Noname manuscript No.
(will be inserted by the editor)

On adaptive density deployment to mitigate the sink-hole problem in mobile sensor networks

Novella Bartolini · Tiziana Calamoneri · Annalisa Massini · Simone Silvestri

Received: date / Accepted: date

Abstract The use of mobile sensors is of great relevance to monitor critical areas where sensors cannot be deployed manually. The presence of data collector sinks causes increased energy depletion in their proximity, due to the higher relay load under multi-hop communication schemes (sink-hole phenomenon). We propose a new approach towards the solution of this problem by means of an autonomous deployment algorithm that guarantees the adaptation of the sensor density to the sink proximity and enables their selective activation.

The proposed algorithm also permits a fault tolerant and self-healing deployment, and allows the realization of an integrated solution for deployment, dynamic relocation and selective sensor activation.

We formally prove the termination of our algorithm. Performance comparisons between our proposal and previous approaches show how the former can efficiently reach a deployment at the desired variable density with moderate energy consumption under a wide range of operative settings.

Keywords Mobile sensors · Self-deployment · Sink-hole problem

1 Introduction

The deployment of mobile sensors is attractive in many scenarios. For example, mobile sensors may be used for environmental monitoring to track the dispersion of pollutants, gas plumes or fires. They may also be used for public safety, for example to monitor the release of harmful agents as a result of an accident. In such scenarios it is difficult to achieve an exact sensor placement through manual means. Instead, sensors may be deployed somewhat randomly from a distance, and then reposition themselves to provide the required sensing coverage. The potential of such applications has inspired a great deal of work on algorithms for deploying mobile sensors. Most of this work has addressed the deployment of homogeneous sensors to achieve a uniform coverage of a certain density in a specific Area of Interest (AoI). When the sensor network centralizes the communications towards a single or a few sinks, the energy depletion due to communications is uneven and may possibly cause the so-called *sink-hole phenomenon* [1–3]. In this paper we address this practical and challenging problem by deploying sensors at variable densities to ensure uniform energy depletion even under imbalanced communication load.

We propose an algorithm which is based on a generalization of the Push & Pull approach presented in [4]. In summary, our contributions are:

- We identify the models of load imbalance caused by centralized communications towards one or more sinks in the network and propose a density function that models the varying density requirements over the AoI as a consequence of those unbalanced communications;
- We propose a new algorithm based on the known Push & Pull algorithm so as to allow it a more direct control over the placement of redundant sensors, to provide a sensor deployment at variable controlled density;
- We formally prove the termination of our algorithm, showing that the sensors stop moving after a finite time.
- We extend a virtual forces based algorithm to operate in a scenario with variable density requirements,

Department of Computer Science
"Sapienza" University of Rome, Italy
E-mail: {bartolini, calamo,
massini, simone.silvestri}@di.uniroma1.it

in order to make fair comparisons between our approach and the one based on virtual forces.

The Push & Pull algorithm is practical as it provides very stable sensor behavior, with fast and guaranteed termination and moderate energy consumption. It does not require manual tuning or perfect knowledge of the operating conditions, and works properly if the sensor positioning is imprecise. The algorithm does not require any synchronization during the deployment phase. The achieved deployment permits the use of alternate sensor activation that can be adopted if a loose synchronization is possible during the operative phase of the network. Because it converges quickly and does not require a priori knowledge of the deployment environment, it is also well suited for dynamic environments in which multiple sinks can be dynamically placed in consequence to dynamically changing missions.

The paper is organized as follows. Related work is presented in Section 2. In Section 3 we motivate the problem and introduce some preliminary concepts. Section 4 is the core of the paper and presents a new algorithm for variable density sensor deployment. We prove the termination of the δ-Push& Pull algorithm in Section 5. In Section 6 we show how to exploit the described algorithm to jointly solve the problem of sensor deployment, dynamic relocation, self-healing and selective activation. Section 7 is devoted to summarize a virtual force based algorithm that we use to perform experimental comparisons whose results are shown in Section 8. Section 9 concludes the paper addressing some final remarks.

2 Related Work

Various solutions have been proposed to the problem of mobile sensor self deployment. The majority of them are either based on the virtual force approach (VFA) or on computational geometry models. According to the VFA technique [5–8] the interaction among sensors is modelled as a combination of attractive and repulsive forces. Other solutions [9,10] have been inspired by different physical models. In [9] the sensors are modelled as particles of a compressible fluid and regulate their movement acting upon a diffusive behavior. In [10], the authors propose two approaches that make use of gas theory to model sensor movements in the presence of obstacles.

All these approaches require a laborious tuning of thresholds and constants to determine the magnitude of the forces and to control possible oscillations. The choice of these values influences the resulting deployment, the overall energy consumption and the convergence rate. Possible improvements to decrease the oscillations include the introduction of dissipative forces [11,12].

Most of the deployment methods based on computational geometry model the deployment problem in terms of Voronoi diagrams or Delaunay triangulations [13,14]. In [13] each sensor iteratively calculates its own Voronoi polygon, determines the existence of coverage holes and moves to a better position if necessary. The algorithm proposed in [14] exploits the Delaunay triangulation formed by the current sensor placement and moves sensors so as to position them as vertices of a triangular lattice. Similarly to the VFA approach, these proposals rely on the off-line tuning of key parameters to reduce movement oscillations.

All the above mentioned solutions do not address the sink-hole problem. In [12] a unified solution for sensor deployment and relocation crowding sensors in the presence of events is presented. This approach could be adopted to increase the sensor density in proximity of the sink. Other papers, dealing with the sink-hole problem explicitly, only focus on static sensor deployment [2, 3, 15, 16]. The aim is to mitigate the effects of the uneven energy depletion due to communication with a sink by means of a variable density deployment. In the next section we will detail some of these results that will be useful for our contribution. Only [17] specifically address the sink-hole problem in the context of mobile sensors. According to such an approach, sensor can move only once and location information are communicated to the sink which centrally compute the movement strategy and then disseminate the results to the sensors.

Many works deal with the k-coverage deployment problem. In [18], Vu and Verma reduce the problem of sensor placement with a redundancy of at least k sensors to the problem of distributing k points evenly on a torus manifold by minimizing the Riesz energy. In [19] the k-coverage sensor deployment problem is considered in both cases of the binary and probabilistic sensing models. They also distinguish the problem of sensor placement in the case of the different relation between the sensing radius r_s and communication radius r_c, i.e. $r_c < \sqrt{3}r_s$ and $r_c \geq \sqrt{3}r_s$ and propose two different dispatch schemes.

The k-coverage sensor placement can be obtained by shrinking a grid deployment until the k-coverage is achieved. In both [4] and [20] the shrinking is used to obtain a denser hexagonal grid.

In the present work, a redundant coverage with adaptive redundancy level k is obtained by superimposing several grid translated from each other to the purpose of achieving a variable controlled density deployment.

Furthermore the k-coverage is exploited to the purpose of ensuring uniform energy depletion by performing a selective activation of the sensors.

3 Energy consumption due to communications

Li and Mohapatra address the sink-hole problem in [2]. The authors analyze the applicative context of environmental monitoring and data gathering. In this context they assume that each sensor generates new traffic with a constant bit rate (CBR) and sends it to the sink via multi-hop communications. The examined deployment consists of a uniform random placement of devices over the AoI, where N is the total number of devices and A_{net} is the measure of the area of the AoI, hence the uniformly deployed density is $\rho = N/A_{net}$. Sensors transmit their packets to the destination by selecting the next-hop which is closest to the destination.

The authors propose a model to evaluate the per-node energy consumption, by considering three main contributions, namely energy spent for sensing, transmissions and receptions. They divide the AoI into several concentric circular crowns of radius equal to the transmission range r, centered at the sink position. The energy consumption of the sensors is then calculated separately in each crown.

According to this model the per-node energy consumption of the i-th crown is the following:

$$ECR_{ring^{ith}} = \alpha_1 b + \gamma_1 \frac{(\frac{M^2}{\pi} - (i+1)^2)}{2i+1} b + (\beta_1 + \beta_2 r^n) \frac{(\frac{M^2}{3} - i^2)}{2i+1} b \quad (1)$$

where $i = 0, 1, \ldots, (\frac{M}{2} - 1)$, and the parameters are the following: b is the constant bit rate generated by each sensor, α_1, β_1, β_2 and γ_1 are technology dependent constant factors that are considered in the definition of the three energy contributions mentioned above, and the AoI is divided into $\frac{M}{2}$ concentric circular crowns with a step size of r meters.

Also Olariu and Stojmenović deal with the sink-hole problem in [3]. The authors also consider a uniformly deployed sensor network, with devices transmitting the same number of reports towards the sink. The authors conclude that the energy consumption of sensors located inside the i-th circular crown centered at the sink, and determined by the radii r_{i-1} and r_i, is as follows:

$$E_i = \frac{T}{\rho\pi} \left[1 - \frac{r_{i-1}^2}{{r_k}^2}\right] \frac{(r_i - r_{i-1})^\alpha + c}{r_i^2 - r_{i-1}^2} \quad (2)$$

where T is the number of tasks handled by the network during its lifetime, c is a technology dependent positive constant, $\alpha > 2$ is the power attenuation and ρ is the sensor uniform density over the AoI.

Finally the problem of uneven energy depletion due to many-to-one communications is addressed in [1] under nonuniform sensor deployment. The authors find a suboptimal deployment technique to ensure energy efficiency and mitigate the sink-hole problem. They propose to deploy sensors into circular crowns at different densities where the ratio between the sensor densities of the adjacent $(i+1)$-th and the i-th crowns is equal to

$$\frac{\rho_{i+1}}{\rho_i} = \frac{(2i-1)}{q(2i+1)} \quad (3)$$

and $q > 1$ is the geometric proportion defining the increase in the number of sensors from the outer to the inner crowns. The circular crowns are centered at the sink position, and are dimensioned so as to ensure that the sensors of each crown act as forwarders for the outer crowns.

The authors assume a constant bit rate generated by each sensor and two energy contributions due to transmissions and receptions.

In this paper we refer to the above mentioned work [1] to define the non-uniform density requirements to be addressed by the deployment algorithm in order to balance the energy consumption among the sensors of the network. By deploying the sensors according to Equation (3) the proposed approach ensures the network energy efficiency and prolong the network lifetime avoiding the generation of sink holes due to communications.

4 Variable density self deployment of mobile sensors

The proposed algorithm, called δ-Push&Pull, is inspired by the algorithm introduced in [4], to which we made major modifications to the purpose of deploying sensors at variable densities according to position dependent requirements.

Given a point P in the AoI, we define $\delta(P)$ the coverage density required in position P. Let V be a set of equally equipped sensors able to determine their own location, endowed with boolean sensing capabilities and isotropic sensing and communication model. Notice that location capabilities are only necessary to recognize the borders of the AoI while, in order to make movement decisions, each sensor only needs to know the position of its communicating neighbors.

As in its original counterpart, according to δ-Push&Pull, the sensors aim at realizing a complete coverage of the AoI and a connected network by means of a hexagonal tiling deployment, where the side of each hexagon is set to the sensing radius r_s. The hexagonal tiling is realized by snapping the necessary number of sensors over the AoI in grid positions located in correspondence to the vertices of a triangular lattice with side $\sqrt{3}r_s$. Such sensors will be referred to as *snapped*. Given a snapped sensor x, we refer to $Hex(x)$ as to the hexagonal area that is covered by the sensor x and to P_x as to the position of the sensor x.

At the same time, δ-Push&Pull deploys redundant sensors over the covered area, by distributing them at variable density, according to $\delta(P)$ as follows: the number of sensors that will be located in $Hex(x)$ centered at P_x is

$$n_\delta(P_x) = \lceil \delta(P) \cdot \frac{3\sqrt{3}}{2} r_s^2 \rceil$$

The $n_\delta(P_x) - 1$ sensors utilized to obtain the desired density in a specific hexagon will be indicated as *adjunct-snapped sensors*. The sensors located in $Hex(x)$ which are neither snapped nor adjunct-snapped will be named *slaves* of x. We hereafter refer to $S(x)$ as the set of slave sensors of x.

The algorithm starts with the concurrent creation of several tiling portions. Every sensor not yet involved in the creation of a tiling portion gives start to its own portion in an instant which is randomly selected in a given time interval. Such a starter sensor is called $s_{\texttt{init}}$. The algorithm consists of four main interleaved activities: *snap*, *push*, *pull* and *merge*.

4.1 Snap activity

The sensor $s_{\texttt{init}}$ elects its position P_{init} as the center of the first hexagon of its tiling portion. It collects information on the sensors in radio proximity, that will compose the set $L(s_{\texttt{init}})$. Among the sensors located in its own hexagon, $s_{\texttt{init}}$ chooses up to $n_\delta(P_{init}) - 1$ sensors for the role of adjunct-snapped. Such sensors will remain in their original hexagon and will not participate in the following activities. The sensors belonging to $L(s_{\texttt{init}})$ which have not been declared adjunct-snapped can be used to cover adjacent hexagons. To this purpose, $s_{\texttt{init}}$ selects at most six sensors among those belonging to $L(s_{\texttt{init}})$ and makes them snap to the center of adjacent hexagons. Such deployed sensors, in turn, give start to their own selection and snap activity, thus expanding the boundary of the current tiling portion. This process continues until no other snaps are possible, because either the whole AoI is covered, or the boundary tiles do not contain any unsnapped sensors.

Sensor x starts the push activity if slave sensors are still present in $Hex(x)$ after the adjunct-snapped declaration and the adjacent positions are all covered by snapped sensors. By contrast, sensor x starts the pull activity if (1) the number of adjunct-snapped sensors is lower than necessary to fulfill the density requirement, or (2) some hexagons adjacent to $Hex(x)$ are left uncovered and x has no slaves.

All the snapped sensors position the adjunct-snapped sensors in their hexagon according to a same common rule. This way it is possible to obtain the desired distribution of sensors over the hexagon area. Moreover, it is possible to perform a selective sensor activation which allows energy saving during the operative phase of the network, giving rise to alternate activation of different hexagonal grids composed by adjunct-snapped sensors in the same position. Obviously, these adjunct grids have the same coverage and connectivity features of the main hexagonal grid, that is the grid composed by the snapped sensors.

4.2 Push activity

After the completion of their snapping activity, snapped sensors may have slave sensors located inside their hexagon. In this case, they pro-actively push such slave sensors towards the areas demanding a higher number of sensors. Consequently, slave sensors being in overcrowded areas migrate to zones with unsatisfied density requirements.

In order to avoid endless cyclic movements of slaves, we introduce the following δ-Moving Condition. The offer of slave sensors by a sensor x to a sensor y located in radio proximity is allowed if and only if:

$$\begin{aligned} &\{|S(x)| > (|S(y)| + 1)\} \vee \\ &\{|S(x)| = (|S(y)| + 1) \wedge id(x) > id(y)\} \end{aligned} \quad (4)$$

where $id(\cdot)$ is a function initially set to the unique identity code of the sensor radio device.

If the δ-Moving Condition is verified, sensor x can push at least one of its slaves towards the destination hexagon $Hex(y)$ selected as the one that needs a higher number of sensors to fulfill the local density requirements or to fill an adjacent coverage hole; among the slave sensors which can be pushed to the destination, x selects the closest to $Hex(y)$.

4.3 Pull activity

The sole snap and push activities are not sufficient to ensure the maximum expansion of the tiling and the

achievement of a deployment at the required density. In the δ-Push&Pull algorithm, the pull activity starts whenever a sensor x notices either a hole in its adjacent snapping position or a density in $Hex(x)$ that is lower than $n_\delta(P_x)$.

Snapped sensors may detect a coverage hole adjacent to their hexagon and may not have available sensors to make them snap. Similarly, a snapped sensor may need more adjunct-snapped sensors than available to fulfill the density requirements. In these cases, they send hole trigger messages, and re-actively attract non-snapped sensors and make them fill the hole or the density gap.

In order to start the pull activity, sensor x broadcasts an invitation message at a higher and higher number of hops, until it receives an acceptance of invitation from a snapped sensor having a redundant slave. The inviter acknowledges the acceptance message if it has not found a number of slave sensors sufficient to fill the hole or the density gap, or reject it otherwise. In the former case, an agreement has been reached between the two sensors and the slave can start moving. When the snapped sensor that is performing the pull activity reaches its objective (to fill either the hole or the density gap), it stops sending slave invitation messages.

4.4 Merge activity

The possibility that many sensors act as starters can give rise to several tiling portions with different orientations. In order to characterize and distinguish each tiling portion, the time-stamp of each starter is included in the header of all exchanged messages. Then, messages coming from sensors located in different tiling portions include different starter time-stamps. When the boundaries of two tiling portions come in radio proximity with each other, the one with older starter time-stamp absorbs the other one by making its snapped sensors move into more appropriate snapping positions. Hence this activity provides a mechanism to merge all the tiling portions into a unique regular and uniformly oriented tiling, simply adjusting the positions of already snapped sensors.

We conclude this description of the algorithm with an activity called **role exchange**. According to the previous description of δ-Push&Pull, slaves consume more energy than snapped and adjunct-snapped sensors, because they are involved in a larger number of message exchanges and movements. We introduce a *mechanism to balance the energy consumption* over the set of available sensors making them exchange their roles. This mechanism is similar to the technique of *cascaded movements* introduced in [21]. Namely, any time a slave has to make a movement across a hexagon as a consequence of either push or pull activities, it evaluates the opportunity to substitute itself with the snapped and adjunct-snapped sensors of the hexagon it is traversing. The criterion at the basis of this mechanism is that two sensors exchange their role whenever the energy imbalance is reduced. As a result, the energy balance is significantly enhanced, though the role exchange has a small cost for both the slave and the snapped sensor involved in the substitution. Indeed, the slave sensor has to reach the center of the current hexagon and perform a *profile packet* exchange with the snapped sensor that has to move towards the destination of the slave. A profile packet contains the key information needed by a sensor to perform its new role after a substitution.

4.5 An example of the algorithm execution

Figure 1 illustrates the interleaved execution of the algorithm actions through an example. For simplicity, we do not consider the role exchange activity.

Figure 1(a) shows a starting configuration in which a sink is positioned in the central point of the right vertical side of the AoI and requires a density variation in its proximity. The sensor 8 assumes the starter role.

This sensor snaps three of its slaves, as shown in Figure 1(b), where the *id* values of such snapped sensors are highlighted.

Figure 1(c) shows that the snapped sensor 8 has some un-snapped sensors in its hexagon, and therefore starts the push activity towards its three adjacent hexagons. In the meantime, the sensor 4 acts as starter and another grid portion is initiated. As it is in a zone with density requirement 4, it designates the sensors 20, 36 and 11 as adjunct-snapped.

In Figure 1(d) the snapped sensor 19 detects a coverage hole. As it has an un-snapped sensor in its hexagon, it performs the snap activity. The sensor 6 must satisfy a density requirement 2, so it designates the sensor 34 as adjunct-snapped. Notice that the snapped sensor 1 does not have any hole around its hexagon, so its slave remains where it is; furthermore, it does not execute any push action as the Moving Condition is not satisfied. The snapped sensor 8, having many slaves, continues its push activity. At the same time, the snapped sensor 4 snaps three of its slaves. Figure 1(e) shows that, while the snapped sensors 4 and 8 continue their push activities, the sensors 3 and 7 start the pull activity, as both detect a coverage hole and do not have any slaves to snap, so new sensors are snapped in the left grid.

In view of the pull activity, some sensors arrive in the hexagons of sensors 3 and 7, and become adjunct-

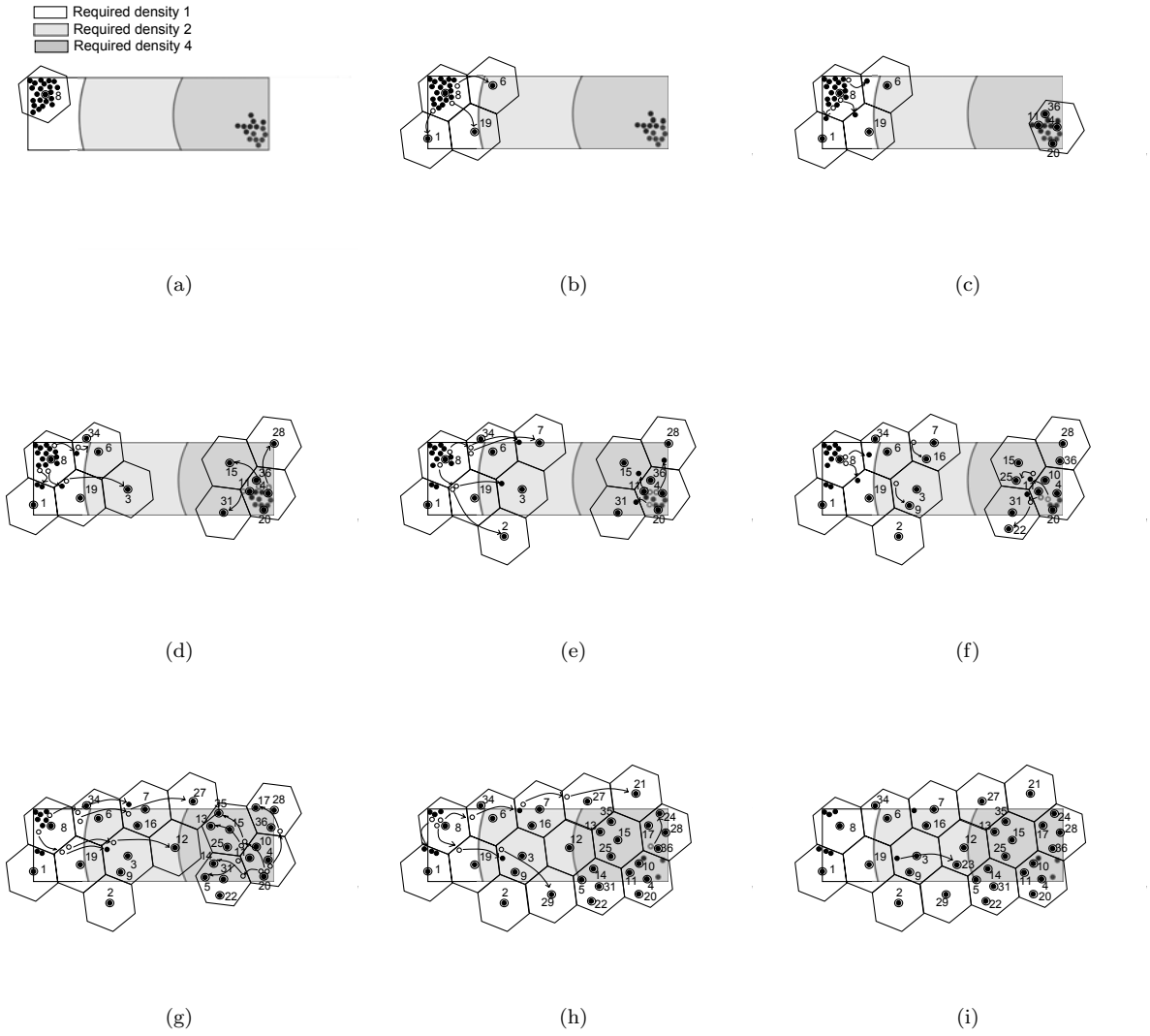


Fig. 1 Algorithm execution: an example

snapped. The same happens in the right grid, with sensors 15, 28 and 31 – see Figure 1(f). The sensors 4 and 8 continue their push activity.

In Figure 1(g) the snapped sensors 4 and 8 continue their push activity while some new sensors are snapped. In the meantime, the snapped sensors in the zone with density requirement 4 designate some adjunct-snapped sensors.

As soon as the grid portions come in radio proximity with each other, the tiling merge activity is started (Figure 1(h)) and a unique grid is built. The adjunct-snapped sensors located inside the hexagon of the sensor 31 will change their status from adjunct-snapped to slaves, because the sensor 31 has been snapped outside the AoI in consequence of the merge activity. Finally, Figure 1(j) concludes this example, showing the last activities performed to completely cover the AoI.

5 Algorithm termination

In this section we formally prove the termination of our algorithm. Let $L = \{\ell_1, \ell_2, \ldots, \ell_{|L|}\}$ be the set of snapped sensors.

Definition 1 A *network state* is a vector $\mathbf{s}$ whose i-th component represents the number of slave sensors deployed inside the hexagon $Hex(i)$ governed by the snapped sensor i. Therefore $\mathbf{s} =< s_1, s_2, \ldots, s_{|L|} >$ where $s_i = |S(i)|$, $\forall i = 1, \ldots, |L|$.

Notice that, adjunct-snapped sensors are not considered in the network state as they are not considered by the push activity.

Definition 2 A state $\mathbf{s} =< s_1, \ldots, s_{|L|} >$ is *stable*, if the Moving Condition is false for each couple of snapped sensors in L located in radio proximity to each other.

Theorem 1 *Algorithm δ-Push&Pull terminates in a finite time.*

Proof As long as new sensors are being snapped, the covered area keeps on growing. This process eventually ends either because the AoI has been completely covered or because the sensors have reached a stable state. In order to prove the theorem, it suffices to prove that, once the algorithm has reached the maximum coverage for the current execution (i.e. no more snap actions are performed), the network reaches a stable configuration in a finite time. Therefore we can consider the set of snapped sensors L as fixed. The value of the order function related to each snapped sensor, $id(\ell_i)$, is set during the unfolding of the algorithm, it can be modified only temporarily by the pull activity a finite number of times and remains steady onward. Let us define $f : \mathbb{N}^{|L|} \rightarrow \mathbb{N}^2$ as follows:

$$f(\mathbf{s}) = \left(\sum_{i=1}^{|L|} s_i^2, \sum_{i=1}^{|L|} s_i \cdot id(\ell_i) \right) \tag{5}$$

We say that $f(\mathbf{s}) \succ f(\mathbf{s}')$ if $f(\mathbf{s})$ and $f(\mathbf{s}')$ are in lexicographic order. Observe that the function f is lower bounded by the pair $(|L|, \sum_{i=1}^{|L|} id(\ell_i))$, in fact $1 \leq s_i \leq |V|$. Therefore, if we prove that the value of f decreases at every state change, we also prove that no infinite sequence of state changes is possible.

To this purpose, let us show that every state change from $\mathbf{s}$ to $\mathbf{s}'$ causes $f(\mathbf{s}) \succ f(\mathbf{s}')$. Let us consider a generic state change which involves the snapped sensors x and y, with x sending a slave sensor to $Hex(y)$. We denote with $as(y)$ the number of adjunct-snapped sensors currently located in $Hex(y)$. Two cases may occur:

$$(1)\ as(y) < \left\lceil \delta(P_y) \frac{3\sqrt{3}}{2} r_s^2 \right\rceil$$

The snapped sensor y still needs some sensors to meet the density requirement.

$$(2)\ as(y) = \left\lceil \delta(P_y) \frac{3\sqrt{3}}{2} r_s^2 \right\rceil$$

The snapped sensor y has already fulfilled the density requirements, i.e. there is a sufficient number of adjunct-snapped sensor in $Hex(y)$.

Notice that, it is not possible that $as(y) > n_\delta(P_y)$, as the algorithm provides that the status of all exceeding sensors with respect to $n_\delta(P_y)$ is set to slave. In the following we prove separately that $f(\mathbf{s}) \succ f(\mathbf{s}')$ in both cases.

Case (1): the pushed sensor will be an adjunct-snapped of the snapped sensor y. We have that $s'_i = s_i \quad \forall i \neq x, y$, $s'_x = s_x - 1$ and $s'_y = s_y$. Thus, $f(\mathbf{s}) \succ f(\mathbf{s}')$ because $\sum_{i=0}^{L} s'_i < \sum_{i=0}^{L} s_i$.

Case (2): the pushed sensor will be a slave of the snapped sensor y. We have that $s_i = s'_i \quad \forall i \neq x, y$, and $s'_x = s_x - 1$ and $s'_y = s_y + 1$. As the transfer of the slave has been done according to the Moving Condition (Equation 4), two cases are possible: either $s_x > s_y + 1$, or $(s_x = s_y + 1) \wedge (id(x) > id(y))$. In the first case, the inequality $s_x > s_y + 1$ implies that $\sum_{i=1}^{|L|} s_i^2 > \sum_{i=1}^{|L|} s'^2_i$. In the second case, since $s_x = s_y + 1$ and $id(x) > id(y)$, lead to $\sum_{i=1}^{|L|} s_i^2 = \sum_{i=1}^{|L|} s'^2_i$ and $\sum_{i=1}^{|L|} s_i \cdot id(\ell_i) > \sum_{i=1}^{|L|} id(\ell_i) s'_i$. Therefore in both cases $f(\mathbf{s}) \succ f(\mathbf{s}')$.

The function f is lower bounded and always decreasing by discrete quantities (integer values) at any state change. Thus, after a finite number of steps, it is impossible to perform a further state change, i.e. the network will be in a *stable state* in a finite time.

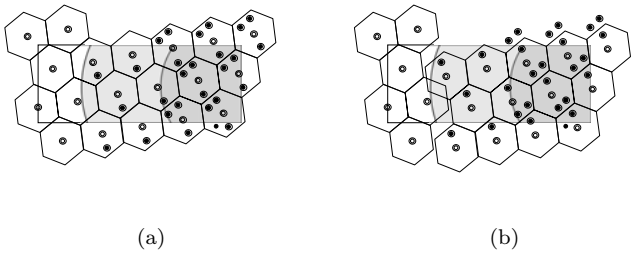
6 Joint solution to sensor deployment, selective activation, self-healing and dynamic relocation

6.1 Selective activation

Our approach relies on the availability of a sufficient number of sensors to cover each hexagonal tile at the required density, namely with a given number of adjunct-snapped sensors. If the necessary number of sensors is available, the algorithm achieves a complete coverage, with a regular pattern that permits the use of topology control algorithms [22] and allows a selective sensor activation which saves energy during the operative phase of the network. As already highlighted, each snapped sensor will place its adjunct-snapped in fixed positions according to a predefined oriented pattern inside each hexagonal tile.

The deployment of the adjunct-snapped sensors according to the same pattern in each tile with the same density requirements, allows us to define a *selective activation pattern*. The selective activation of the sensors in a pattern guarantees the continuity and completeness of the coverage of the tiles that belong to the same circular crown.

When in an AoI there are crowns with different density requirements, temporary holes can appear along



(a) (b)

Fig. 2 Coverage holes at the borders of the circular crowns during the execution of the alternate activation of the adjunct-snapped sensors.

the boundary of these zones since sensors in different positions of the hexagons are activated in neighboring areas. This situation is described in Figure 2. Observe that the coverage discontinuity of Figure 2(b) is only intermittent, and many real applications may not suffer from it. Indeed, for some applications a continuous sensing of the AoI is not required, for example in the case of monitoring systems for the detection of pollutant levels, temperature or humidity conditions. In these cases, the monitoring activity can rely on the sole interpolation of local measurements taken at discrete points in the AoI.

By contrast, other more critical applications require that every point in the target area be accurately monitored, for example when the sensors are deployed to monitor the presence of human-life threats such as radioactive or chemical plumes or a forest fire. In these cases, coverage discontinuities can be eliminated by positioning the adjunct-snapped sensors in the *wiggle region* of the snapped sensor. Indeed, the wiggle region has been defined in [20] as the region comprising all those points in which a sensor could be repositioned such that full coverage is maintained. Of course, the adoption of the wiggle region requires a slight shrinking of the hexagonal lattice. In particular, if w is the radius of the circle inscribed in the wiggle region, then the grid size must be set to $\sqrt{3}(r_s - w)$, instead of $\sqrt{3}r_s$. It follows that in order to create a wiggle region that is sufficiently large to accommodate all the adjunct-snapped sensors, it is necessary to deploy a larger number of sensors.

Notice that only a loose clock synchronization is actually necessary to perform the described selective activation scheme.

6.2 Self-healing and dynamic relocation

The proposed algorithm ensures that, when a sufficient number of sensors are available, the density requirements defined in correspondence to the center of each tile, will be fulfilled. Nevertheless, the algorithm does not give any indication on where to place redundant slave sensors, which instead are uniformly spread over the network as a consequence of the push activity. The redundant slave sensors will thus be available to recover possible failures. More in detail, as soon as a coverage hole is detected by the sensors located in proximity (for example, the detection may happen thanks to a periodic polling scheme or signalled by a failing sensor whose battery is almost exhausted), the detecting sensors can restart the algorithm with the consequence that the hole is immediately covered or a pull activity is executed to attract the closest slave sensors. The redundant slave sensors can thus be dynamically relocated to respond to pull invitations issued by the sensors located nearby failed devices. This process endows the network with self-healing and self-adapting capabilities that are not present in previous solutions.

In addition, a sensor network application may require sensor relocation capabilities (see [12, 21]) also to respond to dynamically occurring events when the deployment of new sensors is not possible, and the only choice is to re-use and move the available ones. In consequence of a dynamically occurred event, each snapped sensor may declare a new density requirement, which better reflects the required position dependent accuracy.

This way the new set of redundant slave sensors become available to respond to new pull invitations necessary to reactivate the algorithm execution and fulfill the new density requirements.

7 On the use of the virtual force approach for variable density deployment

In order to evaluate the performance of the δ-Push&Pull algorithm proposed in this paper, we compare it with an algorithm based on virtual forces called Parallel and

Distributed Network Dynamics (PDND), proposed in [23]. In PDND the force exerted by the sensor s_i on the sensor s_j is modelled as a piecewise linear function. It is repulsive when the distance between s_i and s_j is lower than an arbitrarily tuned parameter r^*; it is attractive when the distance is larger, until it vanishes at another arbitrarily set distance. In order to ensure the convergence of PDND, the formulation of this force must respect the condition of Lipschitz continuity. In this case, the single sensor movement is limited by an upper bound that guarantees that the potential energy is always decreasing, hence avoiding oscillations.

PDND works under the assumption that density requirements are uniform over the AoI. The algorithm PDND addresses the problem arising when a sensor that approaches the boundary of the AoI calculates a target position outside the sensing field. Since sensors have a prior knowledge of the shape of the boundary, in this case the sensor calculates the point that is closest to the target position inside the AoI and moves to that point following the optimal path. This point is obtained by decomposing the part of the movement vector outside the AoI in its orthogonal components (using the boundary line as an axis) and using the sole component which is parallel to the boundary line.

In order to make the algorithm achieve a variable density deployment, we need to redefine the force that one sensor exerts on the others. According to the algorithm PDND, this implies the definition of the rest distance r^* at which the force exerted by two interacting sensors is null. More specifically, we assign to all sensors inside a region with the same density requirement a position dependent virtual sensing radius. In particular, given a sensor x located at P_x, we set the virtual sensing radius $r_s^{virtual}(x)$ to the radius of a regular hexagonal tessellation that would be obtained by optimally deploying the sensors at the desired density, that is:

$$r_s^{virtual}(x) = \sqrt{\frac{2}{3\sqrt{3}\delta(P_x)}}$$

We consider a value of r^* that allows to minimize the overlaps among sensing disks, obtained as a combination of the sensing radii of two interacting sensors i and j, r_i and r_j, namely $r^* = r_i + r_j$. This value of r^* models the interaction between two sensors trying to position themselves so that their sensing circles are tangential.

It is to notice that the discontinuity of the density requirements over the AoI implies a discontinuity in the force function, that no longer respects the Lipschitz condition. For this reason, the convergence of the algorithm PDND is no longer guaranteed. In this particular setting, PDND looses its peculiar characteristic of guaranteed convergence and behaves as all the other algorithms based on virtual forces that, since the inspiring model is inherently dynamic, are prone to oscillations. In order to halt the execution of the PDND algorithm, we introduce a centralized oscillation control method as in [6]. By examining the history of movements of each sensor, we determine if oscillations are going on by checking if the sensor has moved back and forth around the same location many times. More formally, we say that a sensor is in an oscillatory state if in the last m movements it has not moved away more than ϵ_m meters from the barycenter of such movements. We artificially terminate the algorithm as all the sensors are in an oscillatory state. We highlight that, although impractical, this oscillation control is of benefit for the performance of PDND and, for this reason, our comparisons are fair.

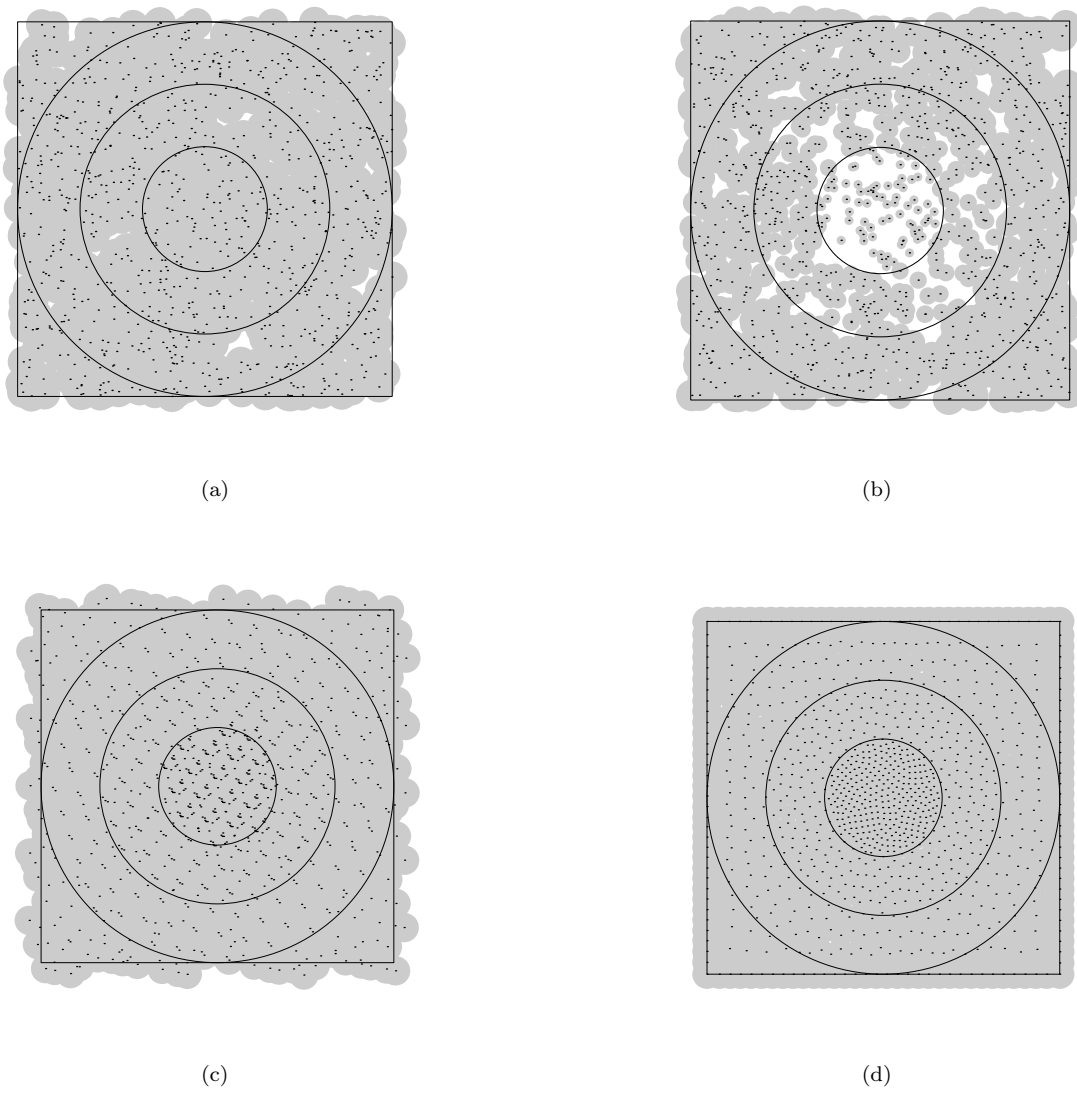
8 Simulation results

In this section we compare our proposal with the PDND algorithm, adapted to our context as described in Section 7. To this purpose, we developed an OPNET based simulator. We use the following parameter setting: $r_{\mathtt{tx}} =$ 10 m, $r_{\mathtt{s}} = 5$ m, sensor speed $v = 1$ m/sec. We consider a squared AoI of 120 m $\times$ 120 m with three concentric circular crowns, centered at the sink position, located at the center of the AoI. According to [1], each crown has a different density requirement increasing geometrically towards the sink as described by Equation 3. In particular, we set the density requirement of the most external zone to one sensor per hexagon, and we use a parameter $q = 1.2$ for the geometric progression. In such a setting, the crown density requirements are 1, 2, 4 and 12 sensors per hexagon as we move from the outer to the inner crown.

We consider a random sensor initial deployment, as depicted in Figure 3(a). Figure 3(b) also shows an example of the initial deployment, highlighting the position dependent sensing radii used under PDND. Figure 3(c) and 3(d) show an example of the final deployment achieved with 950 sensors by δ-Push&Pull and PDND, respectively. The clusters of nodes of Figure 3(b) are motivated by the random choice of the positions of the adjunct-snapped sensors. As it will be explained in the following, PDND achieves a more uniform deployment at the cost of a higher energy consumption and deployment time.

In order to compare the performance of the two algorithms we increase the number of deployed sensors from 800 to 1100. The results are obtained by averaging over 30 simulation runs.

Figure 4(a) shows the completion time, i.e. the time required to reach the final deployment. Recall that the PDND algorithm is artificially halted since it does not



(a) (b) (c) (d)

Fig. 3 Initial configuration with homogeneous (a) and position dependent (b) sensing radii. Final deployment under δ-Push&Pull (b) and PDND (c).

guarantee the termination. Despite this external intervention to halt the execution of PDND, the termination time of δ-Push&Pull is two orders of magnitude shorter than PDND. The slowness of PDND is due to the limitation to the distance each sensor is allowed to traverse at each round. On the other hand, δ-Push&Pull let sensors traverse entire hexagons at each movement, thus resulting in a shorter termination time.

Figure 4(b) shows the average traversed distance. δ-Push&Pull has a decreasing traversed distance as the number of sensors increases. This is due to the fact that less sensors have to be pulled in order to achieve the desired density as the number of deployed sensors increases. The PDND algorithm shows a higher traversed distance than δ-Push&Pull due to the oscillating movements typical of virtual force based solutions.

The average number of starting/stopping actions is shown in Figure 4(c). This is an important metric for mobile sensor deployment algorithms, because start and stop actions consume high energy [13]. PDND shows an average number of starting/stopping two orders of magnitude higher than δ-Push&Pull. As for the deployment time, this is due to the short distance each sensor can traverse at each round. δ-Push&Pull, instead, moves the sensors precisely and without oscillations, resulting in a lower number of movements.

We now consider the average energy consumption of a sensor under the two algorithms. A sensor con-

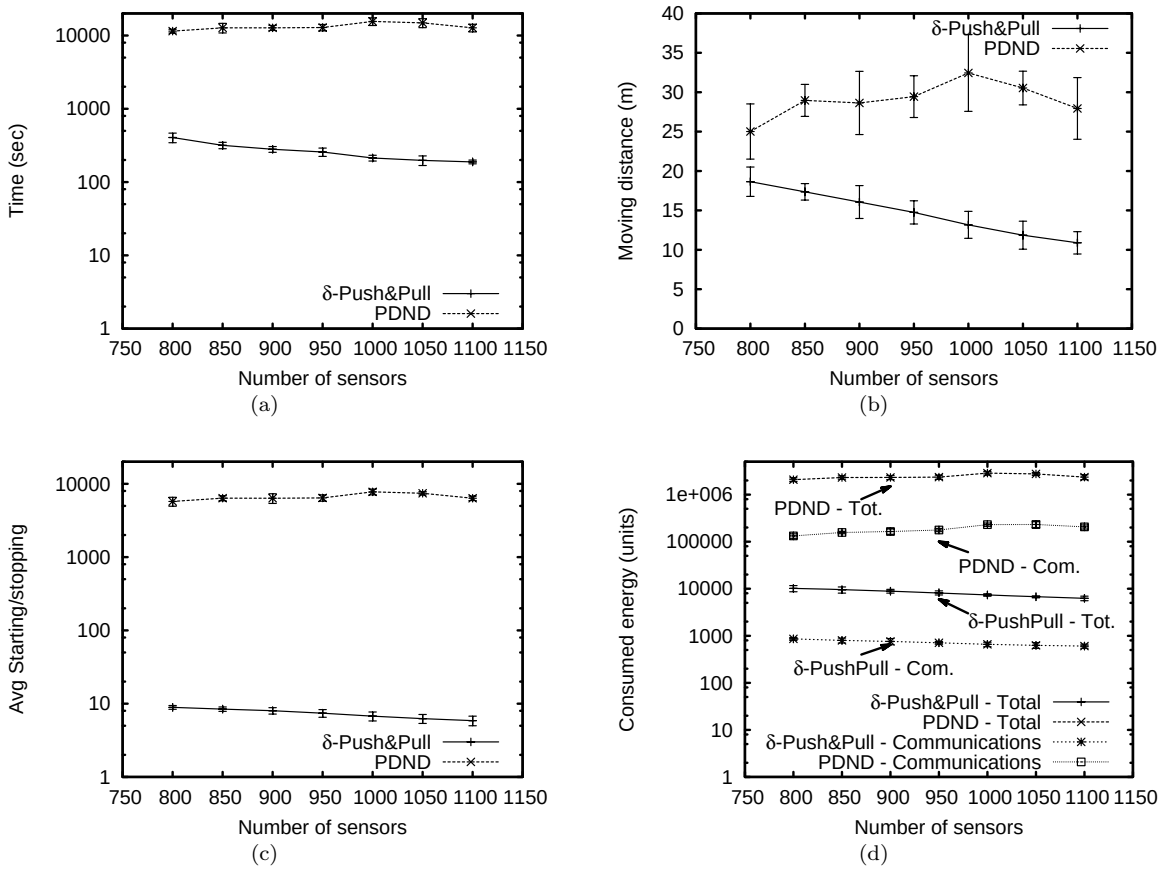


Fig. 4 Performance comparisons between δ-Push&Pull and PDND.

sumes energy due to communications (sending and receiving messages) and movements (travelling and starting/stopping movements). We consider two cumulative energy consumption metrics, namely the average energy spent in communication and the average total energy consumed by sensors. Such metrics are expressed in energy units: the reception of a message corresponds to one energy unit, a single transmission costs the same as 1.125 receptions [24], a 1 meter movement costs the same as 300 transmissions [13] and a starting/stopping action costs the same as 1 meter movement [13].

Figure 4(d) shows the energy spent in communications and the total energy consumption. As expected, PDND has worse performance under both metrics. On the one hand, the energy spent in communications is higher because of the high number of rounds required by PDND to terminate. Indeed, under PDND, each sensor advertises its position to the neighborhood at each round. δ-Push&Pull, instead, has no round based communications, and messages are only exchanged to perform the algorithm activities. On the other hand, the higher number of starting/stopping actions as well as the higher traversed distance, result in a major total energy consumption of PDND with respect to δ-Push&Pull.

We finally evaluate the two algorithms considering the quality of the achieved deployments. We compared the percentage of AoI not meeting the desired density at the end of the algorithm execution. The results are shown in Figure 5. The regularity of the deployment achieved by PDND results in a better fulfilment of the requirements. However, such regularity is achieved at the cost of a higher energy consumption and a longer deployment time. δ-Push&Pull consumes two orders of magnitude less energy with respect to PDND, and is able to achieve a final stable deployment in a much shorter time. It shows a small gap in the percentage of area not meeting the desired density, that decreases as the number of sensors increases. This gap corresponds to the boundaries between adjacent circular crowns. Indeed, the density requirement of a tile is advertised according to the position of its snapped sensor. Nevertheless, when a tile is crossed by the boundary line of a circular crown, one of the two sections lies on a crown where the density requirement is different from the one declared by the snapped sensor.

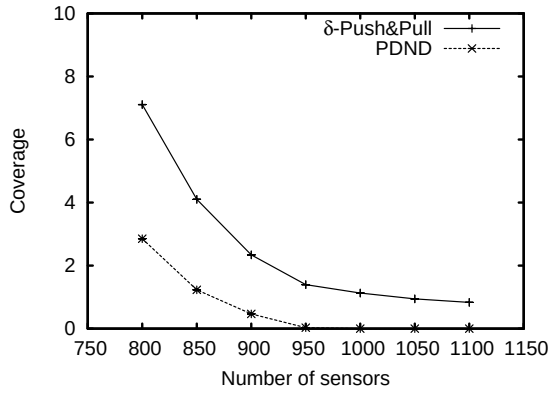


Fig. 5 Percentage of area not meeting the density requirements.

9 Conclusions

We proposed an original algorithm for mobile sensor self deployment, according to which sensors autonomously coordinate their movements to achieve a complete coverage with variable density. The sensor density varies so as to uniform the energy depletion due to communications towards the sink. The final deployment consists in a hexagonal tiling with a variable number of sensors deployed in each tile. We formally prove the termination of our algorithm. Simulations show that our algorithm performs better than previous approaches in terms of several performance parameters. Furthermore, we discussed some of the benefits related to the regularity of the obtained deployment. In particular we show how the regularity of the sensor distribution can be exploited to implement energy saving techniques and to achieve fault tolerance and self-healing capabilities.

References

1. Wu, X., Chen, G., Das, S.K.: On the energy hole problem of nonuniform node distribution in wireless sensor networks. IEEE Transactions on Parallel and Distributed System **19**(5) (2008) 710–720
2. Li, J., Mohapatra, P.: Analytical modeling and mitigation techniques for the energy hole problem in sensor networks. Pervasive and Mobile Computing (3) (2007) 233–254
3. Olariu, S., Stojmenovic, I.: Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting. Proc. of INFOCOM (2006)
4. Bartolini, N., Calamoneri, T., Fusco, E., Massini, A., Silvestri, S.: Push & pull: autonomous deployment of mobile sensors for a complete coverage. ACM/Springer Wireless Networks (2009)
5. Zou, Y., Chakrabarty, K.: Sensor deployment and target localization based on virtual forces. Proc. IEEE INFOCOM (2003)
6. Heo, N., Varshney, P.: Energy-efficient deployment of intelligent mobile sensor networks. IEEE Transactions on Systems, Man and Cybernetics **35** (2005)
7. Chen, J., Li, S., Sun, Y.: Novel deployment schemes for mobile sensor networks. Sensors **7** (2007)
8. Poduri, S., Sukhatme, G.S.: Constrained coverage for mobile sensor networks. Proc. of IEEE ICRA (2004)
9. Pac, M.R., Erkmen, A.M., Erkmen, I.: Scalable self-deployment of mobile sensor networks; a fluid dynamics approach. Proc. of IEEE IROS (2006)
10. Kerr, W., Spears, D., Spears, W., Thayer, D.: Two formal fluid models for multi-agent sweeping and obstacle avoidance. Proc. of the Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS) (2004)
11. Howard, A., Mataric, M.J., Sukhatme, G.S.: Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. Proc. of DARS (2002)
12. Garetto, M., Gribaudo, M., Chiasserini, C.F., Leonardi, E.: A distributed sensor relocation scheme for environmental control. The ACM/IEEE Proc. of MASS (2007)
13. Wang, G., Cao, G., Porta, T.L.: Movement-assisted sensor deployment. IEEE Transaction on Mobile Computing **6** (2006)
14. Ma, M., Yang, Y.: Adaptive triangular deployment algorithm for unattended mobile sensor networks. IEEE Transactions on Computers **56** (2007)
15. Wu, X., Chen, G., Das, S.K.: On the energy hole problem of nonuniform node distribution in wireless sensor networks. Proc. of IEEE MASS (2006) 180–187
16. Cardei, M., Yang, Y., Wu, J.: Non-uniform sensor deployment in mobile wireless sensor networks. Proc. of WoWMoM (2008) 1–8
17. Yang, Y., Cardei, M.: Movement-assisted sensor redeployment scheme for network lifetime increase. In: Proc. of the ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems (MASS). (2007) 13–20
18. Wu, C., Verma, D.: A sensor placement algorithm for redundant covering based on riesz energy minimization. Proc. ISCAS (2007)
19. Wang, Y.C., Tseng, Y.C.: Distributed deployment schemes for mobile wireless sensor networks to ensure multilevel coverage. IEEE Transactions on Parallel and Distributed System **19** (2008)
20. Johnson, M., Sarioz, D., Bar-Noy, A., Brown, T., Verma, D., Wu, C.: More is more: the benefits of denser sensor deployment. Proc. INFOCOM (2009)
21. Wang, G., Cao, G., Porta, T.L., Zhang, W.: Sensor relocation in mobile sensor networks. Proc. of IEEE INFOCOM (2005)
22. Pattem, S., Poduri, S., Krishnamachari, B.: Energy-quality tradeoffs for target tracking in wireless sensor networks. Proc. of ACM International Conference on Information Processing in Sensor Networks (IPSN), Springer Lecture Notes in Computer Science **2634** (2003)
23. Ma, K., Zhang, Y., Trappe, W.: Managing the mobility of a mobile sensor network using network dynamics. IEEE Transaction on Parallel and Distributed Systems **19**(1) (2008) 106–120
24. Anastasi, G., Conti, M., Falchi, A., Gregori, E., Passarella, A.: Performance measurements of mote sensor networks. (Proc. of ACM MSWiM 2004)