Mobile sensor deployment in unknown fields

Novella Bartolini, Tiziana Calamoneri
University of Rome "Sapienza"
Rome, Italy
Email: {bartolini,calamo}@di.uniroma1.it

Thomas La Porta
Pennsylvania State University
University Park, PA, USA
Email:tlp@cse.psu.edu

Simone Silvestri
University of Rome "Sapienza"
Rome, Italy
Email: silvestris@di.uniroma1.it

***Abstract*—In this paper we propose GREASE, a distributed algorithm to deploy mobile sensors in an unknown environment with obstacles and field asperities that may cause sensing anisotropies and non uniform device capabilities. These aspects are not taken into account by traditional approaches to the problem of mobile sensor self-deployment. GREASE works by realizing a grid-shaped deployment throughout the Area of Interest (AoI) and adaptively refining the grid to find new sensor positions to cover the target area more precisely in the zones where devices experience reduced movement, sensing and communication capabilities. We give bounds on the number of sensors necessary to cover an AoI with obstacles and noisy zones.**

Simulations show that GREASE provides a fast deployment with precise movements and no oscillations, with moderate energy consumption.

I. Introduction

Mobile sensors are of great importance for monitoring a field that is inaccessible, unfamiliar or even hostile, where sensors are deployed from a distance, e.g. from a safe location or from an aircraft, and then reposition themselves to provide the required sensing coverage. A sensor deployment algorithm is necessary to automate the positioning phase.

In this paper we focus on the practical and realistic problem of deploying sensors over a real target field whose local characteristics become known only during the positioning phase. In particular, we consider target fields where noise or ground asperities significantly affect the sensing, communication and movement capabilities of devices. We take into account possible position dependent anisotropies both in the sensing and communication capabilities as well as positioning errors. In these cases a deployment algorithm should adapt the sensor positions to the terrain and should create a denser deployment nearby obstacles and noisy zones.

Various approaches have been proposed to self-deploy mobile sensors. They are based either on the virtual force model [1], [2], [3], [4] or on computational geometry models [5], [6], [7], with few exceptions in which the proposed algorithms aim at deploying sensors in geometric patterns [8], [9].

We propose a distributed deployment algorithm, named GREASE, that performs a recursive local refinement of the sensor positions inside the tiles of a regular grid. In this paper we give the following contributions: (i) we define, for the first time, an algorithm which requires no a priori knowledge to automatically and effectively self-deploy sensors in environments with obstacles and noisy areas in which sensing ranges are reduced; (ii) we show that GREASE terminates even in settings with obstacles or noisy areas, unlike other algorithms that rely on virtual forces; (iii) we show that GREASE provides full coverage if the number of available sensors exceeds a given threshold; and (iv) we show through extensive simulation the operational benefits of this algorithm with respect to others in the literature.

II. The mobile sensor deployment algorithm

We assume that devices can be heterogeneous and that, in running the algorithm, the sensor communication and sensing capabilities may be reduced and become anisotropic due to the presence of noise, ground asperities and obstacles.

We also assume that each sensor knows the coordinates of the AoI and can determine its own location (e.g. using low cost GPS). Sensors may move at variable speed and position themselves with some error around the target position. The algorithm does not require any device synchronization.

GREASE is based on the use of grids to position sensors. The algorithm starts by positioning sensors into a *top level grid*. In order to define the size of this grid, we introduce the following notation: r_s is the maximum sensing radius among the available devices. The side of the root grid tile l_s is set such that $l_s \leq \sqrt{2}r_s$. In this way, the root grid alone is sufficient to guarantee the coverage completeness when enough equally equipped sensors are provided and the AoI has no obstacles or impairments which affect the sensors. In order to accommodate small errors in sensor placement, we reduce the size of the grid with respect to its maximum possible value, as discussed in [10]. In particular, we set $l_s = \sqrt{2}r_s - 2\sigma$, where σ is the positioning error that can be tolerated without loss of coverage.

The top-level grid can be formed by means of any grid oriented deployment algorithm for mobile sensors. In this paper we adopt a modified version of Push & Pull [9]. We refer to *tile roots* as the sensors located in the top level grid positions. Wherever nodes located in adjacent grid points are not connected with each other, the algorithm provides the deployment of *stripes* of sensor, as proposed in [8], to guarantee the device connectivity. Due to space limitation we do not give more details on our use of stripes. If coverage holes (due to imprecise sensor positioning, obstacles or ground asperities) are detected, the tile roots recursively partition the region according to a grid of smaller dimension in a process called *re-gridding*, detailed in Algorithm 1. the regridding follows a *quadtree structure*, namely, each tile T

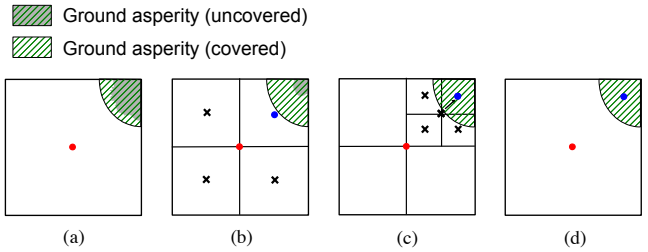


Fig. 1. Construction of a finer grid over a tile, to complete the coverage.

that is not completely covered, is divided into 4 *sub-tiles* $T[i]$, $i = 1, \ldots, 4$, whose coverage is evaluated separately and if a coverage hole is detected in a subtile, additional sensors are invited (among those that have not been positioned yet, called *movable sensors*), to be positioned in the uncovered or not completely covered subtiles.

The algorithm provides two different modalities of execution for a positioned sensor x, whether x is a tile root (lines 3-5) or a subtile leader (lines 6-23). The two different modalities differ in a small detail that has a significant impact on the performance of the algorithm. In order not to incur an excessive growth in the number of recursively placed sensors, the algorithm provides a *self-release* technique which is executed by subtile leaders only. According to this technique, whenever a sensor x, leader of the subtile T_x has only one subsubtile $T_x[i]$ to be covered, it tries to move itself to the center of the subsubtile $T_x[i]$. If this movement leaves new coverage holes in the subtile T_x, x invites a new movable p to its previous position at the center of the subtile T_x.

The re-gridding action goes recursively on dividing subtiles into subsubtiles and so forth. The process terminates as soon as either the AoI is completely covered or the available sensors have all been positioned.

In order to accelerate the termination and to achieve a more balanced deployment, we propose two alternative approaches: (1) we limit the size of the total uncovered area in each root tile to ϵ or (2) we limit the number of recursive re-gridding to a maximum level $k_{\texttt{Max}}$.

An example of the re-gridding technique is shown in Figure 1, where a tile-root detects a coverage hole due to ground asperities (a), hence it executes a re-gridding (b) and places only one sensor in the top-right sub-tile, as the other three positions refer to sub-tiles completely covered by the tile-root. In (c) the previously added sensor performs another re-gridding action that requires its movement to the top-right tile, completing the coverage of the root grid tile. The final deployment on the considered tile is shown in (d).

A. Dealing with obstacles

In a real environment the AoI contains different types of obstacles which may constitute an impediment to movements, sensing and transmission operations of the sensor devices, to a different extent. GREASE adopts the Lumelsky and Stepanov's path planning algorithm [11] called Bug2 to regulate the sensor movements in the presence of obstacles. Each root sensor adopts two techniques for covering the points of its

Algorithm 1 Multi-resolution grid deployment executed by a tile-root or sub-tile leader x on its partially uncovered sub-tile

```
1:  MULTI_RESOLUTION(Tile T)
2:  c ← number of uncovered sub-tiles, c> 0
3:  if x is a tile-root then
4:      for i = 1 . . . c do
5:          invite a movable and fix it in T[i]
6:  else
7:      //regridding with self-release attempt
8:      for i = 1 . . . c − 1 do
9:          invite a movable p and fix it in T[i]
10:         listen to self-release permission/denial from p
11:     if ∃ a self-release denial then
12:         invite a movable and fix it in T[c]
13:     else
14:         move to T[c] //self-release
15:         if the movement generates new holes in T and it enhances
            the coverage of some zones of T[c] then
16:             fix in T[c]
17:             invite a movable p and fix it in T
18:         if the movement generates new holes in T and does not
            enhance the coverage of T[c] then
19:             go back and fix in T
20:         if the movement does not generate new holes and improves
            the coverage of T[c] then
21:             fix in T[c]
22:         if T[c] is not completely covered then
23:             MULTI_RESOLUTION(T[c])
```

tiles that are occluded by an obstacle. Indeed, the placement of new sensors within a tile is performed either by following the re-gridding technique, as in the case of ground asperities and noise in the tile, or by selecting some focal points of the uncovered region as destinations.

Regridding is performed whenever a sensor detects more than one obstacle in its tile, in order to reach a configuration in which each sensor has only one obstacle in its tile/subtile. When a sensor has only one obstacle in its tile, it first tries to surround the obstacle by placing additional sensors in some focal points of the uncovered region, and then tries to refine the achieved coverage with the regridding technique, if it is still necessary.

The use of stripes is provided also in this case, to address the problem of connectivity between the additional nodes and the tile root.

More in detail, if a tile root or a sub-tile leader perceives more than one obstacle in its tile, it performs recursive re-gridding actions (lines 22-23). If, instead, it perceives only one obstacle inside its tile (lines 2-21), it selects/invites a movable sensor p (line 3) to be placed in the miniMax point mM (line 4) of the uncovered area (i.e. the point that minimizes the maximum distance from the boundaries of the hole). Three cases may occur: mM is uncovered and is reachable (lines 5-6), mM is uncovered and is unreachable (lines 5-12), mM is already covered (lines 13-18). Whether the placement of the additional sensor is not possible or not sufficient to completely cover the hole, the re-gridding phase (lines 19-21) is also executed.

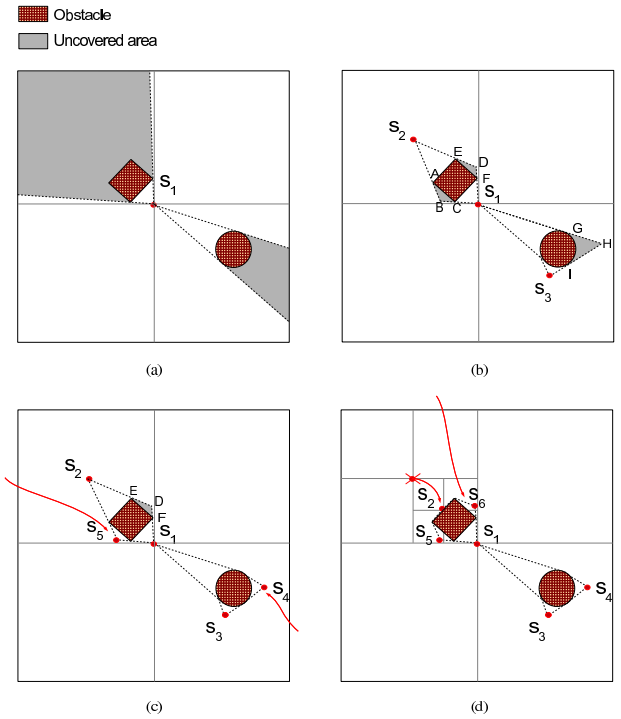


Fig. 2. Execution of GREASE in a tile with multiple obstacles

Notice that a sensor placed according to the miniMax rule can become redundant after a re-gridding action. In this case such a sensor is released.

Algorithm 2 Handling of obstacles executed by tile-root or sub-tile leader x

```
OBSTACLES(Sensor x)
if a single obstacle is detected then
    invite a movable p
    Let A be the uncovered area and mM ← minimax(A)
    if mM is uncovered then
        move p to mM
        if mM is unreachable then
            //i.e. p went round the obstacle
            if the obstacle covers the remainder of the tile then
                uninvited p and EXIT
            else
                move p to the farthest point of A wrt the obstacle
    else
        move p to explore the obstacle
        if the obstacle covers the remainder of the tile then
            uninvite p and EXIT
        else
            move p to the farthest point of A wrt the obstacle
    check coverage of tile of x
    if coverage is not complete then
        MULTI_RESOLUTION(tile of x)
else
    MULTI_RESOLUTION(tile of x)
```

In Figure 2 we show an example of the algorithm execution in a tile that contains two obstacles. Figure (a) shows that sensor s_1 detects two obstacles in its tile. Hence, s_1 re-grids its tile according to the quadtree technique, and places the sensors s_2 and s_3 at the centers of its uncovered subtiles, as shown in (b). The sensor s_2 detects a unique uncovered region ABCFDE (it still does not know the exact shape of the obstacle), as well as s_3 detects the uncovered region GHI. Figure (c) shows how s_3 places a new sensor s_4 in the miniMax point of the region GHI, which is thus completely covered. The sensor s_2 instead tries, without success, to place s_5 in the miniMax of the uncovered area detected in its sub-tile. Such a point is occupied by the obstacle. For this reason s_2 places the sensor s_5 in B, that is the point of the uncovered region that is the farthest from the obstacle. The sensor s_5 is not sufficient to cover the tile of s_2 completely. Hence s_2 performs a re-gridding action, as shown in (d). According to the stale avoidance mechanism, s_2 moves to the only sub-tile that necessitates the presence of a leader. Unfortunately such a movement does not enhance the coverage of the tile. According to GREASE, s_2 proceeds recursively, and invites s_6 to move to the miniMax position of the triangle EDF, thus completing the coverage of the obstacle.

III. Algorithm properties

We now state several properties of the GREASE algorithm, although we do not show their proofs due to the lack of space.

Theorem III.1. *(Termination) The algorithm GREASE always terminates, regardless of the size and shape of the AoI, and the presence of obstacles and ground asperities.*

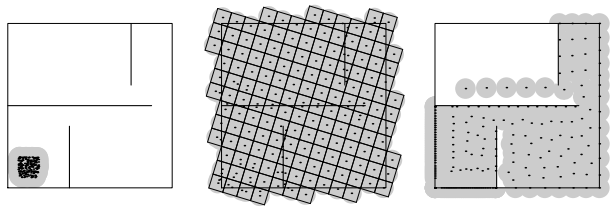
If the ground is obstacle-free and the device sensing capabilities are uniform, the maximum number of sensors that GREASE needs to cover the AoI entirely, with a grid of size l_s, regardless of the position and orientation of the grid, is called $N_{\mathtt{t}}(AoI, l_s)$ (sufficient number), where the minimum of this value is $N_{\mathtt{opt}}(AoI, l_s)$ (necessary number).

Theorem III.2. *(Complete coverage) Under the assumption of having a number of equally equipped sensors greater than or equal to $N_{\mathtt{t}}(AoI, l_s)$, GREASE guarantees a complete coverage of the AoI, even in the presence of ground asperities.*

We now show that $N_{\mathtt{t}}(AoI, l_s)$ is not too large with respect to $N_{\mathtt{opt}}(AoI, l_s)$.

Let the *effective ratio* of our algorithm be the ratio of $N_{\mathtt{t}}(AoI, l_s)$ over $N_{\mathtt{opt}}(AoI, l_s)$. This ratio heavily depends on the shape of the AoI and the length of its perimeter, and hence it cannot be evaluated in general. Nevertheless, in the special case of a rectangular AoI of length L and width W, with no obstacles and ground asperities, it is upper bounded by $1+\eta$ where $\eta = \frac{2r_s^2}{LW} + \frac{4r_s}{LW}(L+W+4r_s)$ is a small constant, provided that the sides of the AoI are sufficiently larger than r_s. As an example, if W and L are both greater than or equal to $80r_s$, then η is less than about 0.1. As a numerical example if the grid is dimensioned so that the lower bound $N_{\mathtt{opt}}(AoI, l_s)$ is 100, the upper bound $N_{\mathtt{t}}(AoI, l_s)$ is less than 110.

The same properties hold in the presence of noise, obstacles and ground asperities, provided that some knowledge on the AoI is available to calculate the proper value of $N_{\mathtt{t}}(AoI, l_s)$, that in this case depends heavily on the number, extension and shape of obstacles and ground asperities, and on the intensity



(a) (b) (c)

Fig. 3. Initial random deployment in the presence of obstacles (a); Deployment after the execution of GREASE (b); Deployment after the execution of PDND (c)

of the noise factors. As an example, we are able to calculate the value of $N_{\mathtt{t}}(AoI, l_s)$ and the effective ratio in the case of a rectangular AoI of dimension $L \times W$ in the presence of m segment shaped obstacles, each one of length L_i, $i = 1, \ldots, m$. In this case, the effective ratio of our algorithm is bounded by $1 + \eta + l_s^2((\sqrt{2}-1)\delta' + (\sqrt{2}+3)\delta)$. where δ is the *density* of the segments in the AoI, i.e the ratio of m over the measure of the area of the AoI, and δ' is the *normalized density*, i.e. the ratio of $\sum_{i=1}^{m} \lfloor L_i/l_s \rfloor$ over the measure of the area of the AoI. As an example, in a square AoI of side length 80 m, $l_s = 1$m, with 20 segment shaped obstacles, each one 10 m long, it turns out that η is about 0.1 and this effective ratio is less than 1.12.

IV. Experimental results

In order to make a performance evaluation of GREASE we modified the algorithm called Parallel and Distributed Network Dynamics (PDND), proposed in [3], to deal with both position dependent sensing capabilities and obstacles, as we did in [7].

We observe that in the presence of ground asperities, the convergence of PDND is not guaranteed. Therefore, we adopt the centralized oscillation control proposed in [2]. We highlight that, although impractical, this oscillation control constitutes an advantage for PDND and, for this reason, our comparisons are still fair.

We consider an AoI of 80 m $\times$ 80 m and use the following device parameter settings: $r_{\mathtt{tx}} = 15$ m, $r_{\mathtt{s}} = 5$ m, sensor speed $v = 1$ m/sec. We neglect possible positioning errors, hence we set the GREASE root grid size at $5\sqrt{2}$ m. For the PDND algorithm, the round length is set to 1 sec while the minimum moving distance is set to 0.1 m, as in [3].

Experiments with obstacles

In the next figures we analyze the deployment achieved under the execution of GREASE and PDND over an AoI that contains three segment shaped obstacles, as shown in Figure 3(a). The obstacles impede the device sensing and movements. 200 devices are spread over the AoI starting from a dense region and become aware of the presence of the obstacles only when they fall within their sensing range. Figures (b) and (c) show the final deployment achieved under GREASE and PDND, respectively. Unlike GREASE, PDND is not able to complete the coverage of the AoI. Furthermore, notice the occurrence with PDND of a *sticky border effect*, common to many virtual force based approaches. This is due to the fact that sensors that have been pushed towards the border are not called back unless there is a low density nearby. Other virtual force based approaches address this problem by modeling a repulsive force from the obstacles [1], [4]. We did not use these models as repulsive obstacles impede the passage of sensors through narrows and openings.

In order to clarify the different behavior of the two algorithms, we also consider a variant of PDND, which we refer to as PDND-STOPPED, that is the algorithm PDND is forcedly terminated at the termination time of GREASE if it is not already terminated by itself.

In order to compare the performance of these algorithms we increase the number of deployed sensors from 100 to 350. The results are obtained by averaging over 30 simulation runs. Figure 4(a) shows the completion time of the two algorithms. As PDND always has completion times one order of magnitude longer than GREASE, the two lines of GREASE and PDND-STOPPED coincide. The authors of PDND introduce a limitation to the distance each sensor is allowed to traverse at each round in order to ensure that the force acting on a sensor actually decreases at each round. As a consequence of this limitation, PDND is particularly slow in achieving its final deployment. On the contrary, GREASE lets sensors traverse entire grid tiles at each movement, thus resulting in a short termination time and very regular and precise movements.

Figure 4(b) illustrates the percentage of AoI that is covered at the end of the algorithm execution. GREASE achieves the complete coverage with about 200 sensors. By increasing the number of sensors, GREASE continues to guarantee the coverage completeness in a shorter time. This is the reason why PDND-STOPPED shows a decreasing coverage percentage when the number of sensors increases. PDND needs more than 350 sensors to achieve a complete coverage by the time of its natural termination.

Figures 4(c) and 4(d) introduce a comparison in terms of movements. GREASE shows an increasing average traversed distance by increasing the number of sensors until the complete coverage is reached. A further increase in the number of sensors causes a decrease of the traversed distance because the movement of the additional redundant sensors is not necessary to improve the coverage. The average traversed distance under GREASE is higher than under PDND, as PDND does not complete the coverage of the AoI. Furthermore PDND makes very small movements to achieve its final deployment, whereas GREASE is slightly penalized by the use of Bug2. The shorter movements of PDND have a drawback in terms of the huge number of starting/stopping actions shown in Figure 4(d) (notice the logarithmic scale). This is an important metric for mobile sensor deployment algorithms, because start and stop actions consume high energy [12].

We now consider the average energy consumption of a sensor under the two algorithms. A sensor consumes energy due to communications (sending, receiving and listening messages) and movements (traveling and starting/stopping actions). We consider two cumulative energy consumption metrics, namely

the average energy spent in communications and the average energy consumed for all the activities. Such metrics are expressed in *energy units* (eu): the reception of a message corresponds to 1 eu, a single transmission costs 1.2 eu, a 1 meter movement costs 340 eu as a single starting or stopping action.

Figure 4(e) shows that PDND spends more energy in communications than GREASE. Indeed, under PDND, each sensor advertises its position to the neighborhood at each round. GREASE, instead, has no round based communications. Figure 4(f) shows the average energy spent for all the actions, namely communications, movements and starting/stopping. Notice the difference of two orders of magnitude between the energy consumption of the two algorithms, in favor of GREASE.

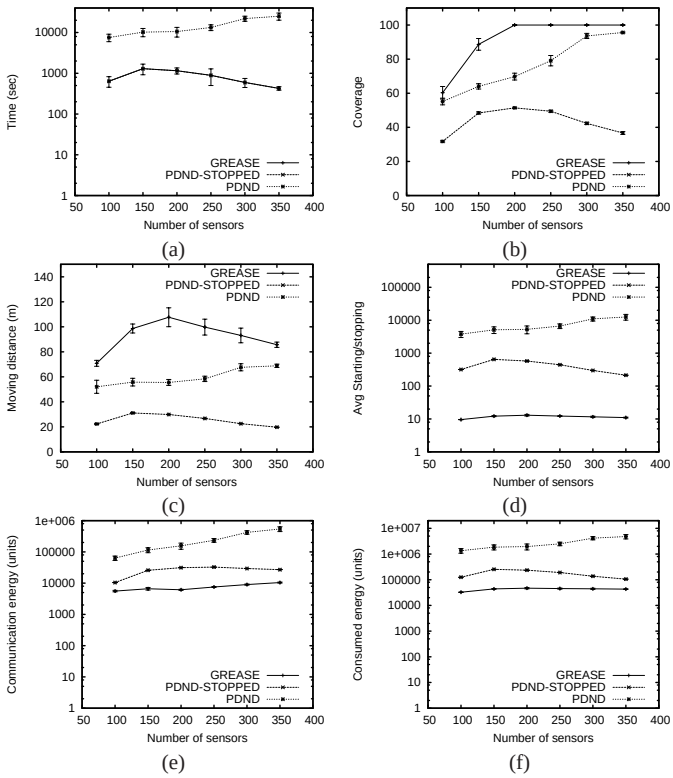


Fig. 4. Comparisons in the presence of obstacles

Experiments with ground asperities

We now describe another set of experiments conducted over an AoI of 80 m $\times$ 80 m, which contains some ground asperities as shown by the shadowed areas in Figure 5(a). The area located at the bottom right corner entails a reduction of the sensing capabilities of a factor 0.5, whereas the other area has a reduction of a factor 0.25.

The Figures 5(b) and 5(c) show that with 450 sensors both GREASE and PDND cover the target area entirely. GREASE achieves a complete coverage by performing recursive re-gridding actions in the zones with asperities. On the other hand, PDND lets the sensors located inside the zones with ground asperities advertise a sensing range that is much lower

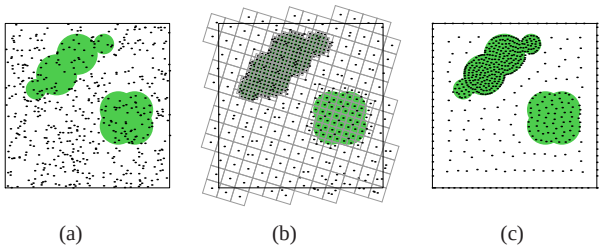


Fig. 5. Initial random deployment in the presence of ground asperities (a); Deployment after the execution of GREASE (b); Deployment after the execution of PDND (c)

than its "nominal" value, and cause a denser deployment inside the noisy areas as shown in Figure 5(c). Notice that, even though PDND seems to work properly in this operative setting, it is forcedly terminated as soon as each moving node is in an oscillatory state. The forced termination is necessary because the algorithm does not necessarily converge when the sensing capabilities are position dependent. Furthermore, this termination can only be achieved via a centralized control, that is unpractical in many situations. Although we introduced such a favorable termination technique, the algorithm PDND performs worse than GREASE.

Indeed, also in this operative context, the experiments lead to similar conclusions to those we detailed in the previous subsection. Further details are omitted due to lack of space.

REFERENCES

[1] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," *Proc. of DARS*, 2002.

[2] N. Heo and P. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 35, pp. 78–92, 2005.

[3] K. Ma, Y. Zhang, and W. Trappe, "Managing the mobility of a mobile sensor network using network dynamics," *IEEE Trans. on Parallel and Distributed Systems*, vol. 19, no. 1, pp. 106–120, 2008.

[4] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks," *ACM Trans. on Embedded Computing Systems*, vol. 3, no. 1, pp. 61–91, February 2004.

[5] G. Wang, G. Cao, and T. La Porta, "Movement-assisted sensor deployment," *IEEE Trans. on Mobile Computing*, vol. 6, pp. 640–652, 2006.

[6] M. Ma and Y. Yang, "Adaptive triangular deployment algorithm for unattended mobile sensor networks," *IEEE Trans. on Computers*, vol. 56, no. 7, pp. 946–958, 2007.

[7] N. Bartolini, T. Calamoneri, T. La Porta, A. Massini, and S. Silvestri, "Autonomous deployment of heterogeneous mobile sensors," *Proc. of IEEE ICNP*, 2009.

[8] G. Tan, S. A. Jarvis, and A.-M. Kermarrec, "Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks," *IEEE Trans. on Mobile Computing*, pp. 836–848, 2009.

[9] N. Bartolini, T. Calamoneri, E. Fusco, A. Massini, and S. Silvestri, "Autonomous deployment of self-organizing mobile sensors for a complete coverage," *ACM/Springer Wireless Networks (in press)*, 2009.

[10] M. B. Johnson, D. Sariöz, A. Bar-Noy, T. Brown, D. Verma, and C. W. Wu, "More is more: the benefits of denser sensor deployment," *Proc. of IEEE INFOCOM*, 2009.

[11] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, pp. 403–430, 1987.

[12] G. Sibley, M. Rahimi, and G. Sukhatme, "Mobile robot platform for large-scale sensor networks," *Proc. IEEE Intl Conf. Robotics and Automation*, 2002.