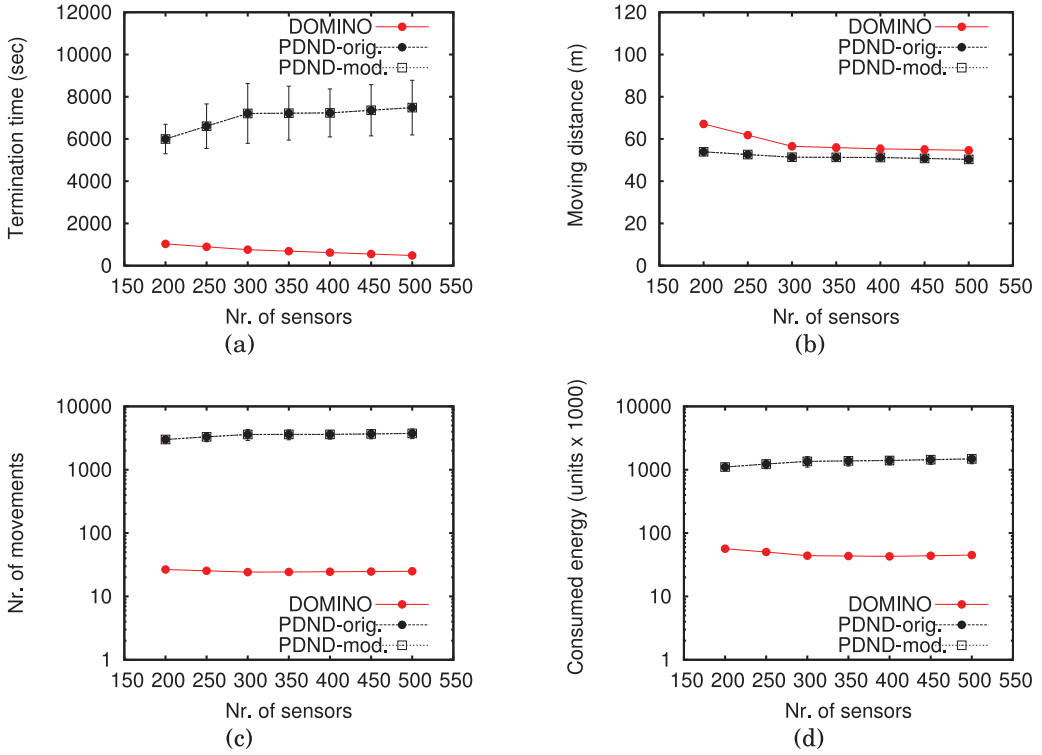


Fig. 6. Benchmark scenario: No asperities, no obstacles: Termination time (a), traversed distance (b), number of movements (c), consumed energy (d).

force exerted among sensors that are close to each other. Obstacles, corridors, and narrows create the condition for the equilibrium of virtual forces, as a considerable component of the force vectors is directed toward walls and obstacles. Therefore, the sensors are not able to spread and increase coverage of the AoI.

In the following experiments, we increased the number of deployed sensors from 100 to 450. Figure 7(a) shows the percentage of covered AoI achieved by DOMINO and PDND. DOMINO outperforms PDND as it achieves a higher coverage under all the

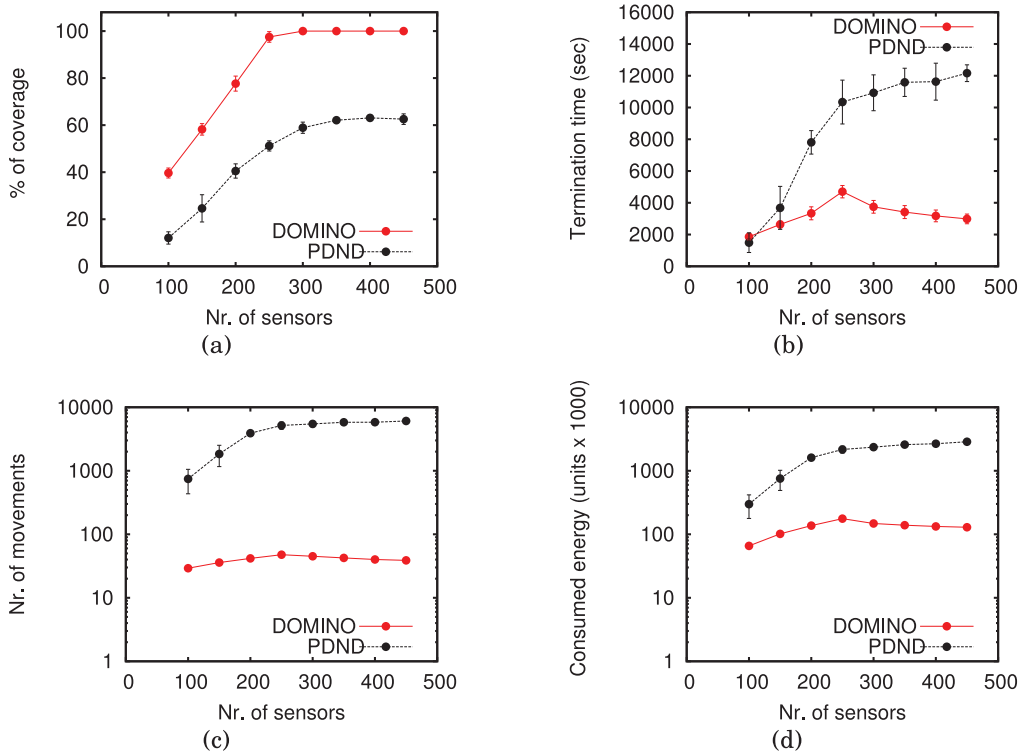
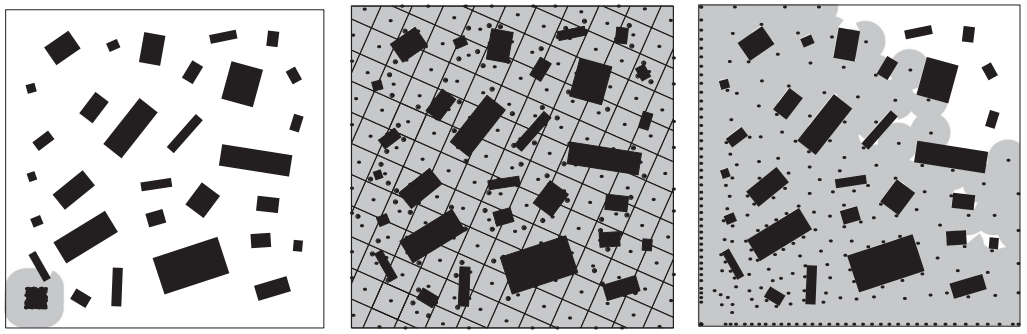


Fig. 7. Scenario 1: Indoor: Coverage (a), termination time (b), number of movements (c), consumed energy (d).

considered settings. The coverage of DOMINO linearly increases with the number of sensors and becomes complete with about 270 sensors. On the contrary, PDND is not able to spread the sensors over the AoI and it only achieves 60% of coverage even when the number of sensors is 50% higher than needed by DOMINO for full coverage.

Figure 7(b) compares the termination time of the two algorithms, that is, the time required to achieve the final deployment. PDND shows an increasing termination time with respect to the number of sensors, highlighting the difficulties of finding an equilibrium of the virtual forces when more sensors are deployed. By contrast, DOMINO enables a fast convergence to the final deployment, resulting in a much shorter deployment time than PDND. The peak of the termination time of DOMINO when the number of sensors is about 270 can be explained as follows. When the number of sensors is not sufficient for achieving full coverage, the push activity is not able to proactively spread the slaves over the area. Thus, the termination time is dominated by the pull activity performed to move sensors from the initial locations to the boundary of the covered zone. When the number of sensors is higher, the push activity is able to better spread the sensors over the AoI, reducing the time required by the Pull activity to attract sensors where needed.

The average number of start/stop actions is depicted in Figure 7(c). This is an important metric to evaluate the performance of mobile sensors' deployment algorithms since starting and stopping a movement consumes a significant amount of energy. PDND shows a number of moving actions that is two orders of magnitude higher than the one of DOMINO. The difficulty of finding an equilibrium of virtual forces causes a high number of small movements. DOMINO, instead, performs precise movements,



(a) (b) (c)

Fig. 8. Scenario 2: Outdoor: an example of initial deployment with 300 sensors (a) and final deployment under DOMINO (b) and PDND (c).

resulting in a much lower number of moving actions. It must be noted that PDND has such a higher number of movements than DOMINO, despite sensors' traversed distance being considerably lower, because PDND only achieves coverage of the 60% of the AoI. Figure 7(d) shows a cumulative energy consumption metric that includes the energy consumed for communications and movements. Although PDND lets sensors traverse shorter distances with respect to DOMINO, the higher communication cost due to its round-based nature, as well as the higher number of moving actions, dominates the resulting energy consumption.

7.3. Experimental Scenario 2: Outdoor

The second set of experiments considers an outdoor scenario with random obstacles over the AoI. In particular, we deployed 30 rectangular obstacles with random positions over the area. Obstacles have random rotation, and side length is uniformly distributed in the interval (0,20]. Sensors are initially deployed from a safe location similar to the one of the previous scenario. We increase the number of deployed sensors from 100 to 450.

Figure 8(a) shows an initial deployment with 250 sensors, while Figures 8(b) and 8(c) depict the final deployment achieved by DOMINO and PDND, respectively. DOMINO is able to successfully complete the coverage of the AoI. Thanks to the pull and regridding activities, root sensors are able to attract the required sensors and to fill the coverage holes created by the obstacle distribution. On the contrary, although PDND works better indoors than outdoors because sensors do not remain trapped in rooms and narrows, the algorithm is not able to efficiently use the available sensors and complete the coverage.

Figure 9(a) shows the coverage achieved by the two algorithms. PDND is able to achieve full coverage in this scenario, but the presence of random obstacles impedes the free flow of sensors, making it require a higher number of sensors with respect to DOMINO. As a result, DOMINO exploits the available sensors more efficiently and requires 44% fewer sensors than PDND to achieve full coverage. It must be noted that DOMINO requires fewer sensors in Scenario 2 than in Scenario 1 to achieve full coverage for two reasons: first, in Scenario 2, a large portion (more than 10%) of the AoI does not need to be covered as it is occupied by obstacles; second, while the walls of Scenario 1 shield portions of adjacent rooms that require more sensors to be covered, in Scenario 2, this shield effect is rare.

The average traversed distance and the average number of start/stop actions are depicted in Figures 9(b) and 9(c), respectively. Similarly to the previous scenario,

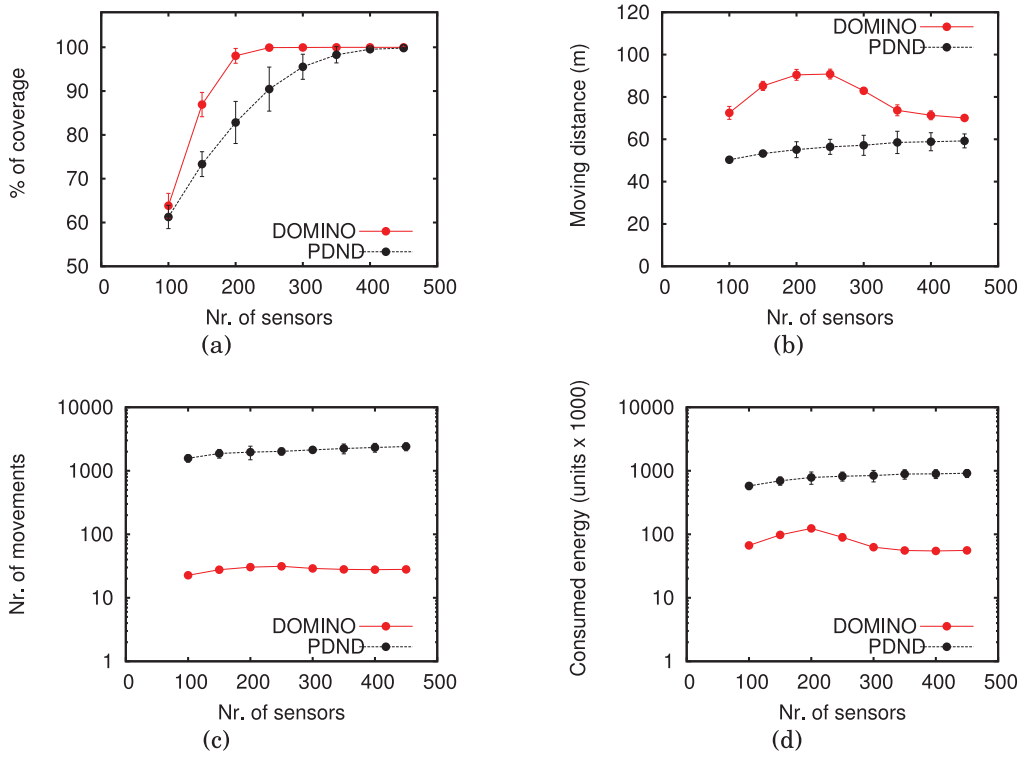


Fig. 9. Scenario 2: Outdoor: Coverage (a), traversed distance (b), number of movements (c), consumed energy (d).

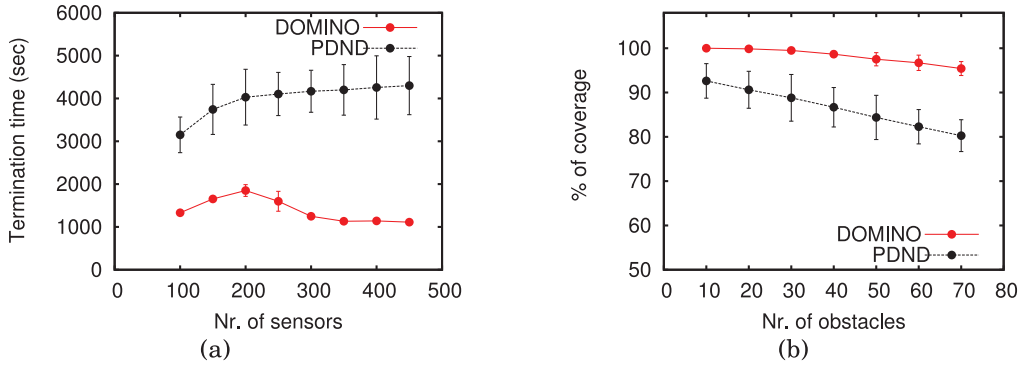
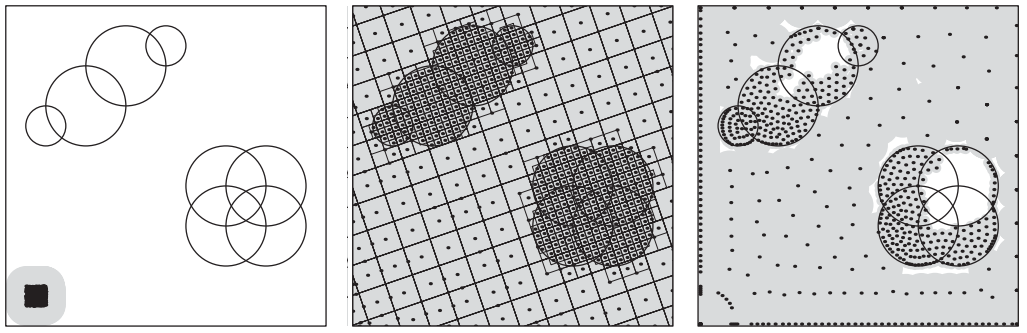


Fig. 10. Scenario 2: Outdoor: termination time (a), coverage with increasing number of obstacles (b).

DOMINO lets sensors traverse longer distances, but it achieves a higher coverage. Moreover, it requires two orders of magnitude fewer start/stop actions per sensor with respect to PDND. As a result, DOMINO requires an order of magnitude less energy with respect to PDND, as shown in Figure 9(d). Furthermore, the high number of small movements required by PDND to find an equilibrium of the virtual forces results in a longer termination time, as Figure 10(a) shows.

Notice that DOMINO terminates faster than in Scenario 1 due to the shorter network diameter. On the one hand, the push activity spreads sensors more effectively. On the



(a) (b) (c)

Fig. 11. Scenario 3: Asperities: An example of initial deployment with 700 sensors (a) and final deployment under DOMINO (b) and PDND (c).

other hand, the Pull activity is able to attract sensors by using a search with a smaller horizon.

The previously described results highlight the good scalability of DOMINO with respect to the number of sensors. The same cannot be said for PDND, where the availability of more sensors makes it harder to find an equilibrium of virtual forces.

In order to further study the performance of the two algorithms in complex outdoor environments, we performed some experiments by increasing the number of random obstacles and fixing the number of deployed sensors to 300. The coverage achieved by the two algorithms is shown in Figure 10(b). Both algorithms achieve a lower coverage as the number of obstacles increases. Nevertheless, DOMINO is less affected than PDND. The increase in the number of obstacles creates narrows, which precludes sensor movements under PDND as explained for the indoor scenario. On the contrary, DOMINO only has to perform a larger number of regridding actions to complete the coverage of each tile. The decrease in coverage under DOMINO is due to the lack of a sufficient number of sensors to fill all regridding positions.

7.4. Experimental Scenario 3: Asperities

In this scenario, we compare the performance of PDND and DOMINO in the presence of asperities that reduce the sensing capabilities. We consider an AoI with two zones Z_1, Z_2 with equal withering factor $\alpha = 0.3$. The sensors are initially deployed from a safe location at the left-bottom corner of the AoI (see Figure 11(a)). We increased the number of deployed sensors from 200 to 900.

Figures 11(b) and 11(c) depict an example of the final deployment achieved by 700 sensors under DOMINO and PDND, respectively. DOMINO is able to entirely cover the AoI with the available sensors. Notice that, under the considered setting, two levels of regridding are necessary to completely cover a region located inside an asperity. As a result, thanks to the release activity, each root sensor, whose responsibility region intersects an asperity, places at most 16 adjunct sensors. On the contrary, PDND does not efficiently use the available sensors and leaves the asperities uncovered.

Figures 12(a) and 12(b) show the percentage of coverage achieved by PDND and DOMINO over the entire AoI and over the asperities, respectively. DOMINO delays the regridding thanks to the parameter Δ, giving precedence to the construction of the top-level grid. As a result, when few sensors are available, DOMINO achieves a higher coverage of the AoI but a lower coverage of the asperities with respect to PDND. As the number of sensors increases, all the snap positions are filled and the remaining sensors can be used for regridding actions, and thus DOMINO achieves a higher coverage even

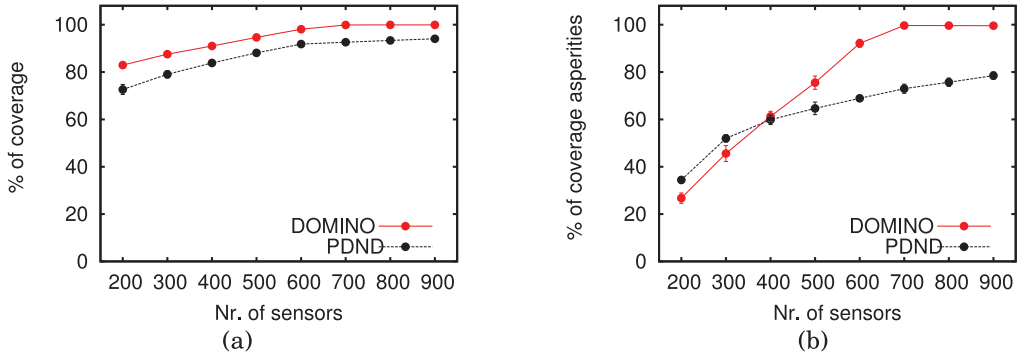


Fig. 12. Scenario 3: Asperities: Coverage of the AoI (a) and of the asperities (b).

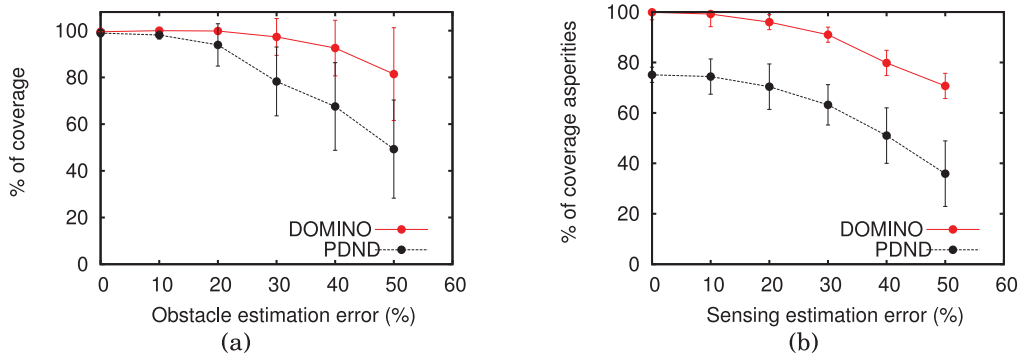


Fig. 13. Coverage of the AoI with inaccurate obstacle estimation (a); coverage of the asperities with inaccurate estimate of sensing range (b).

inside the asperities. PDND does not efficiently use the available sensors with respect to DOMINO. As a numerical example, when DOMINO achieves complete coverage of the asperities, PDND only covers 65%.

For the sake of brevity, we omit the results related to the traversed distance, number of movements, total energy consumption, and termination time, which are similar to what we described for the previous scenarios.

7.5. Sensitivity to Inaccuracies

In this section, we analyze the sensitivity of DOMINO and PDND with respect to inaccuracies that may occur in the estimation of obstacle locations and sensing ranges.

In a first scenario, sensors are deployed in an outdoor environment with randomly deployed obstacles, as in Section 7.3. Sensors may incur in a random error estimating the obstacle size. The error is expressed as a percentage of the actual value. We assume that the error may only result in an overestimation of the obstacle size, as underestimations can more easily be detected. In the experiments, we deploy 250 sensors and 20 random obstacles. The coverage of the AoI is shown in Figure 13(a). PDND is significantly affected by the estimation error because some narrows between obstacles may be perceived as obstructed, preventing sensors from flowing and covering the AoI. On the contrary, DOMINO is more robust since sensors can circumnavigate perceived obstructions by using different paths.

In the second scenario, we consider an AoI with two asperities, as in Section 7.4. We assume that sensors erroneously estimate the withering factor α. We express the error

as a percentage of the actual factor. In the experiments, we deploy 800 sensors. The resulting coverage of the asperities is shown in Figure 13(b). DOMINO is more robust than PDND. The presence of errors worsens the performance of PDND, preventing sensors from covering the asperities. On the contrary, the quadtree-based deployment of DOMINO reduces the impact of the erroneous estimate.

8. CONCLUSIONS

In this article, we introduced DOMINO, a deployment algorithm for mobile sensors designed to work in unknown environments characterized by the presence of obstacles and noise sources that make sensor capabilities anisotropic. According to DOMINO, mobile sensors construct a top-level grid following a regular pattern. Such a grid is autonomously distorted by the mobile sensors and adapted to the environment during the deployment phase. Furthermore, additional sensors are placed if the sole top-level grid does not meet coverage requirements.

We formally prove that DOMINO achieves full coverage and terminates in a finite time if enough sensors are available. Furthermore, we provide some bounds on the minimum number of sensors necessary to achieve full coverage. We compare DOMINO to a modified version of the PDND algorithm. DOMINO outperforms PDND both in indoor and outdoor scenarios, achieving full coverage with fewer sensors, in a shorter time and with a lower energy consumption.

REFERENCES

Giuseppe Anastasi, Marco Conti, Alessio Falchi, Enrico Gregori, and Andrea Passarella. 2004. Performance measurements of mote sensor networks. In *ACM Proceedings of MSWiM 2004*, 174–181.

Amotz Bar-Noy, Theodore Brown, Matthew P. Johnson, and Ou Liu. 2009. Cheap or flexible sensor coverage. *IEEE Proceedings of DCOSS, Lecture Notes in Computer Science* 5516 (2009), 245–258.

Novella Bartolini, Giancarlo Bongiovanni, Tom La Porta, and Simone Silvestri. 2014a. On the vulnerabilities of the virtual force approach to mobile sensor deployment. *IEEE Transactions on Mobile Computing (TMC)* 13, 11, (2014), 2592–2605.

Novella Bartolini, Giancarlo Bongiovanni, Tom La Porta, Simone Silvestri, and Francesco Vincenti. 2014b. Voronoi-based deployment of mobile sensors in the face of adversaries. *IEEE Proceedings of the International Conference on Communications (ICC'14)*.

Novella Bartolini, Tiziana Calamoneri, Annalisa Massini, and Simone Silvestri. 2011a. On adaptive density deployment to mitigate the sink-hole problem in mobile sensor networks. *Springer Mobile Networks and Applications* 16, 1 (2011), 134–145.

Novella Bartolini, Tiziana Calamoneri, Thomas F. La Porta, and Simone Silvestri. 2011b. Autonomous deployment of heterogeneous mobile sensors. *IEEE Transactions on Mobile Computing (TMC)* 10, 6 (2011), 753–766.

Tomas Beran, Richard B. Langley, Sunil B. Bisnath, and Luis Serrano. 2007. High-accuracy point positioning with low-cost GPS receivers. *Navigation* 54, 1 (2007), 53–63.

Nirupama Bulusu, John Heidemann, and Deborah Estrin. 2000. Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine* 7, 5 (October 2000), 28–34.

Jiming Chen, Shijian Li, and Youxian Sun. 2007. Novel deployment schemes for mobile sensor networks. *Sensors* 7 (2007), 2907–2919.

Yoann Dieudonné, Ouiddad Labbani-Igbida, and Franck Petit. 2008. Circle formation of weak mobile robots. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 3, 4, Article 16 (Dec. 2008).

Adriano Fagiolini, Antonio Bicchi Lisa Tani, and Gianluca Dini. 2008. Decentralized deployment of mobile sensors for optimal connected sensing coverage. In *IEEE Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS'08).* 486–491.

Rafael Falcon, Xu Li, and Amiya Nayak. 2011. Carrier-based focused coverage formation in wireless sensor and robot networks. *IEEE Transactions on Automatic Control (TAC)* 56, 10 (2011), 2406–2417.

Michele Garetto, Marco Gribaudo, Carla-Fabiana Chiasserini, and Emilio Leonardi. 2007. A distributed sensor relocation scheme for environmental control. *International Conference on Mobile Ad Hoc and Sensor Systems (MASS'07)*.

Seth Gilbert, Nancy Lynch, Sayan Mitra, and Tina Nolte. 2009. Self-stabilizing robot formations over unreliable networks. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 4, 3, Article 17 (July 2009).

Hongliang Guo, Yaochu Jin, and Yan Meng. 2012. A morphogenetic framework for self-organized multirobot pattern formation and boundary coverage. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 7, 1, Article 15 (May 2012).

Azwirman Gusrialdi, Sandra Hirche, David Asikin, Takeshi Hatanaka, and Masayuki Fujita. 2009. Voronoi-based coverage control with anisotropic sensors and experimental case study. *Intelligent Service Robotics* 2 (2009), 195–204.

Nojeong Heo and Pramod K. Varshney. 2005. Energy-efficient deployment of intelligent mobile sensor networks. *IEEE Transactions on Systems, Man and Cybernetics* 35 (2005), 78–92.

Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. 2002. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems (DARS'02)*.

Khoufi Ines, Minet Pascale, Laouiti Anis, and Livolant Erwan. 2014. A simple method for the deployment of wireless sensors to ensure full coverage of an irregular area with obstacles. In *Proceedings of the ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MsWIM'14)*. 203–210.

Matthew B. Johnson, Deniz Sariöz, Amotz Bar-Noy, Theodore Brown, Dinesh Verma, and Chai Wah Wu. 2009b. More is more: The benefits of denser sensor deployment. In *IEEE Proceedings of the International Conference on Computer Communications (INFOCOM'09)*

Wesley Kerr, Diana F. Spears, William M. Spears, and David R. Thayer. 2004. Two formal fluid models for multi-agent sweeping and obstacle avoidance. In *Proceedings of the Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*.

Xu Li, Greg Fletcher, Amiya Nayak, and Ivan Stojmenovic. 2014. Placing sensors for area coverage in a complex environment by a team of robots. *ACM Transactions on Sensor Networks (TOSN)* 11, 1, Article 3 (August 2014).

Vladimir J. Lumelsky and Alexander A. Stepanov. 1987. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica* 2 (1987), 403–430.

Ke Ma, Yanyong Zhang, and Wade Trappe. 2008. Managing the mobility of a mobile sensor network using network dynamics. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 19, 1 (2008), 106–120.

Ming Ma and Yuanyuan Yang. 2007. Adaptive triangular deployment algorithm for unattended mobile sensor networks. *IEEE Transactions on Computers (TOC)* 56, 7 (2007), 946–958.

Sonia Martinez, Jorge Cortes, and Francesco Bullo. 2007. Motion coordination with distributed information. *IEEE Control Systems Magazine* (August 2007), 75–88.

Memsic MTS420/400 Datasheet. 2017. https://www.memsic.com/userfiles/files/Datasheets/WSN/mts400_420_datasheet-t.pdf, last accessed on May 2017.

Akihisa Ohya, Takayuki Ohno, and Shin'ichi Yuta. 1996. Obstacle detectability of ultrasonic ranging system and sonar map understanding. *Robotics Autonomous System* 18, 1 (1996), 251–257.

Celal Ozturk, Dervis Karaboga, and Beyza Gorkemli. 2011. Probabilistic dynamic deployment of wireless sensor networks by artificial bee colony algorithm. *Sensors* 11 (2011), 6056–6065.

Muhammed R. Pac, Aydan M. Erkmen, and Ismet Erkmen. 2006. Scalable self-deployment of mobile sensor networks: A fluid dynamics approach. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06)*.

Neal Patwari, Joshua N. Ash, Spyros Kyperountas, Alfred O. Hero, Randolph L. Moses, and Neiyer S. Correal. 2005. Locating the nodes: Cooperative localization in wireless sensor networks. *IEEE Signal Processing Magazine* 22, 4 (2005), 54–69.

Riverbed - OPNET Technologies. Retrieved from. http://www.riverbed.com/products/performance-management-control/.

Gabriel T. Sibley, Mohammad H. Rahimi, and Gaurav S. Sukhatme. 2002. Mobile robot platform for large-scale sensor networks. In *IEEE Proceedings of the International Conference on Robotics and Automation (ICRA'02)*.

Guang Tan, Stephen A. Jarvis, and Anne-Marie Kermarrec. 2009a. Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks. *IEEE Transactions on Mobile Computing (TMC)* 8, 6 (2009), 836–848.

Muhammad Tariq, Zhenyu Zhou, Yong-Jin Park, and Takuro Sato. 2010. Diffusion based self-deployment algorithm for mobile sensor networks. In *IEEE Proceedings of the Vehicular Technology Conference (VTC'10)*.

Zhiliang Tua, Qiang Wang, Hairong Qi, and Yi Shena. 2012. Flocking based distributed self-deployment algorithms in mobile sensor networks. *Journal of Parallel and Distributed Computing* 72, 3 (2012), 437–449.

Guiling Wang, Guohong Cao, and Thomas La Porta. 2006. Movement-assisted sensor deployment. *IEEE Transactions on Mobile Computing (TMC)* 6 (2006), 640–652.

Received October 2015; revised June 2016; accepted January 2017