Efficient Placement and Dispatch of Sensors in a Wireless Sensor Network

You-Chiun Wang, Chun-Chi Hu, and Yu-Chee Tseng

Abstract—Sensor deployment is a critical issue because it affects the cost and detection capability of a wireless sensor network. In this work, we consider two related deployment problems: *sensor placement* and *sensor dispatch*. The former asks how to place the least number of sensors in a field to achieve sensing coverage and network connectivity, and the latter asks how to determine from a set of mobile sensors a subset of sensors to be moved to an area of interest with certain objective functions such that the coverage and connectivity properties are satisfied. This work is targeted toward planned deployment. Our solution to the placement problem allows an arbitrary-shaped sensing field possibly with arbitrary-shaped obstacles and an arbitrary relationship between the communication distance and sensing distance of sensors and, thus, significantly relaxes the limitations of existing results. Our solutions to the dispatch problem include a centralized one and a distributed one. The centralized one is based on adopting the former placement results and converting the problem to the *maximum-weight maximum-matching problem* with the objective of minimizing the total energy consumption to move sensors or maximizing the average remaining energy of sensors after movement. Designed in a similar way, the distributed one allows sensors to determine their moving directions in an autonomous manner.

Index Terms—connectivity, coverage, deployment, mobile sensors, network management, topology control, wireless sensor networks.

1 Introduction

The emerging wireless sensor networks provide an inexpensive and powerful means to monitor the physical environment. Such a network is composed of many tiny low-power nodes, each consisting of actuators, sensing devices, a wireless transceiver, and possibly a mobilizer [1]. These sensor nodes are massively deployed in a region of interest to gather and process environmental information. Wireless sensor networks have applications in agricultural production [2], traffic management [3], emergency navigation [4], and surveillance [5].

How to deploy sensors is a critical issue because it affects the cost and detection capability of a wireless sensor network. This work investigates the sensor deployment problem. We target at planned deployment in environments such as buildings or known fields. We address two related problems: *sensor placement* and *sensor dispatch*. The placement problem asks how to place the least number of sensors in a field to achieve desired *coverage* and *connectivity* properties [6]–[8], where coverage is to guarantee that every location in the sensing field is monitored by at least one sensor, and connectivity is to ensure that there are sufficient routing paths between sensors. Note that coverage is affected by a sensor's sensitivity, whereas connectivity is decided by a sensor's communication range. The dispatch problem assumes that sensors are mobilized and the goal is, given a set of mobile sensors and an area of interest inside the sensing field, to choose a subset of sensors to be delegated to the area of interest with certain objective functions such that coverage and connectivity properties are satisfied.

In the literature, the *art gallery problem* [9], [10] also aims to use the minimum number of guards to watch a polygon area. However, it is assumed that a guard can watch any point as long as line-of-sight exists, and it does not address the communication issue between guards. Several studies [11]–[13] model a sensing field as grid points and discuss how to place sensors on some grid points to satisfy certain coverage requirements. Reference [11] discusses how to place two types of sensors with different costs and sensing ranges such that every grid point is covered by sensors, and the total cost is minimized. The work in [12] considers a probabilistic sensing model and discusses how to place sensors in a field possibly with obstacles such that every grid point is covered with a minimum confidence level. In [13], the objective is to place sensors to ensure that every grid point is covered by different sensors; the result is to distinguish from different grid points for localization applications. Using grid approximation may cause high computation cost. Also, these works do not address the relationship between a sensor's communication distance r_c and sensing distance r_s. The work in [14] suggests placing sensors strip by strip, but it only addresses the $r_c = r_s$ case and does not consider the existence of obstacles. Sensor deployment has also been discussed in the area of robotics [15], [16]. With robots, sensors can be deployed one by one, and the result can be applied to an unknown environment. Some works [7], [8], [17], [18] address the coverage and connectivity issue by assuming that there is redundancy in the initial deployment, and the goal is to alternate sensors between sleep and active modes to reduce energy consumption while maintaining a full coverage of the sensing field.

Mobilizers have been assumed in several studies. References [19]–[21] discuss how to move sensors to enhance the coverage of the sensing field by using the Voronoi diagram or attractive/repulsive forces between sensors. The works in [22], [23] partition the sensing field into grids and move sensors from high-density grids to low-density ones to construct a uniform topology. The work in [24] suggests moving some sensors to make the network biconnected. In [25], it discusses how to move sensors to some locations (such as where events happen) while still maintaining complete coverage of the sens-

The authors are with the Department of Computer Science, National Chiao-Tung University, Hsin-Chu, 30010, Taiwan.
E-mail: {wangyc, cchu, yctseng}@cs.nctu.edu.tw

ing field. As can be seen, the goals of existing works are quite different from the dispatch problem defined in this work. In fact, the works in [5], [26], [27] have proposed their design and implementation of *mobile sensors*. Such mobile platforms are controlled by embedded computers and mounted with sensors. These works do motivate us to investigate the dispatch problem.

In this work, we propose more general solutions to the sensor placement problem than existing results. Our approach allows an arbitrary relationship between a sensor's communication distance and its sensing distance. The sensing field is assumed to be a polygon of any shape in which there may be arbitrary-shaped obstacles. Therefore, the results can model an indoor environment. Our approach first partitions the sensing field into smaller subregions. In each subregion, we arrange sensors row by row such that each row guarantees continuous coverage and connectivity and that adjacent rows ensure continuous coverage. Finally, columns of sensors are added to ensure connectivity between rows. The result requires fewer sensors compared to other schemes. For the sensor dispatch problem, we have proposed a centralized and a distributed scheme based on the former placement results. Both schemes attempt to minimize the total energy consumption to move sensors or to maximize the average remaining energy of those sensors that are moved into the area of interest. The first scheme converts the dispatch problem to the maximum-weight maximum-matching problem, whose optimal solution can be found in polynomial time. With a greedy strategy, the second scheme is distributed in that sensors will select the most suitable locations as their destinations and compete with each other to move to these locations.

The rest of this paper is organized as follows: Section 2 formally defines the sensor placement and dispatch problems. Sections 3 and 4 propose our solutions to these problems. Simulation results are presented in Section 5. Conclusions are drawn in Section 6.

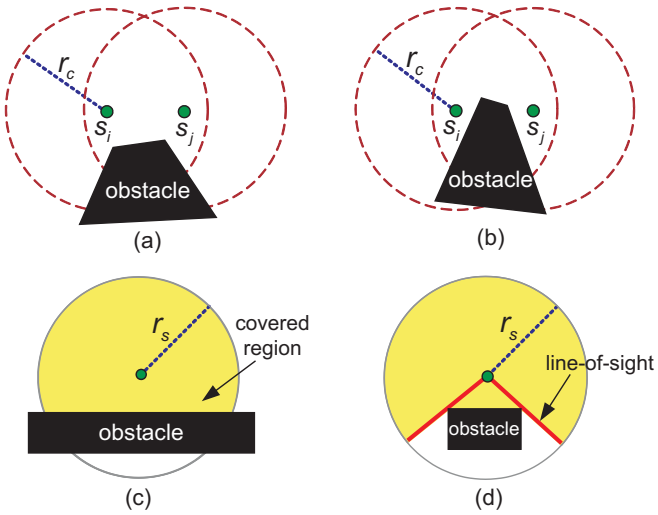

Fig. 1: Assumptions on connectivity and coverage. (a) s_i and s_j are connected. (b) The obstacle disconnects s_i and s_j. (c) Coverage with a large obstacle. (d) Coverage with a small obstacle.

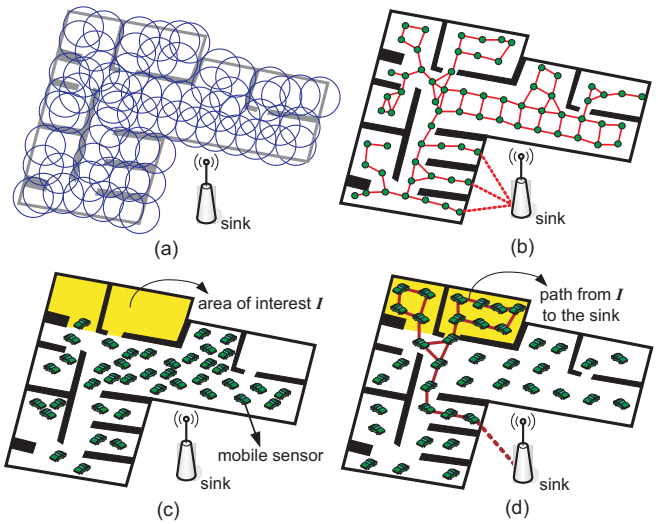

Fig. 2: An example of sensor deployment in an office environment. (a) Sensing coverage. (b) Network connectivity. (c) Sensor dispatch before dispatch. (d) Sensor dispatch after dispatch.

2 Problem Definitions

2.1 The Sensor Placement Problem

We are given a sensing field $\mathcal{A}$ to be deployed with sensors. Each sensor has a communication distance r_c and a sensing distance r_s. Sensors are homogenous, but we allow an arbitrary relationship of r_c and r_s. The sensing field $\mathcal{A}$ is modeled by an arbitrary 2D polygon. Obstacles may exist inside $\mathcal{A}$, which are also modeled by polygons of arbitrary shapes. However, obstacles do not partition $\mathcal{A}$ (otherwise, maintaining network connectivity would not be possible). With the presence of obstacles, we define two sensors s_i and s_j to be *connected* if $|\overline{s_i s_j}| \leq r_c$ and the line segment $\overline{s_i s_j}$ does not intersect any obstacle or boundary of $\mathcal{A}$; otherwise, they are *disconnected*. Fig. 1(a) and (b) show two examples. Obstacles may also reduce the coverage of a sensor. We assume that a point can be monitored by a sensor if it is within a distance of r_s and line of sight exists with the existence of obstacles. Fig. 1(c) and (d) give two examples. Note that, here, we adopt the *binary* sensing model [14], [19] of sensors, where a location can be either monitored or not monitored by a sensor. In Section 3.4, we will discuss how to adjust our placement solution to adapt to the *probabilistic* sensing model [12], [28], [29], where a location will be monitored by a sensor with some probability function.

Our objective is to place sensors in $\mathcal{A}$ to ensure both *sensing coverage* (in the sense that no point in $\mathcal{A}$ is unmonitored) and *network connectivity* (in the sense that no sensor gets disconnected) using as few sensors as possible. The concepts of coverage and connectivity in an office environment are illustrated in Fig. 2(a) and (b). Note that we assume $r_c = r_s$ in this example.

2.2 The Sensor Dispatch Problem

We are given a sensing field $\mathcal{A}$, an area of interest $\mathcal{I}$ inside $\mathcal{A}$, and a set of mobile sensors $\mathcal{S}$ resident in $\mathcal{A}$. The *sensor dispatch problem* asks how to find a subset $\mathcal{S}' \subseteq \mathcal{S}$ of sensors to be moved to $\mathcal{I}$ such that after the deployment, $\mathcal{I}$ satisfies our coverage and connectivity requirements, and the movement cost satisfies some objective function. Here, we consider two functions. The first one is to minimize the total energy consumption to move sensors, that is,

$$\min \sum_{i \in \mathcal{S}'} \Delta_m \times d_i, \tag{1}$$

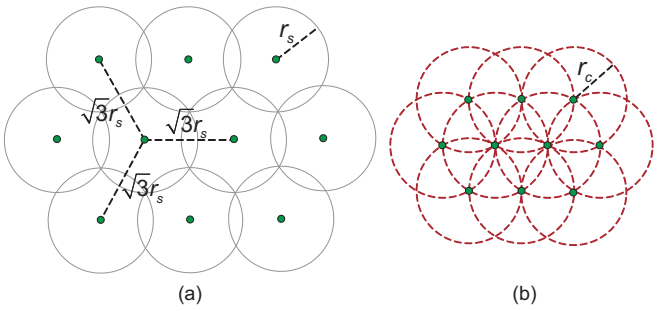

Fig. 3: Two possible sensor placements: (a) coverage-first placement and (b) connectivity-first placement.

where Δ_m is the unit energy cost to move a sensor in one step and d_i is the distance that sensor i has to be moved. The second one is to maximize the average remaining energy of sensors after the movement, that is,

$$\max \frac{\sum_{i \in \mathcal{S}'} (e_i - \Delta_m \times d_i)}{|\mathcal{S}'|}, \quad (2)$$

where e_i is the initial energy of sensor i. Note that the calculation of d_i should take the existence of obstacles into account. Fig. 2(c) and (d) illustrate the concept of sensor dispatch.

3 Solutions to the Sensor Placement Problem

To start with, we first consider two possible placements. The first one tries to reduce the number of sensors by minimizing the overlapping coverage. The result would be as shown in Fig. 3(a), where neighboring sensors are evenly separated by a distance of $\sqrt{3}r_s$. This scheme is efficient when $r_c \geq \sqrt{3}r_s$ since connectivity is automatically guaranteed. However, when $r_c < \sqrt{3}r_s$, extra sensors have to be added to maintain connectivity. It is inefficient because all of the sensing field has been covered and these newly added sensors will not make any contribution to coverage. The second possible placement is to meet the connectivity requirement first. This placement would be as shown in Fig. 3(b), where neighboring sensors are evenly separated by a distance of r_c. This scheme is efficient when $r_c \leq \sqrt{3}r_s$ because coverage is automatically guaranteed. However, when $r_c > \sqrt{3}r_s$, extra sensors have to be added to maintain coverage. It is inefficient because the overlapping coverage could be large.

Our placement has the following features: First, it avoids the dilemma in the above placements by taking both r_c and r_s into account. Second, our solution is more general as it allows an arbitrary shape of sensing field $\mathcal{A}$ and possibly obstacles in $\mathcal{A}$. Our scheme works in two steps. First, it partitions $\mathcal{A}$ into a number of regions. Regions are classified into *single-row regions* and *multi-row regions*. A single-row region is a belt-like area with width no larger than $\sqrt{3}r_{\min}$, where $r_{\min} = \min\{r_c, r_s\}$, so a row of sensors is sufficient to fully cover the region while maintaining connectivity. A multi-row region is perceivably larger and can be covered by several rows of sensors. Fig. 4 gives an example where the sensing field is partitioned into eight single-row regions and six multi-row regions.

3.1 Partitioning the Sensing Field

Algorithm 1 gives the pseudo code of our partition algorithm. The idea is to first identify all single-row regions. After excluding single-row regions, the remaining regions are multi-row regions.

To identify single-row regions, we expand the boundaries of $\mathcal{A}$ *inward* and perimeters of obstacles *outward* by a distance of $\sqrt{3}r_{\min}$. If there is a single-row region between one obstacle and $\mathcal{A}$'s boundary line segment $\overline{uv}$, the expanded parallel line $\overline{u'v'}$ must cut off a partial region, say $\mathcal{O}$, of the obstacle or $\overline{\mathcal{A}}$ (the area outside $\mathcal{A}$). Then, we can take a projection from $\mathcal{O}$ to $\overline{uv}$ to obtain the single-row region. Fig. 4(a) shows how to find single-row regions for the boundary, where the dotted lines are the expanded parallel lines of $\mathcal{A}$'s boundaries. After taking projections, we can obtain six single-row regions, a, b, d, e, f, and h, in Fig. 4(b). Then, we can perform the same steps for each obstacle. Note that a single-row region obtained from one obstacle may have overlapping with those obtained earlier (due to different projections). In this case, we can simply merge those with overlappings into one single-row region. This guarantees that our partition algorithm will produce a unique output. Fig. 4(b) shows all obtained single-row regions.

The aforementioned step may obtain several single-row regions. Excluding such regions, the remaining areas of $\mathcal{A}$ are multi-row regions. An example is given in Fig. 4(c). Note that there could be still obstacles inside a multi-row region (for example, the region 6).

Algorithm 1: Partition

Input: $\mathcal{A}$: sensing field
$\mathcal{B}$: $\mathcal{A}$'s boundaries and obstacles' perimeters
Output: single-row and multi-row regions

foreach $\overline{uv} \in \mathcal{B}$ **do** /* find all single-row regions */
  expand a parallel line $\overline{u'v'}$ by a distance of $\sqrt{3}r_{\min}$;
  if $\overline{u'v'}$ *cuts off a partial region* $\mathcal{O}$ *of an obstacle* **then**
    take a project $\mathcal{P}$ from $\mathcal{O}$ to $\overline{uv}$;
    if $\mathcal{P}$ *overlaps an existing single-row region* $\mathcal{P}'$ **then**
      merge $\mathcal{P}$ and $\mathcal{P}'$ into one single-row region;
    else
      make $\mathcal{P}$ a new single-row region;
end
exclude all single-row regions from $\mathcal{A}$ and the rest of the regions are multi-row regions;

3.2 Placing Sensors in Single-Row Regions

For a single-row region, we can find its bisector and then place a sequence of sensors along the bisector to satisfy both coverage and connectivity. A bisector can be found by doing a triangulation on that region, as shown in Fig. 5, and then connecting the midpoints of all dotted lines. Following the bisector, we can place a sequence of sensors each separated by a distance of $r_{\min}$ to ensure coverage and connectivity of that region, as shown in Fig. 5. Note that we always add an extra sensor at the end of the bisector for ensuring connectivity to neighboring regions.

3.3 Placing Sensors in Multi-Row Regions

Multiple rows of sensors will be placed in such regions. Below, we first consider a simple 2D plane without boundaries and obstacles. Then, we extend our result to an environment with boundaries and obstacles. Finally, we discuss the property of network connectivity in our placement scheme.

3.3.1 A Simple 2D Plane

Given a 2D plane without boundaries and obstacles, we will place sensors row by row. The basic idea is to form a row of

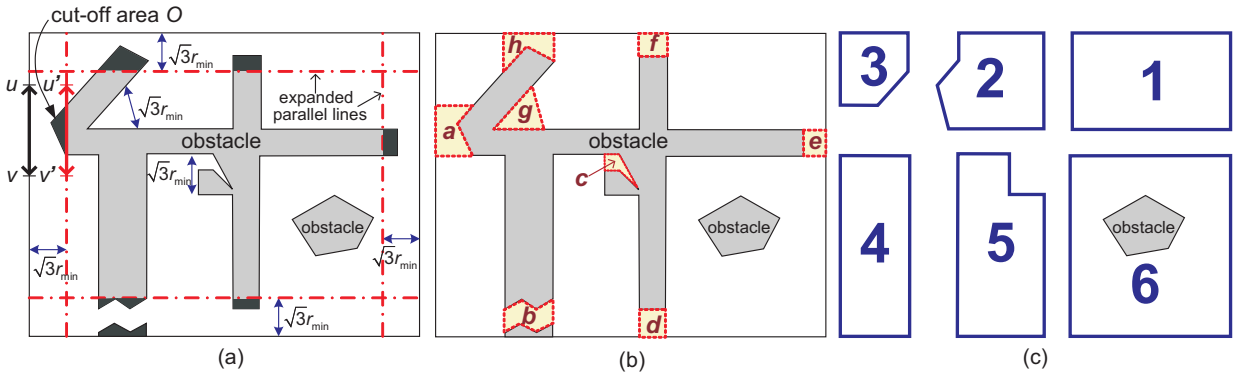

Fig. 4: Partitioning a sensing field. (a) A sensing field with obstacles. (b) Single-row regions. (c) Multi-row regions.

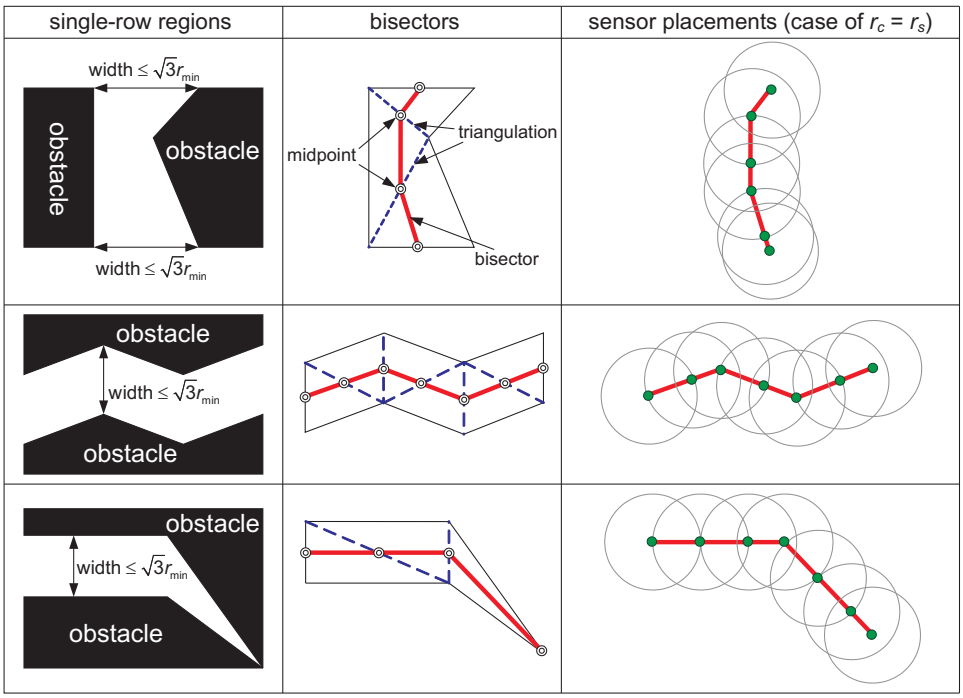

Fig. 5: Finding bisectors of single-row regions and their sensor placements.

sensors that can guarantee continuous coverage and connectivity. Adjacent rows should guarantee continuous coverage of the area. Finally, we may add some columns of sensors between adjacent rows, if necessary, to ensure connectivity. Based on the relationship of r_c and r_s, we separate the discussion into two cases.

Case 1: $r_c < \sqrt{3}r_s$. In this case, sensors on each row are separated by a distance of r_c, so the connectivity of each row can be guaranteed. Since $r_c < \sqrt{3}r_s$, each row of sensors can cover a belt-like area with a width of 2δ, where $\delta = \sqrt{r_s^2 - \frac{r_c^2}{4}}$. Adjacent rows will be separated by a distance of $r_s + \delta$ and shifted by a distance of $\frac{r_c}{2}$. With such an arrangement, the coverage of the whole area is guaranteed. Fig. 6(a)—(c) show three possible subcases. Note that, in this case, we have to add a column of sensors between two adjacent rows, each separated by a distance no larger than r_c, to connect them.

Case 2: $r_c \geq \sqrt{3}r_s$. In this case, the previous approach will waste a lot of sensors because a small r_s will cause two rows to be very close. Therefore, when $r_c \geq \sqrt{3}r_s$, we propose to place sensors in a typical triangular-latticed manner such that adjacent sensors are regularly separated by a distance of $\sqrt{3}r_s$, as shown in Fig. 6(d). Both coverage and connectivity properties are ensured.

3.3.2 *Multi-Row Regions with Boundaries and Obstacles*

Next, we modify the above solution for placing sensors in a region with boundaries and obstacles. Observe that, in our solution, sensors are placed with regular patterns. Thus, it can be transformed into an incremental approach, where sensors are added into the field one by one. In Table 1, we list the coordinates of a sensor's six neighbors. We can place the first sensor in any location of the region. From the first sensor, the six locations that can potentially be added with sensors are determined. These locations are inserted into a queue $\mathcal{Q}$. We then enter a loop in which each time an entry (x, y) is dequeued from $\mathcal{Q}$. If (x, y) is not inside any obstacle and not outside the multi-row region, a sensor will be placed in (x, y). Also, the six neighboring locations of (x, y) in Table 1 are inserted into $\mathcal{Q}$ if they have not been checked before. This process is repeated until $\mathcal{Q}$ becomes empty.

There are three minor issues left in the above solution. First, some areas near the boundaries or obstacles may be uncovered. Second, when $r_c < \sqrt{3}r_s$, we need to add extra

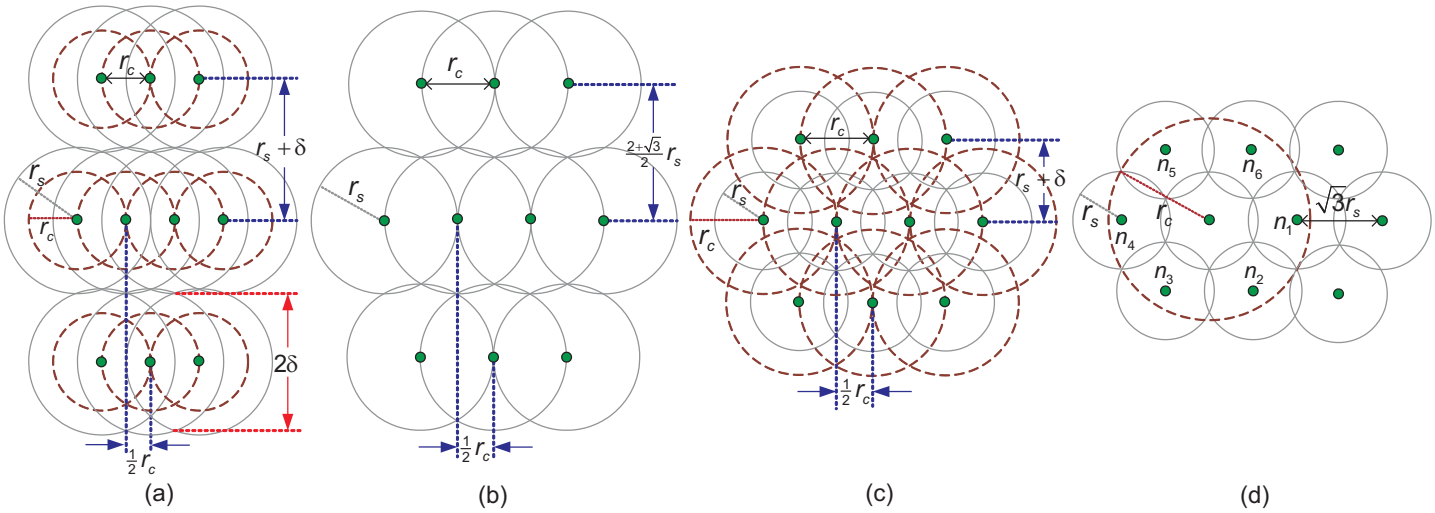

Fig. 6: Placing sensors in a simple 2D plane: (a) case of $r_c < r_s$, (b) case of $r_c = r_s$, (c) case of $r_s < r_c < \sqrt{3}r_s$, and (d) case of $r_c \geq \sqrt{3}r_s$.

neighbor	$r_c < \sqrt{3}r_s$	$r_c \geq \sqrt{3}r_s$
n_1	$(x + r_c, y)$	$(x + \sqrt{3}r_s, y)$
n_2	$(x + \frac{r_c}{2}, y - (r_s + \delta))$	$(x + \frac{\sqrt{3}r_s}{2}, y - \frac{3r_s}{2})$
n_3	$(x - \frac{r_c}{2}, y - (r_s + \delta))$	$(x - \frac{\sqrt{3}r_s}{2}, y - \frac{3r_s}{2})$
n_4	$(x - r_c, y)$	$(x - \sqrt{3}r_s, y)$
n_5	$(x - \frac{r_c}{2}, y + (r_s + \delta))$	$(x - \frac{\sqrt{3}r_s}{2}, y + \frac{3r_s}{2})$
n_6	$(x + \frac{r_c}{2}, y + (r_s + \delta))$	$(x + \frac{\sqrt{3}r_s}{2}, y + \frac{3r_s}{2})$

TABLE 1: Coordinates of the six neighbors of a sensor at location (x, y) in a multi-row region.

sensors between adjacent rows to maintain connectivity. Third, connectivity to neighboring regions needs to be maintained. Fig. 7(a) presents an example. These problems can be solved by sequentially placing sensors along the boundaries of the multi-row region and the perimeters of obstacles, as shown in Fig. 7(b). There are two cases to be considered. When $r_c < \sqrt{3}r_s$, since the maximum width of the uncovered area does not exceed r_c, the sensors should be separated by a distance of r_c. When $r_c \geq \sqrt{3}r_s$, the maximum width of the uncovered area does not exceed $\sqrt{3}r_s$, so the sensors should be separated by a distance of $\sqrt{3}r_s$. Since $r_c \geq \sqrt{3}r_s$, the connectivity between these extra sensors and the regularly placed sensors are guaranteed.

Note that we can save sensors in the last step by carefully selecting the first sensor's position in each multi-row region. In particular, for each multi-row region, we can place the first sensor near its longest boundary with a distance of δ if $r_c < \sqrt{3}r_s$ and a distance of $\frac{r_s}{2}$ otherwise. This will make the first row of sensors fully cover the longest boundary of the multi-row region, and thus, we do not have to add extra sensors in the last step. In addition, if the distance between a row of sensors and a boundary of the multi-row region (or an obstacle) is no larger than δ when $r_c < \sqrt{3}r_s$ and no larger than $\frac{r_s}{2}$ when $r_c \geq \sqrt{3}r_s$, we can also skip the last step. For example, some boundaries in Fig. 7 are not added with extra sensors.

3.3.3 *Network Connectivity in a Multi-Row Region*

Here, we discuss the property of network connectivity in our placement scheme. There are two cases to be discussed. When $r_c \geq \sqrt{3}r_s$, because a sensor will directly connect to its six neighbors (refer to Fig. 6(d)), the network is guaranteed to be at least 6-connected. This means that the network will not be partitioned unless there are more than five sensors broken.

On the other hand, when $r_c < \sqrt{3}r_s$, if both ends of each row are connected with sensors (refer to Fig. 7(b)), the network is guaranteed to be at least 2-connected. To improve the network connectivity, we can add several columns of sensors, each evenly separated and connecting rows together. With this, not only the network connectivity is improved but also the lengths of routing paths are reduced. For example, in Fig. 7(b), two additional columns of sensors are added on the top part. The original hop count between sensors a and b is thus reduced from eight to five with the help of additional columns. In Section 5, we will further address this issue through simulations.

3.4 Adapting to the Probabilistic Sensing Model

Up to now, our placement solution is based on the assumption of binary sensing model. In some cases, however, the detection probability of a sensor will decay with the distance from the sensor to the object. For example, references [12], [29] suggest that the detection probability of a location u by a sensor s_i can be modeled by

$$p^u_{s_i} = \begin{cases} e^{-\alpha d(s_i,u)} & \text{if } d(s_i,u) \leq r_s \\ 0 & \text{otherwise,} \end{cases}$$

where α is a parameter representing the physical characteristics of the sensor and $d(s_i, u)$ is the distance between s_i and u. Thus, when an object located at u is within the sensing ranges of a set $\widehat{\mathcal{S}}$ of sensors, the detection probability can be evaluated as

$$p(u) = 1 - \prod_{s_i \in \widehat{\mathcal{S}}} (1 - p^u_{s_i}).$$

It can be observed that in our placement solutions, for any combination of r_c and r_s, there must exist a location that is covered by only one sensor and has a distance of r_s to the sensor. The detection probability for such a location is $e^{-\alpha r_s}$. Therefore, our placement solutions can guarantee a detection probability of at least $e^{-\alpha r_s}$ in any location of the sensing field. On the other hand, if we want to guarantee that every point in

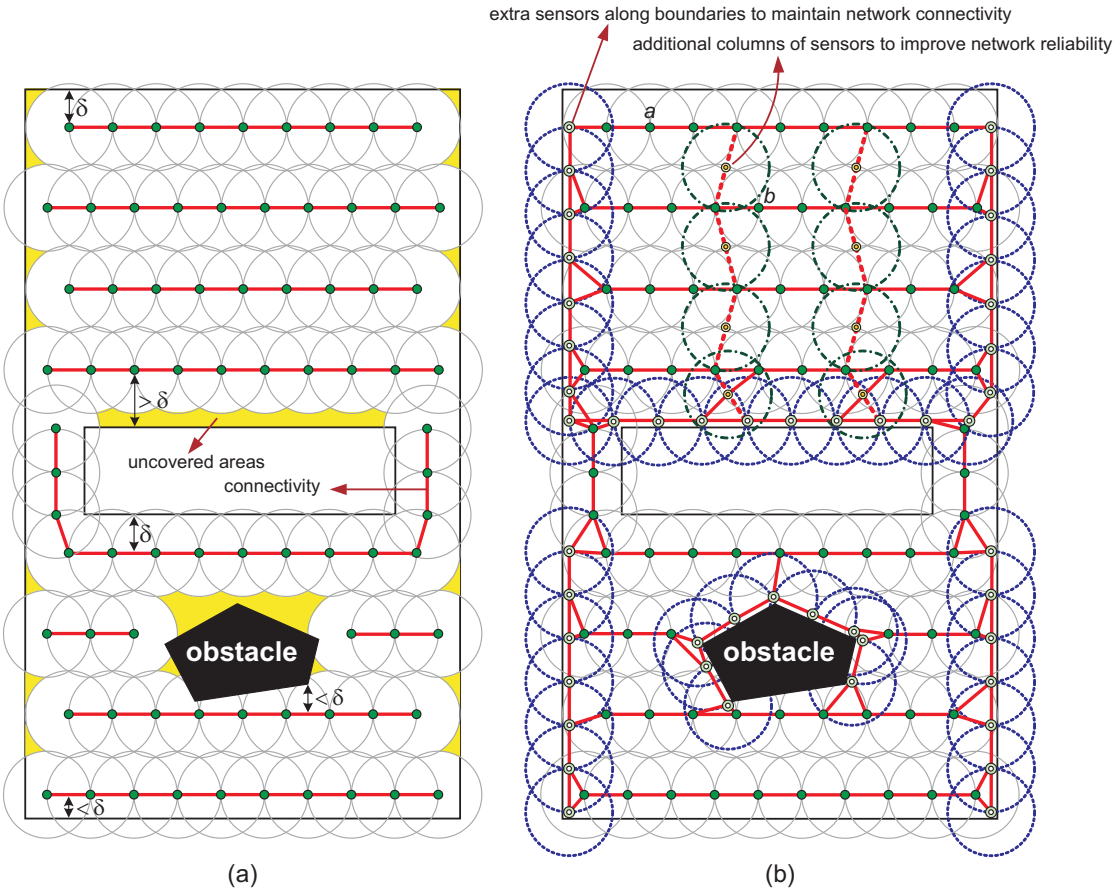

Fig. 7: Placing sensors along boundaries and around obstacles to fill uncovered areas and to maintain network connectivity. This example assumes that $r_c = r_s$. (a) Uncovered areas near the boundaries and obstacles. (b) Add sensors for coverage and connectivity.

the sensing field has a detection probability no smaller than a given threshold p_{th}, we can compute a virtual sensing distance r'_s by

$$e^{-\alpha r'_s} = p_{\text{th}} \Rightarrow r'_s = -\frac{\ln p_{\text{th}}}{\alpha}.$$

According to the above argument, if we replace r_s by r'_s when running our placement solutions, it is guaranteed that every point in the sensing field has a detection probability of at least p_{th}.

4 Solutions to the Sensor Dispatch Problem

Given a set of sensors already deployed in $\mathcal{A}$ and an area of interest $\mathcal{I}$ that has to be monitored intensively, the dispatch problem will be solved by the following steps:

1. Based on our placement results, we first compute the locations to be placed with sensors in $\mathcal{I}$ and then select some sensors to be moved to these locations.
2. In order to correctly report sensed data in $\mathcal{I}$ to the sink, we need to connect sensors in $\mathcal{I}$ and the sink. We then place a row of sensors, each separated by a distance of r_c, from $\mathcal{I}$ to the sink.
3. After dispatching sensors in steps 1 and 2, the remaining sensors can be deployed uniformly in the region of $\mathcal{A}-\mathcal{I}$ to ensure that the coverage of $\mathcal{A}-\mathcal{I}$ is not reduced too much.

We assume that there are sufficient sensors to satisfy the need of steps 1 and 2. Step 2 can be achieved easily. Step 3 can be done by applying the solutions using repulsive forces between sensors [19], [21] on $\mathcal{A} - \mathcal{I}$. As a result, we will only focus on the design of step 1 below. Fig. 2(c) and (d) give an example. In this section, two solutions are proposed. The centralized solution converts the dispatch problem to the maximum-weight maximum-matching problem, whereas the distributed solution is based on a greedy strategy.

4.1 A Centralized Dispatch Solution

Given a set $\mathcal{S}$ of sensors in $\mathcal{A}$ and an area of interest $\mathcal{I}$, our solution involves the following five steps:

1. Run the sensor placement algorithm in Section 3 on the area $\mathcal{I}$ to determine the locations in $\mathcal{I}$ to be placed with sensors. Let the set of locations be $\mathcal{L} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$. If $m \leq |\mathcal{S}|$, go to step 2; otherwise, we are short of sensors and the algorithm terminates.
2. For each sensor $s_i \in \mathcal{S}$, determine the energy cost $c(s_i, (x_j, y_j))$ to move s_i to each location (x_j, y_j), $j = 1 \cdots m$. We define $c(s_i, (x_j, y_j)) = \Delta_m \times d(s_i, (x_j, y_j))$, where $d(s_i, (x_j, y_j))$ is the shortest distance from s_i's current position to (x_j, y_j) considering the existence of obstacles. (How to compute the shortest distance will be discussed in Section 4.1.1.)
3. From $\mathcal{S}$ and $\mathcal{L}$, we construct a weighted complete bipartite graph $\mathcal{G} = (\mathcal{S} \cup \mathcal{L}, \mathcal{S} \times \mathcal{L})$ such that the vertex set contains $\mathcal{S}$ (all sensors) and $\mathcal{L}$ (all locations to be placed with sensors) and the edge set contains all edges from every element $s_i \in \mathcal{S}$ to every element $(x_j, y_j) \in \mathcal{L}$. The

weight of each edge $(s_i,(x_j,y_j))$ can be defined either as

$$w(s_i,(x_j,y_j)) = -c(s_i,(x_j,y_j)),$$

if Eq. (1) is the objective function, or as

$$w(s_i,(x_j,y_j)) = e_i - c(s_i,(x_j,y_j)),$$

if Eq. (2) is the objective function.

4. Solve the maximum-weight maximum-matching problem on graph $\mathcal{G}$. In particular, we construct a new graph $\widehat{\mathcal{G}} = (\mathcal{S} \cup \mathcal{L} \cup \widehat{\mathcal{L}}, \mathcal{S} \times \{\mathcal{L} \cup \widehat{\mathcal{L}}\})$ from $\mathcal{G}$, where $\widehat{\mathcal{L}}$ is a set of $|\mathcal{S}| - |\mathcal{L}|$ elements, each called a *virtual location*. The weight of each edge in $\widehat{\mathcal{G}}$ that also appears in $\mathcal{G}$ remains the same as that in $\mathcal{G}$, and the weight of each edge from $s_i \in \mathcal{S}$ to $(x_j,y_j) \in \widehat{\mathcal{L}}$ is set to $w_{\min}$, where

$$w_{\min} = \min_{s_i \in \mathcal{S},\ (x_j,y_j) \in \mathcal{L}} \{w(s_i,(x_j,y_j))\} - 1.$$

Intuitively, a virtual location is a dummy one. Its purpose is to make the two sets $\mathcal{S}$ and $\{\mathcal{L} \cup \widehat{\mathcal{L}}\}$ of the bipartite graph $\widehat{\mathcal{G}}$ to have equal sizes. This allows us to transform the problem to the *maximum-weight perfect-matching problem* on graph $\widehat{\mathcal{G}}$, whose purpose is to find a perfect matching $\mathcal{M}$ in $\widehat{\mathcal{G}}$ with the maximum total weights of edges in $\mathcal{M}$. (How to compute $\mathcal{M}$ will be discussed in Section 4.1.2.) Note that the value of $w_{\min}$ is set in such a way that selecting an edge incident to a virtual location has no impact to a solution to the maximum-weight perfect-matching problem.

5. For each edge $(s_i,(x_j,y_j))$ in $\mathcal{M}$ such that $(x_j,y_j) \notin \widehat{\mathcal{L}}$, we move sensor s_i to location (x_j,y_j) via the shortest path. However, if there is any edge $(s_i,(x_j,y_j)) \in \mathcal{M}$ such that $(x_j,y_j) \notin \widehat{\mathcal{L}}$ and $e_i - c(s_i,(x_j,y_j)) \leq 0$, it means that we do not have sufficient energy to move sensors to all locations in $\mathcal{L}$ because $\mathcal{M}$ is the optimal solution. Thus the algorithm terminates.

4.1.1 Computing the Shortest Distance $d(s_i,(x_j,y_j))$

Our goal is to find the shortest *collision-free* path from s_i's current position to (x_j,y_j), considering the existence of obstacles. Specifically, the movement of s_i should not collide with any obstacle. Several studies have addressed this issue [30]–[32]. Here we propose a modified approach of [31].

Considering its physical size, s_i is modeled as a circle with a radius r. Intuitively, s_i has a collision-free motion if its center always keeps at a distance of r or larger away from every obstacle and $\mathcal{A}$'s boundaries. This can be done by expanding the perimeters of all obstacles outwardly and $\mathcal{A}$'s boundaries inwardly by a distance of r and preventing s_i from moving into these expanded areas. The problem can be translated to one of finding a shortest path from s_i to (x_j,y_j) in a weighted graph $\mathcal{H} = (s_i \cup (x_j,y_j) \cup \mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ contains all vertices v of the polygons representing the expanded areas of obstacles and $\mathcal{A}$'s boundary such that v is not inside other expanded areas, and $\mathcal{E}$ contains all edges (u,v) such that $u,v \in \{s_i \cup (x_j,y_j) \cup \mathcal{V}\}$ and $\overline{uv}$ does not pass any expanded area of obstacles or $\mathcal{A}$. The weight of $(u,v) \in \mathcal{E}$ is length of $\overline{uv}$. Fig. 8 gives an example, where the double circles are vertices of $\mathcal{H}$. Nodes g and h are not vertices because they are inside obstacles 2's and 3's expanded areas, respectively. Edges (a,c), (a,d), (b,c), and $(b,d) \in \mathcal{E}$, but (b,e) and $(b,f) \notin \mathcal{E}$ because they pass the expanded area of obstacle 2.

4.1.2 Finding the Maximum-Weight Perfect-Matching $\mathcal{M}$

Recall that given the bipartite graph $\widehat{\mathcal{G}} = (\mathcal{S} \cup \mathcal{L} \cup \widehat{\mathcal{L}}, \mathcal{S} \times \{\mathcal{L} \cup \widehat{\mathcal{L}}\})$, the goal is to find a perfect matching $\mathcal{M}$ in $\widehat{\mathcal{G}}$ with the maximum total weights of edges in $\mathcal{M}$. In this section, we discuss how to use the Hungarian method [33] to solve this problem.

Definition 1. Given $\widehat{\mathcal{G}} = (\mathcal{S} \cup \mathcal{L} \cup \widehat{\mathcal{L}}, \mathcal{S} \times \{\mathcal{L} \cup \widehat{\mathcal{L}}\})$, a *feasible vertex labeling* of $\widehat{\mathcal{G}}$ is a real-valued function f on $\{\mathcal{S} \cup \mathcal{L} \cup \widehat{\mathcal{L}}\}$ such that for all $s_i \in \mathcal{S}$ and $(x_j,y_j) \in \{\mathcal{L} \cup \widehat{\mathcal{L}}\}$,

$$f(s_i) + f((x_j,y_j)) \geq w(s_i,(x_j,y_j)).$$

Definition 2. Given a feasible vertex labeling of $\widehat{\mathcal{G}}$, an *equality subgraph* $\widehat{\mathcal{G}}_f = (\mathcal{S} \cup \mathcal{L} \cup \widehat{\mathcal{L}}, \mathcal{E}_f)$ is the subgraph of $\widehat{\mathcal{G}}$ in which $\mathcal{E}_f$ contains all edges $(s_i,(x_j,y_j))$ in $\widehat{\mathcal{G}}$ such that

$$f(s_i) + f((x_j,y_j)) = w(s_i,(x_j,y_j)).$$

Theorem 1. Let f be a feasible vertex labeling of $\widehat{\mathcal{G}}$ and $\mathcal{M}$ be a perfect matching of $\widehat{\mathcal{G}}_f$, then $\mathcal{M}$ is a maximum-weight perfect matching of $\widehat{\mathcal{G}}$.

Proof: We show that no other perfect matching $\mathcal{M}'$ in $\widehat{\mathcal{G}}$ has a total weight larger than $\mathcal{M}$.

$$\begin{aligned} w(\mathcal{M}') &= \sum_{(s_i,(x_j,y_j)) \in \mathcal{M}'} w(s_i,(x_j,y_j)) \\ &\leq \sum_{(s_i,(x_j,y_j)) \in \mathcal{M}'} f(s_i) + f((x_j,y_j)) \\ &= \sum_{(s_i,(x_j,y_j)) \in \mathcal{M}} f(s_i) + f((x_j,y_j)) \\ &= \sum_{(s_i,(x_j,y_j)) \in \mathcal{M}} w(s_i,(x_j,y_j)) = w(\mathcal{M}), \end{aligned}$$

so $\mathcal{M}$ has the maximum total weights of edges. □

The Hungarian method is based on the observation from Theorem 1. It first assigns an arbitrary feasible vertex labeling for the graph $\widehat{\mathcal{G}}$, and then adjusts the labels of vertices until it can find a perfect matching $\mathcal{M}$ in the equality subgraph $\widehat{\mathcal{G}}_f$. One possible feasible vertex labeling is to set $f((x_j,y_j)) = 0$ for all $(x_j,y_j) \in \{\mathcal{L} \cup \widehat{\mathcal{L}}\}$ and to set $f(s_i)$ to the maximum of the weights of the edges adjacent to s_i for all $s_i \in \mathcal{S}$, that is,

$$f(s_i) = \max_{(x_j,y_j) \in \{\mathcal{L} \cup \widehat{\mathcal{L}}\}} \{w(s_i,(x_j,y_j))\}, \forall s_i \in \mathcal{S}.$$

The complete procedure of the Hungarian method is stated as follows:

1. Find a maximum matching $\mathcal{M}$ in $\widehat{\mathcal{G}}_f$. If $\mathcal{M}$ is perfect, we find out the solution and the method finishes. Otherwise, there must be an unmatched vertex $s_i \in \mathcal{S}$. We then assign two sets $\mathcal{X} = \{s_i\}$ and $\mathcal{Y} = \emptyset$.
2. In the graph $\widehat{\mathcal{G}}_f$, if $N_{\widehat{\mathcal{G}}_f}(\mathcal{X}) \neq \mathcal{Y}$, where $N_{\widehat{\mathcal{G}}_f}(\mathcal{X})$ is the set of vertices in $\{\mathcal{L} \cup \widehat{\mathcal{L}}\}$ that are adjacent to the vertices in $\mathcal{X}$, then go to step 3. Otherwise, we set

$$\beta = \min_{\substack{s_i \in \mathcal{X} \\ (x_j,y_j) \in \{\mathcal{L} \cup \widehat{\mathcal{L}}\} - \mathcal{Y}}} \{f(s_i) + f((x_j,y_j)) - w(s_i,(x_j,y_j))\},$$

and construct a new labeling f' for $\widehat{\mathcal{G}}$ by

$$f'(v) = \begin{cases} f(v) - \beta & \text{for } v \in \mathcal{X} \\ f(v) + \beta & \text{for } v \in \mathcal{Y} \\ f(v) & \text{otherwise.} \end{cases}$$

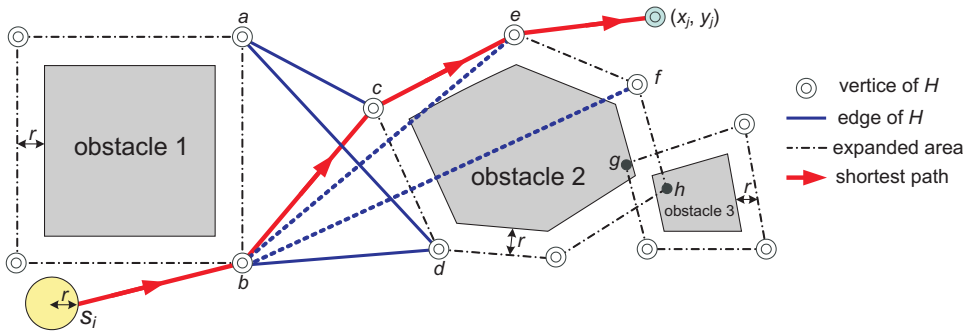

Fig. 8: Finding a collision-free path from s_i to (x_j, y_j). Note that not all edges of $\mathcal{H}$ are shown in the figure.

Then we replace f by f', reconstruct the equality subgraph $\widehat{\mathcal{G}}_{f'}$, and go to step 1. Note that we have to satisfy the conditions of $\beta > 0$ and $N_{\widehat{\mathcal{G}}_{f'}}(\mathcal{X}) \neq \mathcal{Y}$; otherwise, we need to reselect another β value that can satisfy the above conditions.

3. Choose a vertex (x_l, y_l) in $N_{\widehat{\mathcal{G}}_f}(\mathcal{X})$ but not in $\mathcal{Y}$. If (x_l, y_l) is matched with $s_k \in \mathcal{S}$ in $\mathcal{M}$, then we update $\mathcal{X} = \mathcal{X} \cup \{s_k\}$ and $\mathcal{Y} = \mathcal{Y} \cup \{(x_l, y_l)\}$, and go back to step 2.

Note that, each time when we relabel the graph $\widehat{\mathcal{G}}$, we may introduce new edges into the new equality subgraph $\widehat{\mathcal{G}}_f$ until all edges in $\widehat{\mathcal{G}}$ are included. Therefore, the Hungarian method can always find a perfect matching in $\widehat{\mathcal{G}}_f$ since $\widehat{\mathcal{G}}$ is a complete bipartite graph.

4.1.3 Time Complexity Analysis

Next, we analyze the time complexity of our sensor dispatch solution. Let $|\mathcal{S}| = n$, $|\mathcal{L}| = m$, and k be the number of vertices of the polygons of all obstacles and $\mathcal{A}$. In step 2, there are $O(nm)$ pairs of $(s_i, (x_j, y_j))$. To compute the energy cost of each pair, we construct a graph of $O(k)$ vertices. Finding the shortest path on such graph can use the Dijkstra's algorithm [34], which takes $O(k^2)$ time. Therefore, the total time complexity of step 2 is $O(mnk^2)$. The conversion in step 3 takes $O(nm)$ time. In step 4, constructing the graph $\widehat{\mathcal{G}}$ from $\mathcal{G}$ takes $O(n(n-m))$ time since it needs to add $n-m$ vertices and $n(n-m)$ edges. Running the Hungarian method on $\widehat{\mathcal{G}}$ has a time complexity of $O(n^3)$. Finally, it takes $O(n)$ time in step 5 to check all edges in $\mathcal{M}$. Therefore, the total time complexity is $O(mnk^2) + O(nm) + O(n(n-m)) + O(n^3) + O(n) = O(mnk^2 + n^3)$.

4.2 A Distributed Dispatch Solution

The aforementioned solution is optimal but centralized. Here we propose a distributed solution based on a greedy strategy. The solution involves the following steps:

1. The sink runs the placement algorithm in Section 3 on the area $\mathcal{I}$ to obtain a set of locations $\mathcal{L} = \{(x_1, y_1), \cdots, (x_m, y_m)\}$ to be occupied by sensors. The sink then broadcasts $\mathcal{L}$ to all sensors.
2. On receiving the table $\mathcal{L}$, a sensor will keep a copy of $\mathcal{L}$ and mark each location (x_j, y_j) as *unoccupied*, $j = 1 \cdots m$.
3. Each sensor s_i then chooses an unoccupied location (x_j, y_j) from $\mathcal{L}$ as its destination. The selection of (x_j, y_j) is dependent on our objective function.
   - If Eq. (1) is the objective function, s_i will choose the location (x_j, y_j) such that the moving distance $d(s_i, (x_j, y_j))$ is minimized as its destination.
   - If Eq. (2) is the objective function, s_i will choose the location (x_j, y_j) such that after moving to (x_j, y_j), its remaining energy is maximized.

   Sensor s_i will then start moving to (x_j, y_j) and mark (x_j, y_j) as occupied.
4. On s_i's way moving toward its destination, it will periodically broadcast the status of its table $\mathcal{L}$, its destination, and its cost to move to that destination. Note that the cost is based on which objective function is used. The above action can be controlled by setting a timer $T_{\text{broadcast}}$. On sensor s_k receiving s_i's broadcast, the following actions will be taken:
   - For all locations marked as occupied by s_i, s_k will also mark them as occupied.
   - If both s_i and s_k are moving toward the same destination, they will compete by their costs. The one with a lower cost will win and keep moving toward that destination. The one with a higher cost will give up moving toward that destination and go back to step 3 to reselect a new destination. (Note that in case that s_k has arrived at its destination, it will have a cost of zero, in which case, s_i will lose in the competition.)
5. Each sensor will repeat the above steps until it reaches its destination or loses to another sensor and finds that all locations in $\mathcal{L}$ have been marked as occupied. In the former case, the sensor will execute its monitoring job at the designated location. In the latter case, the sensor will continue to support the remaining steps 2 and 3 mentioned in the beginning of Section 4 (to connect $\mathcal{I}$ and the sink or to monitor the area $\mathcal{A} - \mathcal{I}$).

To prove the convergence of this distributed algorithm, we have to show that every location (x_j, y_j) in $\mathcal{L}$ can eventually be covered by one sensor. Step 4 guarantees that a sensor s_i will eventually arrive at the location (x_j, y_j) if it always wins the competition. If s_i loses the competition, it means that (x_j, y_j) has been committed by another sensor. In this case, s_i has to go back to step 3 to reselect another destination. A sensor will continue moving until it finds that all locations are marked as occupied. Therefore, as long as there are sufficient sensors, all locations will eventually be covered by sensors. It is possible that, without sufficient information, a sensor may keep on moving even if all locations in $\mathcal{L}$ are occupied. However, it

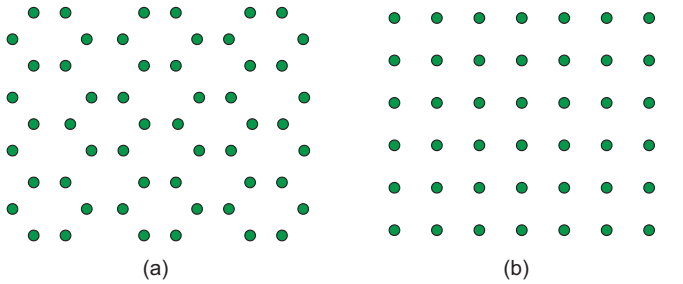

Fig. 10: Two common regular placement patterns: (a) hexagon placement and (b) square placement.

will eventually meet another sensor or reach the area $\mathcal{I}$ and be aware of the fact that $\mathcal{L}$ has been occupied (note that the network in $\mathcal{I}$ must be connected and, thus, sensors in $\mathcal{I}$ must have correct information). Therefore, the convergence of the algorithm is proved.

5 Experimental Results

In this section, we present some simulation results to verify the effectiveness of the proposed algorithms. The first experiment evaluates the number of sensors required to cover a sensing field. We design six types of sensing fields, as shown in Fig. 9. Sensors are assumed to have omnidirectional sensing capability (such as acoustic sensors). The communication distance r_c is set to 10 m (which is close to that specified in IEEE 802.15.4 [35] in an indoor environment). To reflect the relationships of $r_c < r_s$, $r_c = r_s$, $r_s < r_c < \sqrt{3}r_s$, and $r_c \geq \sqrt{3}r_s$, we set the sensing distance r_s to 12 m, 10 m, 7 m, and 5 m, respectively. We compare our result against the *coverage-first* and *connectivity-first* methods discussed in the beginning of Section 3 and two common regular placement patterns [36], *hexagon* and *square*, as shown in Fig. 10. To ensure both coverage and connectivity, we set the distance between adjacent sensors to $r_{\min}$ in the hexagon placement. In the square placement, the distance between adjacent sensors is set to r_c if $r_c < \sqrt{2}r_s$ and set to $\sqrt{2}r_s$ otherwise.

Fig. 11 shows the number of sensors required in different placement methods. As can be seen, our placement method uses the least number of sensors in all cases. The coverage-first method uses more sensors when $r_s < r_c < \sqrt{3}r_s$ because it needs many extra sensors to maintain connectivity between neighboring sensors. The connectivity-first method uses more sensors when $r_c \leq r_s$ because it is dominated by the value of r_c and the overlapping in coverage is large. The hexagon method uses more sensors when $r_c > r_s$ because the distance between adjacent sensors is limited to small r_s. The situation becomes worse as r_s becomes smaller. The square method uses more sensors when $r_c < \sqrt{2}r_s$ because it is dominated by the value of r_c and the overlapping in coverage could be also large. Note that when $r_c \geq \sqrt{3}r_s$, our method works the same as the coverage-first method in each individual region, so both schemes will use the same number of sensors.

When $r_c < \sqrt{3}r_s$, our placement scheme has the flexibility to reduce the routing paths of sensors by adding several columns of sensors. In the second experiment, we evaluate the effect of the number of additional columns on the all-pair shortest paths of sensors. In particular, we measure the average hop count of the shortest path between any two sensors in our placement. In this experiment, we set the sensing field as the one specified in Fig. 9(a), and we consider the case of $r_c = r_s$. Fig. 12 shows the results when r_c is set to 10 m, 5 m, and 3 m, where the number of sensors in each row will be 41, 81, and 161, respectively. In Fig. 12, we can observe that the average hop count of the all-pair shortest paths can be reduced when we add more columns of sensors. However, adding these columns will increase the total number of sensors to be placed in the sensing field. From Fig. 12, we can find that the suitable number of additional columns is around two to three when the number of sensors in each row is among 40 to 160.

The third experiment evaluates different dispatch schemes. The sensing field $\mathcal{A}$ is a 900 m $\times$ 900 m square. The region of interest $\mathcal{I}$ is a 300 m $\times$ 300 m square located at the center of $\mathcal{A}$. Sensors are randomly scattered over the region of $\mathcal{A} - \mathcal{I}$. With the setting of (r_c, r_s) = (28, 16), (23.5, 13.45), (21, 12), (19.5, 11.05), (17.5, 10.1), (16.5, 9.45), and (15.5, 8.9), we will need 150, 200, 250, 300, 350, 400, and 450 sensors, respectively, to be dispatched to $\mathcal{I}$, according to our placement algorithm. To fairly compare the centralized and the distributed schemes, the number of sensors is intentionally set to the required number of sensors in $\mathcal{I}$. The sensors' initial energies are randomly selected from $[1000, 1500]$ units, and we set the moving cost $\Delta_m = 1$ energy unit per meter. The speed of a sensor is set to 0.5 m/s. For our distributed dispatch method, we set the timer $T_{\text{broadcast}}$ as one second. For comparison, we also design a *random* method where we arbitrarily select a sensor to move to each location in a centralized manner. Fig. 13 shows the simulation results under a different number of sensors required in $\mathcal{I}$. In Fig. 13(a), we can observe that our centralized method (using Eq. (1) as the objective function) consumes the least energy compared to other methods. This is a result of our maximum-matching approach. The distributed method consumes more energy than the centralized method since our greedy strategy can make local decisions. The similar result can be observed from Fig. 13(b), where the centralized method (using Eq. (2) as the objective function) can achieve the highest average remaining energy of sensors in $\mathcal{I}$. Note that, under both objective functions, the random method always consumes the most energy even though sensors are selected in a centralized manner. This reflects the importance of the dispatch issue since blindly moving sensors will lead to shorten network lifetime.

With the same settings in the previous experiment, the last experiment evaluates the effect of the broadcast timer $T_{\text{broadcast}}$ on the number of broadcasts and average moving distance of a sensor when our distributed dispatch method is adopted. In this experiment, we use Eq. (1) as the objective function. Fig. 14 illustrates the simulation results when the numbers of sensors are 200 and 400. From Fig. 14, we can observe that, when $T_{\text{broadcast}}$ becomes larger, the number of broadcasts can be reduced. However, this will cause sensors to move longer distances and thus extend the convergence time of the distributed algorithm. In Fig. 14, we can find that the best value of $T_{\text{broadcast}}$ is around two since both the number of broadcasts and average moving distance can be kept quite small.

6 Conclusions

In this work, we have proposed systematical solutions for sensor placement and dispatch. Our solution allows a sensing field of shape as an arbitrary polygon with the possible existence of obstacles. Thus, the result can be used for an indoor environment. Our solution also allows an arbitrary

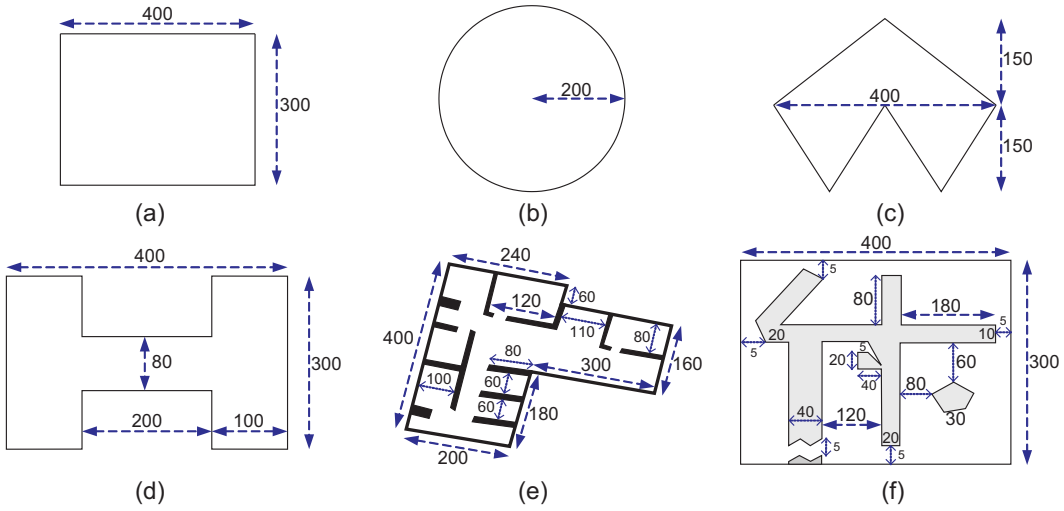

Fig. 9: Six types of sensing fields used in the simulations. The unit of length is meter. (a) Rectangle. (b) Circle. (c) Non-convex polygon. (d) H-shape. (e) Office example in Fig. 2. (f) Arbitrary shape in Fig. 4.

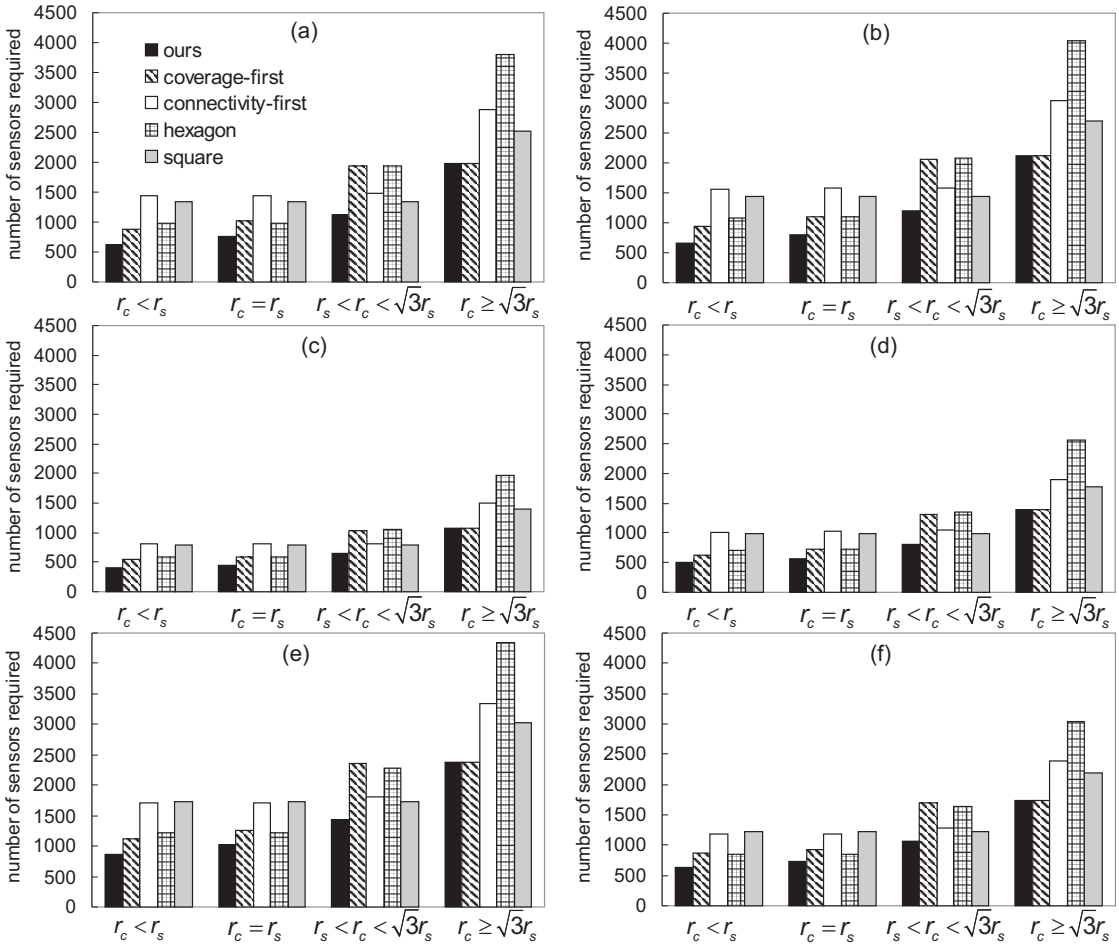

Fig. 11: Comparison of number of sensors required under different types of sensing fields.

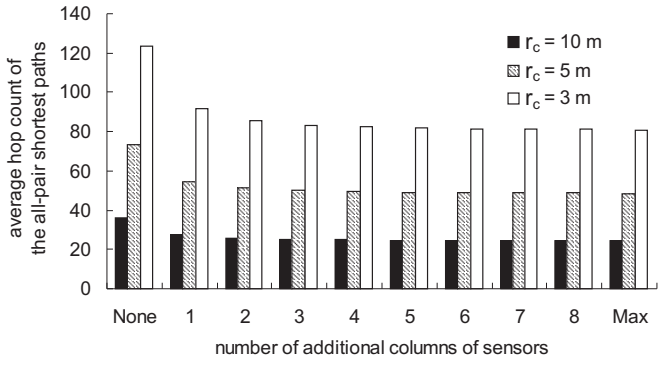

Fig. 12: Effect of additional columns on the all-pair shortest paths of sensors.

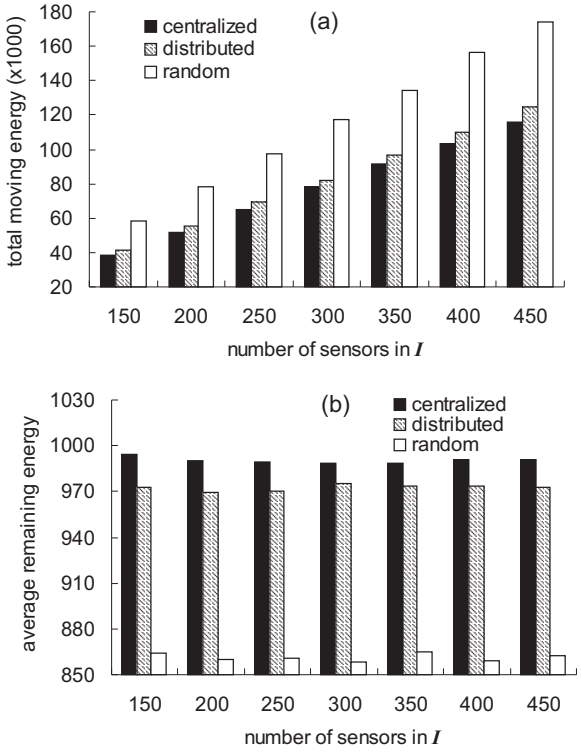

Fig. 13: Comparison of different dispatch methods: (a) the total energy consumption for movement when using Eq. (1) as the objective function, and (b) the average remaining energy of sensors when using Eq. (2) as the objective function.

relationship of sensors' communication distances and sensing distances. It is verified that the proposed schemes require fewer sensors to ensure full coverage of the sensing field and connectivity of the network as compared to other placement schemes in various types of sensing fields. A new sensor dispatch problem is defined and two energy-efficient dispatch algorithms are presented to move sensors to the target locations determined by our sensor placement scheme.

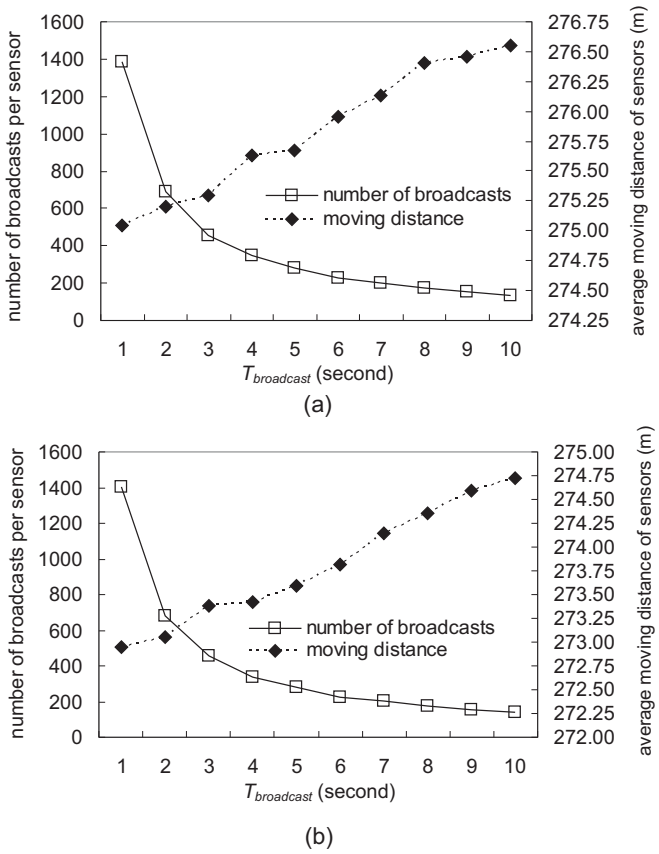

Fig. 14: Effect of $T_{\mathrm{broadcast}}$ on the number of broadcasts and average moving distance of a sensor.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Comm. Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[2] J. Burrell, T. Brooke, and R. Beckwith, "Vineyard computing: sensor networks in agricultural production," *IEEE Pervasive Computing*, vol. 3, no. 1, pp. 38–45, 2004.

[3] T.T. Hsieh, "Using sensor networks for highway and traffic applications," *IEEE Potentials*, vol. 23, no. 2, pp. 13–16, 2004.

[4] Y.C. Tseng, M.S. Pan, and Y.Y. Tsai, "Wireless sensor networks for emergency navigation," *IEEE Computer*, vol. 39, no. 7, pp. 55–62, 2006.

[5] Y.C. Tseng, Y.C. Wang, and K.Y. Cheng, "An integrated mobile surveillance and wireless sensor (iMouse) system and its detection delay analysis," *Proc. ACM Int'l Symp. Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 178–181, 2005.

[6] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M.B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," *Proc. IEEE INFOCOM*, pp. 1380–1387, 2001.

[7] D. Tian and N.D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," *Proc. ACM Int'l Workshop Wireless Sensor Networks and Applications*, pp. 32–41, 2002.

[8] H. Zhang and J.C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Int'l J. Wireless Ad Hoc and Sensor Networks*, vol. 1, no. 1–2, pp. 89–124, 2005.

[9] J. O'Rourke, *Art gallery theorems and algorithms*, Oxford Univ. Press, 1987.

[10] T.C. Shermer, "Recent results in art galleries," *Proc. IEEE*, vol. 80, no. 9, pp. 1384–1399, 1992.

[11] K. Chakrabarty, S.S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Trans. Computers*, vol. 51, no. 12, pp. 1448–1453, 2002.

[12] S.S. Dhillon and K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks," in *IEEE Wireless Comm. and Networking*, 2003, pp. 1609–1614.

[13] F.Y.S. Lin and P.L. Chiu, "A near-optimal sensor placement algorithm to achieve complete coverage/discrimination in sensor networks," *IEEE Comm. Letters*, vol. 9, no. 1, pp. 43–45, 2005.

[14] K. Kar and S. Banerjee, "Node placement for connected coverage in sensor networks," *Proc. Int'l Symp. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[15] E.U. Acar, H. Choset, and P.N. Atkar, "Complete sensor-based coverage with extended-range detectors: a hierarchical decomposition in terms of critical points and voronoi diagrams," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems*, pp. 1305–1311, 2001.

[16] A. Howard, M.J. Mataric, and G.S. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Autonomous Robots*, vol. 13, no. 2, pp. 113–126, 2002.

[17] A. Cerpa and D. Estrin, "ASCENT: adaptive self-configuring sensor networks topologies," *Proc. IEEE INFOCOM*, pp. 1278–1287, 2002.

[18] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang, "PEAS: a robust energy

conserving protocol for long-lived sensor networks," *Proc. Int'l Conf. Distributed Computing Systems*, pp. 28–37, 2003.
[19] N. Heo and P.K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Trans. Systems, Man and Cybernetics–Part A: Systems and Humans*, vol. 35, no. 1, pp. 78–92, 2005.
[20] G. Wang, G. Cao, and T.L. Porta, "Movement-assisted sensor deployment," *Proc. IEEE INFOCOM*, pp. 2469–2479, 2004.
[21] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," *Proc. IEEE INFOCOM*, pp. 1293–1303, 2003.
[22] G. Wang, G. Cao, T.L. Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," *Proc. IEEE INFOCOM*, pp. 2302–2312, 2005.
[23] J. Wu and S. Yang, "SMART: a scan-based movement-assisted sensor deployment method in wireless sensor networks," *Proc. IEEE INFOCOM*, pp. 2313–2324, 2005.
[24] P. Basu and J. Redi, "Movement control algorithms for realization of fault-tolerant ad hoc robot networks," *IEEE Network*, vol. 18, no. 4, pp. 36–44, 2004.
[25] Z. Butler and D. Rus, "Event-based motion control for mobile-sensor networks," *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 34–42, 2003.
[26] T.A. Dahlberg, A. Nasipuri, and C. Taylor, "Explorebots: a mobile network experimentation testbed," *Proc. ACM SIGCOMM Workshop Experimental Approaches to Wireless Network Design and Analysis*, pp. 76–81, 2005.
[27] D. Johnson, T. Stack, R. Fish, D.M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau, "Mobile Emulab: a robotic wireless and sensor network testbed," *Proc. IEEE INFOCOM*, 2006.
[28] T. Clouqueur, K.K. Saluja, and P.Ramanathan, "Fault tolerance in collaborative sensor networks for target detection," *IEEE Trans. Computers*, vol. 53, pp. 320–333, 2004.
[29] Y. Zou and K. Chakrabarty, "A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks," *IEEE Trans. Computers*, vol. 54, pp. 978–991, 2005.
[30] G.M. Dai, A.H. Du, Q.H. Li, and M.C. Wang, "Planning of moving path based on simplified terrain," *Proc. Int'l Conf. Machine Learning and Cybernetics*, pp. 1915–1918, 2003.
[31] Y.H. Liu and S. Arimoto, "Finding the shortest path of a disc among polygonal obstacles using a radius-independent graph," *IEEE Trans. Robots and Automation*, vol. 11, pp. 682–691, 1995.
[32] S.Q. Zheng, J.S. Lim, and S.S. Iyengar, "Finding obstacle-avoiding shortest paths using implicit connection graphs," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, pp. 103–110, 1996.
[33] H.W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.
[34] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms (second edition)*, The MIT Press, 2001.
[35] *IEEE Standard 802.15.4-2003, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 2003, IEEE.
[36] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T.H. Lai, "Deploying wireless sensors to achieve both coverage and connectivity," *Proc. ACM Int'l Symp. Mobile Ad Hoc Networking and Computing*, pp. 131–142, 2006.