Adaptive Triangular Deployment Algorithm for Unattended Mobile Sensor Networks

Ming Ma, *Student Member*, *IEEE*, and Yuanyuan Yang, *Senior Member*, *IEEE*

Abstract—In this paper, we present a novel sensor deployment algorithm, called the *adaptive triangular deployment (ATRI)* algorithm, for large-scale unattended mobile sensor networks. The ATRI algorithm aims at maximizing coverage area and minimizing coverage gaps and overlaps by adjusting the deployment layout of nodes close to equilateral triangulation, which is proven to be the optimal layout to provide the maximum no-gap coverage. The algorithm only needs the location information of nearby nodes, thereby avoiding communication cost for exchanging global information. By dividing the transmission range into six sectors, each node adjusts the relative distance to its one-hop neighbors in each sector separately. The *distance threshold strategy* and the *movement state diagram strategy* are adopted to avoid the oscillation of nodes. The simulation results show that the ATRI algorithm achieves a much larger coverage area and smaller average moving distance of nodes than existing algorithms. We also show that the ATRI algorithm is applicable to practical environments and tasks such as working in both bounded and unbounded areas and avoiding irregularly shaped obstacles. In addition, the density of nodes can be adjusted adaptively to different requirements of tasks.

Index Terms—Sensor deployment, sensor networks, mobile sensors, adaptive algorithms, self-organizing networks.

1 Introduction and Background

In recent years, advanced VLSI and radio frequency (RF) technologies have accelerated the development and applications of wireless sensor networks (WSNs). WSNs are playing an increasingly important role in a wide range of applications such as medical treatment, outer space exploration, battlefield surveillance, emergency response, and so forth [5], [6], [7], [8], [9]. A WSN is generally composed of hundreds and thousands of distributed mobile sensor nodes, with each node having limited and similar communication, computing, and sensing capabilities [4], [10]. Such sensor networks have many special characteristics. The resource-limited sensor nodes are usually thrown into an unknown environment without a preconfigured infrastructure. Before monitoring the environment, sensor nodes must be able to deploy themselves to the working area. At the same time, the sensor nodes organize themselves into a network and subscribe to the network [15]. Although sensor nodes are designed with low energy consumption in mind, they can survive for only a very limited lifetime with current technologies [10], [11], [12], [13]. Furthermore, the low computing capability and limited memory and bandwidth of the sensor nodes prohibit the use of high complexity algorithms and protocols. All of these special characteristics of sensor networks bring a lot of challenges to developing reliable and efficient algorithms and protocols for such networks.

Although most of the existing work on sensor networks is currently still at the laboratory level (see, for example, UCB-smart dusts [1], MIT-μAMPS [2], ISI-pc104 [3], and UCLA-WINS [10], [11]), it is expected that sensor network technologies will be applied widely in various areas in the near future [5], [6], [7], [8], [9]. One of the main functionalities of a sensor network is to take the place of the human being to fulfill the sensing or monitoring work in dangerous or human-unreachable environments such as the battlefield, outer space, volcanos, deserts, the seabed, and so forth [5], [6], [7], [8], [9]. These environments usually have the following similarities: large scale, unknown, and full of unpredictable events which may cause the sudden failure of sensors. Unlike in a well-known environment, it is impossible to throw sensor nodes to their expected targets in an unknown working area or provide a map of the working area to sensor nodes before the placement. However, most of the previous work in this area assumes that all nodes are well deployed or that a global map is prestored in the memory of all sensor nodes. In fact, in some situations, the environment may be completely unknown to the newly arrived sensor nodes. Without control from the human being, sensor nodes must be "smart" enough to learn the working area by themselves and deploy themselves to the expected working targets.

The rest of the paper is organized as follows: Section 2 introduces our design goals and assumptions. Section 3 briefly summarizes the existing work in this area. Section 4 studies the optimum node layout for sensor deployment. Section 5 presents the details of the new self-deployment algorithm for mobile sensor networks. Section 6 gives the simulation results for the proposed algorithm. Finally, Section 7 concludes the paper.

• *The authors are with the Department of Electrical and Computer Engineering, 215 Light Engineering Building, State University of New York at Stony Brook, Stony Brook, NY 11794-2350.*
E-mail: {mingma, yang}@ece.sunysb.edu.

Manuscript received 3 Nov. 2005; revised 21 Oct. 2006; accepted 12 Mar. 2007; published online 10 Apr. 2007.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0392-1105.
Digital Object Identifier no. 10.1109/TC.2007.1054.

2 Design Goals and Assumptions

In this section, we first discuss our design goals. Let us imagine that a bunch of sensor nodes are thrown onto Mars by an aircraft to monitor the surface temperature of a specific area. The first task that the sensors should perform is to explore the working area and deploy themselves to cover the area. If the environment is known in advance, then a map can be stored into the memory of each sensor node and the movement of all the nodes can be well planned so that each of them can move to its target directly without any extra delay and energy consumption. However, sensor networks are more likely to be used in a dangerous or unknown environment to replace human activities, where the environment information and the distribution of interested events are not available. Thus, a sensor node should be able to learn the environment, decide its own expected target, and deploy itself to the target.

First, in such a large-scale unknown environment, distributed deployment algorithms are preferred to centralized algorithms. Even though the movement, location, and coverage of all of the nodes in the network could be planned and scheduled by the human being or a remote controller, long communication delay and limited bandwidth may prevent human-control-based protocols from being used in delay-sensitive and remote exploring applications. For instance, in the recent Mars Rover Project [14] developed by NASA, it takes about 12 minutes to download a small picture from the Mars Rover to the earth. Due to the long distance between the sensors and the controller and the low speed of the wireless channel, the commands can hardly be transmitted to the sensors. Even though a powerful controller is placed at the center of the working area in order to make accurate decisions without the GPS [17], a large amount of location information and global environment information has to be fed back to the controller. During the deployment period, frequent movement, unpredictable damage of sensors, and unknown obstacles in the sensing area may cause the dynamic change of the network topology. However, few of the existing routing algorithms and media access control (MAC) protocols can perform both reliably and with energy efficiency in such highly dynamic networks. Since a large amount of traffic needs to be transmitted to the central controller, the nodes near the controller will die much sooner than those nodes in the marginal area, which will lead to an unbalanced lifetime of the entire network.

Second, in an unknown environment, sensors can hardly be placed precisely at the intended locations with little information about the environment available. In some areas, too many nodes are placed with very high density, whereas, in other areas, some coverage gaps exist due to the sparse density. In this paper, we assume that the sensing module of a sensor is omnidirectional and the coverage area of a sensor is defined as a disk-shaped area around the sensor, within which the sensor can detect the occurring of interested events with a certain probability. In order to utilize the limited number of sensor nodes, the total coverage area should be maximized. In addition, since many tasks require the coverage to be continuous, the coverage gap should be minimized.

Third, sensors should be able to adjust the density adaptively based on the requirements of different tasks or regions. An example of such nonuniform deployment is the temperature monitoring system for the volcano. In the region near the crater, the temperature changes so rapidly that more nodes are needed to measure the temperature precisely. On the other hand, at the foot of the volcano, it is unnecessary to deploy sensor nodes very densely to monitor the smoothly changing temperature. We will focus on how we can make sensor nodes "smarter" so that they can deploy themselves based on different requirements of tasks.

In summary, we expect that our algorithm can deploy sensors to cover the maximum working area with minimum coverage gap and overlap. The algorithm should not only maximize the coverage but also minimize the total moving distance in the network to reduce energy consumption. In addition, the algorithm should be able to adjust the node density based on different task requirements for different regions. Furthermore, no global information should be required in the algorithm to avoid the huge overhead for exchanging global location information.

Next, we give some assumptions used in this paper. In order to enable the proposed algorithm to work in an environment where the GPS is unavailable, such as the outer space, the seabed, and indoors, global location information should not be required. In the area without the GPS, we assume that each sensor can estimate the relative location information to nearby nodes by detecting the relative distances and angles [18], [19], [21], [27]. In addition, we assume that each node has a unique reference direction which can be easily obtained from the electronic compass. Each node is able to monitor a unit-disk-shaped area centered at itself, though our algorithm can work for other regular or irregular shapes of sensing areas with only a minor modification.

3 Related Work

There has been some previous work on the maximum coverage problem for sensor networks in the literature. The *potential-field-based deployment algorithm* [23] assumes that the movement of each node can be affected by virtual force from other nodes and obstacles. In the algorithm, all nodes explore from a compact region and fill the maximum working area in a way similar to the particles in the microworld [20]. Although this approach can maximize the coverage area, since the main idea of this algorithm is obtained from the microworld, the nodes in the network may oscillate for a long time before they reach the static equilibrium state like the particles in the microworld. The oscillation of nodes consumes much more energy than moving to the desired location directly. Moreover, this algorithm can only be used in a bounded area since the nodes must be restricted within the boundary by the virtual force from boundary. Without the boundary, each node will not stop expelling others until there are no other nodes within its transmission range. The *Virtual Force Algorithm (VFA)* [26] divides a sensor network into clusters. Each clusterhead is responsible for collecting the location information of the nodes and determining their targets. The cluster architecture may lead to an unbalanced lifetime

of the nodes and is not suitable for networks that do not have powerful central nodes. The *Constrained Coverage algorithm* [24] can guarantee that each node has at least K neighbors by introducing two virtual forces. However, it still does not have any mechanism to limit the oscillation of nodes. *Movement-Assisted Sensor Deployment algorithms* [25], which consist of three independent algorithms (the *Vector-based algorithm (VEC)*, the *Voronoi-based algorithm (VOR)*, and *MiniMax*), use Voronoi diagrams to discover the coverage holes and maximize the coverage area by pushing or pulling nodes to cover the coverage gaps based on virtual forces. In the VEC algorithm, the nodes that have covered their corresponding Voronoi cells do not need to move, whereas other nodes are pushed to fill coverage gaps. In VOR, nodes will move toward the farthest Voronoi vertices. The MiniMax algorithm moves nodes more gently than VOR, thereby avoiding the generation of new holes in some cases. Compared to the potential-field-based deployment algorithm, movement-assisted sensor deployment algorithms reduce the oscillation and save the energy consumed by node movement. All three algorithms assume that each node knows its Voronoi neighbors and vertices. However, the Voronoi diagram is a global structure and all Voronoi vertices and cells can only be obtained when the global location information of the nodes in the network is known [22], which means that each node must exchange its current location information with all other nodes in the network to acquire its corresponding Voronoi vertices and cell. For each location update message, it may take $O(n)$ one-hop transmissions to reach all other nodes, where n is the total number of nodes in the network. In the case when the GPS is unavailable, the error in one-hop relative locations of the nodes may be accumulated. Thus, the error for two far away nodes may be significant. In addition, so far, most of the existing algorithms are only concerned with deploying nodes within a bounded area. The VOR and Minimax algorithms are based on Voronoi diagrams and require every Voronoi cell to be bounded. However, by the definition of Voronoi graph [22], each periphery Voronoi cell is unbounded since it contains a Voronoi vertex at infinity. Thus, the VOR and Minimax algorithms cannot be used in this situation.

Finally, though all the algorithms discussed above intended to maximize node coverage, minimize coverage overlap and gap, and deploy nodes uniformly, they did not answer a fundamental question in the deployment: What type of node layout can provide the maximum coverage with the smallest overlap and gap? We will address this issue in the next section.

4 Ideal Node Layout for Maximum Coverage

Similarly to the deployment algorithms discussed in Section 3, one of the important goals of our algorithm is to maximize the coverage area. However, before we design a maximum coverage algorithm, we need to know what type of node layout can provide the maximum coverage for a given number of nodes. In order to find the ideal node layout for the maximum coverage, we introduce the Delaunay triangulation [22] to describe the layout of the network. Let N be a set of n nodes, which are randomly thrown into the plane, and T be a Delaunay triangulation of N such that no other nodes in N are inside the circumcircle of any triangle in T. Suppose that a large number of sensor nodes are randomly thrown into a 2D field. The entire sensing area can be partitioned into some Delaunay triangles whose vertices represent sensor nodes. We assume that the number of nodes is so large that the entire working area consists of a large number of Delaunay triangles. We have the following theorem regarding the optimum node layout:

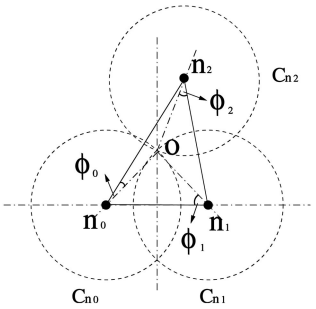


Fig. 1. The maximum no-gap coverage area in a triangle can be obtained if and only if the lengths of all three edges of the Delaunay triangle equal $\sqrt{3}r$.

Theorem 1. *If all Delaunay triangles are equilateral triangles with edge length $\sqrt{3}r$, then the coverage area of n nodes is maximum without coverage gap.*

Proof. Since the entire working area can be decomposed into a large number of Delaunay triangles, if we can prove that the no-gap coverage area in any Delaunay triangle is maximized when the lengths of all of its edges equal $\sqrt{3}r$, then the maximum coverage area of n nodes can be obtained. Let C_{n_0}, C_{n_1}, and C_{n_2} be the circles centered at points n_0, n_1, and n_2, respectively, which denote the coverage area of the corresponding nodes. Without loss of generality, we assume that circle C_{n_0} and circle C_{n_1} cross at point O, where O and n_2 locate on the same side of edge (n_0, n_1), as shown in Fig. 1. Let $\phi_0 = \angle n_2 n_0 O$, $\phi_1 = \angle n_0 n_1 O$, and $\phi_2 = \angle n_1 n_2 O$, where $0 < \phi_0$, ϕ_1, and $\phi_2 < \frac{\pi}{2}$. Let $|\overline{n_i O}|$ denote the distance between node n_i and O, where $n_i \in \{n_0, n_1, n_2\}$. In Fig. 1, since C_{n_0} and C_{n_1} cross at O, we can obtain $|\overline{n_1 O}| = |\overline{n_0 O}| = r$. In order to maximize the area of the triangle without coverage gap, $|\overline{n_2 O}|$ should equal r. The area of triangle $\Delta n_0 n_1 n_2$, denoted as $A(\Delta n_0 n_1 n_2)$, can be calculated as the summation of the area of three triangles $\Delta n_0 n_1 O$, $\Delta n_0 n_2 O$, and $\Delta n_2 n_1 O$:

$$\begin{aligned} A(\Delta n_0 n_1 n_2) &= A(\Delta n_0 n_1 O) + A(\Delta n_0 n_2 O) + A(\Delta n_2 n_1 O) \\ &= r^2(\sin\phi_0 \cos\phi_0 + \sin\phi_1 \cos\phi_1 + \sin\phi_2 \cos\phi_2) \\ &= \frac{r^2}{2} \times (\sin(2\phi_0) + \sin(2\phi_1) + \sin(2\phi_2)). \end{aligned} \tag{1}$$

Since

$$|\overline{n_1 O}| = |\overline{n_0 O}| = |\overline{n_2 O}| = r,$$

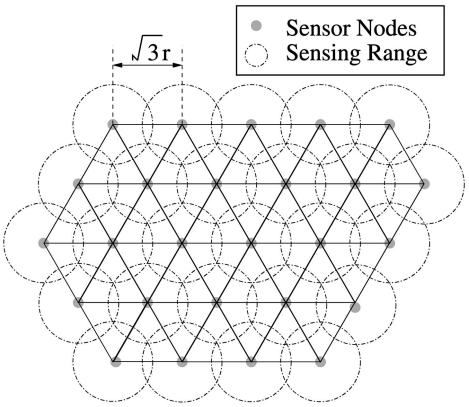


Fig. 2. The perfect node layout for the maximum no-gap coverage.

we have

$$\phi_0 + \phi_1 + \phi_2 = \frac{\pi}{2}. \tag{2}$$

By replacing ϕ_2 with $(\frac{\pi}{2} - \phi_0 - \phi_1)$ in (1), we obtain

$$A(\Delta n_0 n_1 n_2) = \frac{r^2}{2} \times (\sin(2\phi_0) + \sin(2\phi_1) + \sin(2\phi_0 + 2\phi_1)).$$

Let

$$f(\phi_0, \phi_1) = \sin(2\phi_0) + \sin(2\phi_1) + \sin(2\phi_0 + 2\phi_1).$$

When

$$\frac{\partial f(\phi_0, \phi_1)}{\partial \phi_0} = 0 \quad \text{and} \quad \frac{\partial f(\phi_0, \phi_1)}{\partial \phi_1} = 0,$$

the maximal value of $A(\Delta n_0 n_1 n_2)$ can be obtained. Thus,

$$\begin{aligned} \frac{\partial f(\phi_0, \phi_1)}{\partial \phi_0} &= 2\cos(2\phi_0) + 2\cos(2\phi_0 + 2\phi_1) = 0, \\ \frac{\partial f(\phi_0, \phi_1)}{\partial \phi_1} &= 2\cos(2\phi_1) + 2\cos(2\phi_0 + 2\phi_1) = 0. \end{aligned} \tag{3}$$

By solving (3), we obtain

$$\begin{aligned} 2\phi_0 + \phi_1 &= \frac{(2k+1)\pi}{2}, \quad k = 0, \pm 1, \pm 2, \ldots, \\ \phi_0 + 2\phi_1 &= \frac{(2m+1)\pi}{2}, \quad m = 0, \pm 1, \pm 2, \ldots. \end{aligned} \tag{4}$$

Since $0 < \phi_0, \phi_1, \phi_2 < \frac{\pi}{2}$, we have

$$2\phi_0 + \phi_1 = \phi_0 + 2\phi_1 = \frac{\pi}{2}. \tag{5}$$

From (2) and (5), we can obtain that the maximum value of $A(\Delta n_0 n_1 n_2)$ can only be achieved when $\phi_0 = \phi_1 = \phi_2 = \frac{\pi}{6}$. We have shown that, when $\phi_0 = \phi_1 = \phi_2 = \frac{\pi}{6}$ and the lengths of all three edges $\overline{n_0 n_1}$, $\overline{n_1 n_2}$, and $\overline{n_2 n_0}$ equal $\sqrt{3}r$, the area of triangle $\Delta n_0 n_1 n_2$ is maximized. Therefore, as depicted in Fig. 2, if all Delaunay triangles are equilateral triangles with edge length $\sqrt{3}r$, then the no-gap coverage area in a plane is maximized. □

Having considered the maximum coverage problem, we now derive the minimum average moving distance of the nodes under a uniform deployment density. We assume that, initially, all sensor nodes are in a compact area near the origin of the polar coordinate system and eventually will be deployed to a disk-shaped area S with radius D, that is, the sensors are uniformly distributed over area $S = \pi D^2$. We have the following theorem regarding the minimum average moving distance:

Theorem 2. *When sensor nodes are deployed from a compact area to a disk-shaped area S with radius D, the minimum average moving distance of the nodes is $\frac{2D}{3}$.*

Proof. When nodes are uniformly distributed over area S, the minimum average moving distance D_{avg} can be computed as the average distance from the origin of the polar coordinate system. Let (ρ, θ) denote the polar coordinate of the node. Thus,

$$D_{avg} = E(\rho) = \int_0^{2\pi} \int_0^D \frac{\rho}{\pi D^2} \rho d\rho d\theta = \frac{2D}{3}.$$

□

By plugging $S = \pi D^2$ into (6), we have

$$D_{avg} = \frac{2}{3}\sqrt{\frac{S}{\pi}}. \tag{7}$$

It should be pointed out that D_{avg} can be achieved only when every node moves directly to its final position during the deployment. Thus, it represents the minimum average moving distance in the ideal situation. However, this optimum value is difficult to achieve when the final position of each node is unknown before the deployment and there are obstacles that may block the movement of the nodes. Nevertheless, D_{avg} can serve as a lower bound on the average moving distance of the nodes for any deployment algorithm. We will compare the average moving distance of our algorithm with D_{avg} in the simulation section.

5 Adaptive Triangular Deployment (ATRI) Algorithm

In this section, we present a new adaptive deployment algorithm based on the optimum node layout that we obtained in Section 4. For convenience of presentation, we start with a simpler version of the algorithm, called the *triangular deployment (TRI) algorithm*, then discuss some strategies to improve the basic algorithm, and, finally, present the complete algorithm.

5.1 The Basic Triangular Deployment Algorithm

We have known what type of node layout can maximize the coverage in a plane. Now, the next issue that needs to be addressed is how we can deploy nodes from a compact area or an irregular layout to a perfect layout. A large number of sensor nodes are randomly thrown into a working area or placed in a bunch. As discussed earlier, exchanging global location/topology information during such a dynamic deployment period would put a heavy traffic burden on the network. Furthermore, when a bunch of nodes are located within a compact area and most of them need to communicate with others at the same time, the communication will be very inefficient due to the collision at the MAC layer [16]. Thus, the node movement decision should

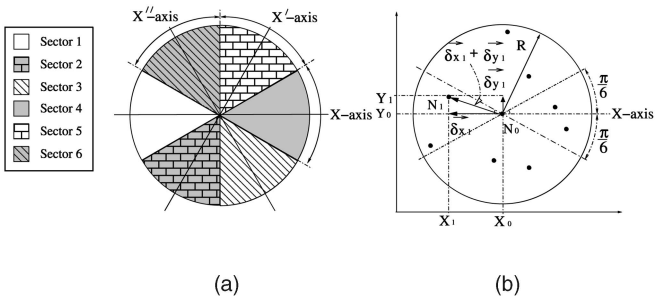


Fig. 3. Local movement strategies based on the location of one-hop neighbors.

be based on the local information in the deployment process. Since the location information is updated periodically, as a result, each node can only decide its movement periodically.

We now present an algorithm to deploy the sensor nodes close to a perfect equilateral triangular layout with the maximum coverage. The basic idea of the algorithm is to adjust the distance between two Delaunay neighbors to $\sqrt{3}r$ in three different coordinate systems, namely, XY, $X'Y'$, and $X''Y''$, where the angles between the X-axis and the X'-axis and between the X-axis and the X''-axis are $\frac{\pi}{3}$ and $\frac{2\pi}{3}$, respectively. As shown in Fig. 3a, the coverage area of a sensor node is divided into six sector areas, called sectors 1 to 6 counterclockwise, where the X-axis, X'-axis, and X''-axis symmetrically partition sectors 1 and 4, sectors 2 and 5, and sectors 3 and 6, respectively. The radius of each sector equals the transmission range of the node. The locations of the nodes in sectors 1 and 4, sectors 2 and 5, and sectors 3 and 6 are expressed by the XY, $X'Y'$, and $X''Y''$ coordinates, respectively.

In each sector, the node adjusts its location along the corresponding axis based on the location of its Delaunay neighbors. However, the adjustment algorithm based on the Delaunay diagram suffers from the similar limitation as the solutions based on the Voronoi graph since the global location of all sensor nodes is needed to determine the Delaunay triangulations and Delaunay neighbors. In practice, we use the nearest neighbors to the node in each sector instead of the Delaunay neighbors. For example, as shown in Fig. 3b, node N_1 is the nearest neighbor of node N_0 in sector 1, where their coordinates in the XY-coordinate system are (X_1, Y_1) and (X_0, Y_0). Let location vectors $\overrightarrow{\delta x_1}$ and $\overrightarrow{\delta y_1}$ denote vectors $[(X_1 - X_0), 0]$ and $[0, (Y_1 - Y_0)]$, respectively. If $|\overrightarrow{\delta x_1} + \overrightarrow{\delta y_1}| < \sqrt{3}r$, then it means that there is too much coverage overlap between node N_0 and node N_1. Thus, the movement of N_0 should be opposite to N_1 for

$$\frac{|\overrightarrow{\delta x_1} - \sqrt{3}r\frac{\overrightarrow{\delta x_1}}{|\overrightarrow{\delta x_1}|}|}{2}$$

to reduce the coverage overlap. On the contrary, if $|\overrightarrow{\delta x_1} + \overrightarrow{\delta y_1}| > \sqrt{3}r$, then a coverage gap may exist between node N_0 and node N_1. We will let N_0 move toward N_1 along the X-axis for

$$\frac{|\overrightarrow{\delta x_2} - \sqrt{3}r\frac{\overrightarrow{\delta x_2}}{|\overrightarrow{\delta x_2}|}|}{2}$$

to fill the coverage gap. Besides the movement on the X-coordinate, the movement vector of N_0 projected on the Y-coordinate equals $\overrightarrow{\delta y_1}/2$. Thus, the movement vector of N_0,

$$\overrightarrow{\delta v_1} = \frac{\overrightarrow{\delta x_1} - \sqrt{3}r\frac{\overrightarrow{\delta x_1}}{|\overrightarrow{\delta x_1}|} + \overrightarrow{\delta y_1}}{2}.$$

In general, for sector s, each node searches the nearest neighbor within the sector and calculates the relative horizontal and vertical location vectors $\overrightarrow{\delta x_s}$ and $\overrightarrow{\delta y_s}$ along its corresponding axis. Here, $\overrightarrow{\delta x_s}$ and $\overrightarrow{\delta y_s}$ are expressed by relative coordinates corresponding to sector s. The movement vector of a node in sector s, $\overrightarrow{\delta v_s}$, can be expressed as

$$\overrightarrow{\delta v_s} = \frac{\overrightarrow{\delta x_s} - \sqrt{3}r\frac{\overrightarrow{\delta x_s}}{|\overrightarrow{\delta x_s}|} + \overrightarrow{\delta y_s}}{2}. \quad (8)$$

Note that each movement vector is obtained in one of the three different coordinate systems: XY, $X'Y'$, and $X''Y''$. After the movement vectors in all six sectors are obtained, they need to be transferred into uniform coordinates and added in order to obtain the total movement vector for the current round. Table 1 gives this TRI algorithm. As will be

TABLE 1
Triangular Deployment Algorithm

Triangular Deployment Algorithm
for each round
  for each node $i = 1$ to n
    Broadcast "Hello" message containing its location information to its one-hop neighbors;
    Receive "Hello" message from nearby nodes and obtain their location information;
    Divide its coverage area into 6 sectors;
    for each sector $s = 1$ to 6
     The node calculates location vector $\overrightarrow{\delta x}$ and $\overrightarrow{\delta y}$ to its nearest neighbor in the coordinate system;
     Calculate and store the movement vector for sector s, $\overrightarrow{\delta v_s} = \frac{\overrightarrow{\delta x} - \sqrt{3}r\frac{\overrightarrow{\delta x}}{|\overrightarrow{\delta x}|} + \overrightarrow{\delta y}}{2}$
    Transfer movement vectors of all sectors into uniform coordinates;
    Add them up to obtain the total movement vector for node i;
Move;

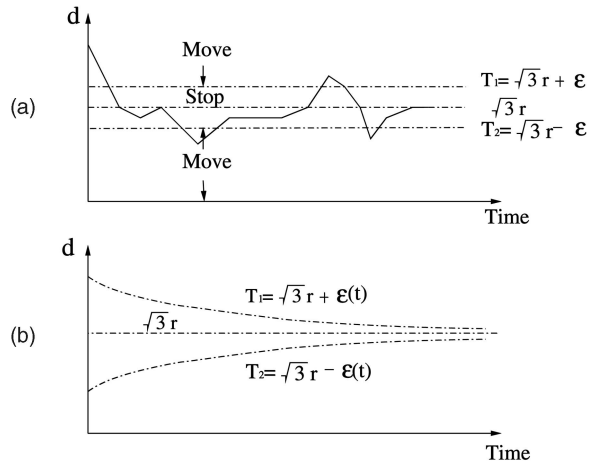


Fig. 4. Threshold strategy for reducing node oscillation. (a) Constant threshold. (b) Variable threshold.

seen in the simulation results, after several rounds of such adjustments, the layout of the network will be close to the ideal equilateral triangle layout. As a result, the coverage area of the network will be maximized.

5.2 Minimizing Oscillation

We have proven that the equilateral triangular layout can maximize the coverage and we also proposed a simple algorithm to adjust the network from an irregular layout to the ideal equilateral triangle layout. However, since the global location information of the network is difficult to obtain in the deployment process, it is impossible for each node to move to its desired target directly. Thus, the sensor nodes may move back and forth frequently before they reach their desired target. To make the algorithm suitable to real-world applications, another important issue is to reduce the total moving distance of the nodes in the deployment.

Recall that the moving strategy in our TRI algorithm is that if the horizontal distance between two neighbors is longer than $\sqrt{3}r$, then the sensors will move toward each other to shorten the gap between them. On the contrary, they will move away from each other to reduce the coverage overlap. According to this strategy, the nodes will move all the time unless the network reaches the perfect layout or its maximum rounds. In order to reduce the oscillation, we adopt a threshold strategy by using two distance thresholds, T_1 and T_2, instead of $\sqrt{3}r$ for making moving decisions, where $T_1 = \sqrt{3}r + \varepsilon$, $T_2 = \sqrt{3}r - \varepsilon$, and ε is a small constant. As described in Fig. 4a, the Y-coordinate d denotes the distance between the node and its nearest neighbor. When two far away nodes move toward each other and the distance between them decreases to T_1, two nodes stop moving. On the other hand, when two close nodes move apart and the distance between them increases to T_2, they will stop and keep the current distance between them. This moving strategy guarantees that the node will not move if it is located between T_1 and T_2 away from its neighbors so that the node is affected less when its neighbors move slightly. Note that if the adjustment granularity is too small, which is given by $\Delta d = T_1 - T_2 = 2\epsilon$, then T_1 and T_2 are close to $\sqrt{3}r$ at the beginning of the deployment process and there will be no obvious difference between the algorithm in Table 1 and the algorithm with threshold strategy. However, if Δd is too large, then it is impossible to adjust the network to the perfect equilateral triangular layout. In order to solve this problem, we let the thresholds T_1 and T_2 be the function of time t. As shown in Fig. 4b, the adjustment granularity decreases as time t increases. That is, $T_1 = \sqrt{3}r + \epsilon(t)$ and $T_2 = \sqrt{3}r - \epsilon(t)$, respectively, where $\epsilon(t)$ is called the threshold function. In practice, since the algorithm is executed round by round, the threshold can be changed to a function of the number of rounds. In our simulations in Section 6, we use $\epsilon(Rd_{cur}) = \frac{\sqrt{3}}{4}r \times e^{-Rd_{cur}/Rd_{total}}$ as the threshold function, where Rd_{cur} and Rd_{total} are the numbers of the current rounds and the total rounds, respectively.

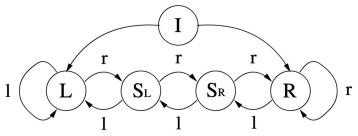


Fig. 5. Movement state diagram for reducing node oscillation. L: move left. R: move right. S_L and S_R: stay. I: initial state. l: expected moving direction is left. r: expected moving direction is right.

The second strategy, called the movement state diagram, is to use a state diagram to reduce the node oscillation. Each movement can be considered as a vector and be decomposed into the projection vectors on the X and Y-coordinates. For the X-coordinate, the nodes can only move left or right. Oscillation exists when a node moves toward the opposite direction of the previous movement. In order to avoid oscillation, the nodes are not allowed to move backward immediately. Two state diagrams are used in the movement vectors projected on the X and Y-coordinates separately. Fig. 5 shows an example of the movement state diagram for the X-coordinate which contains five states and is used in our simulation. The diagram has five states, L, R, S_L, S_R, and I, and two transitions, l and r. L and R denote the movement to the left and the right, respectively. If the state of a node is S_L or S_R, then it has to stay where it is till the next round. I is the initial state. l and r represent the moving decision to the left and the right made by the TRI algorithm. For example, a node plans to move left after running the triangular algorithm, which means that the current transition is l. Then, it needs to check its current state on its state diagram. If its current state is L, I, or S_L, then the next state will go to L after the transition l. A node can move left only when its next state is L. If the current state of the node is S_R (or R), then the next state will transit to S_L (or S_R) upon the transition l. Thus, the node cannot move until the next round. The movement control for the Y-coordinate follows a similar procedure. The simulation results in Section 6 show that the distance threshold strategy and movement state diagram strategy can reduce a significant number of movements during the deployment.

5.3 Adaptive Triangular Deployment Algorithm

We have discussed how we can deploy nodes with equal or almost equal density in an open area where the entire area needs to be sensed uniformly. However, in many real-world applications, the working area is partially or entirely bounded. Also, some irregularly shaped obstacles may be

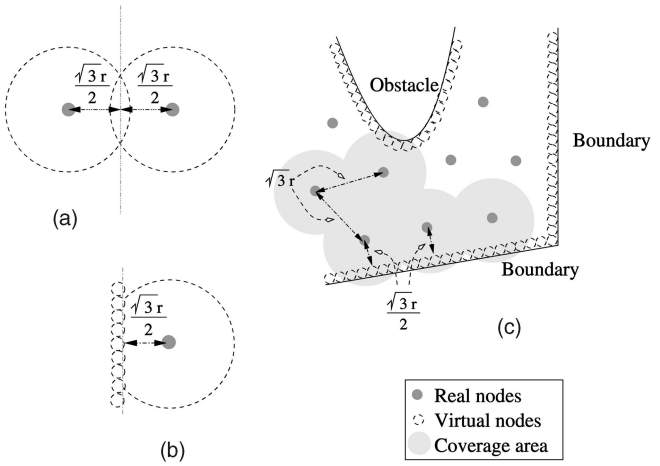


Fig. 6. Examples of the adaptive deployment in a bounded area with obstacles.

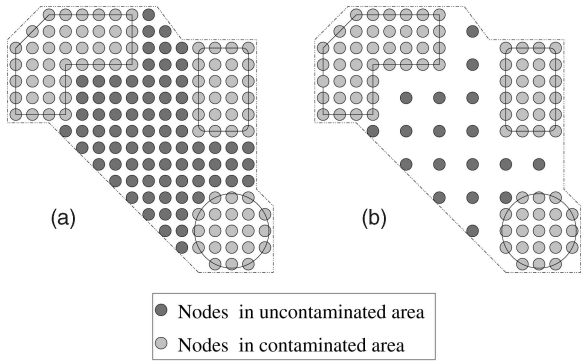


Fig. 7. Examples of the adaptive deployment based on different task requirements. (a) Uniform distribution. (b) Nonuniform distribution based on interested events.

in the working area. In other situations, the sensors may need to be deployed with different density based on the requirements of tasks. Without the control from the human being or central controller and without the map and global information, sensors have to be smart enough to make decisions themselves.

In order to make the deployment algorithm more practical, the sensors must be able to avoid obstacles and boundaries. Because an accurate map of the sensing area may not always be available before the deployment, we assume that each sensor is equipped with an ultrasonic obstacle-detecting module [28], which makes it possible to detect obstacles when it moves close enough to the obstacles. As discussed above, the TRI algorithm can only adjust the relative positions of two sensor nodes. However, unlike sensor nodes, obstacles and boundaries usually have irregular shapes and continuous outlines. In order to enable the TRI algorithm to adjust the relative positions between sensor nodes and obstacles and boundaries with only a minor modification, the outlines of obstacles and boundaries are abstracted as many virtual nodes which surround obstacles and boundaries closely. As shown in Fig. 6, each small dotted circle around the obstacles and boundaries denotes a virtual node. In practice, after each sensor node detects the outlines of the obstacles or boundaries within its coverage area, from a sensor node's point of view, the outlines of the obstacles or boundaries can be considered as many virtual nodes. Like real sensor nodes, these virtual nodes also "push" real nodes away when real nodes are located too close to them or "pull" real nodes close to them when real nodes are located too far away from them. Similarly to the basic TRI algorithm, after a real sensor node divides its coverage area into six sectors, it takes account of both real nodes and virtual nodes in each sector of its coverage area. When the real node finds that other real or virtual nodes are located too close to itself in each sector, it moves away from them. On the contrary, when it is located too far away from other real or virtual nodes, it moves toward them to fill the coverage gap. However, unlike real nodes, virtual nodes can neither move nor cover the sensing area. Since virtual nodes cannot actually cover any area, in order to avoid the coverage gap or overlap between real nodes and virtual nodes, intuitively, the optimum distance between adjacent real nodes and virtual nodes should be shorter than $\sqrt{3}r$. In Fig. 6a, the coverage areas of two real nodes are equally divided by a straight line. As discussed earlier, both real nodes need to be at least $\sqrt{3}r/2$ away from the line bisector in the optimum layout. The straight line can be considered as the virtual "boundary" of the coverage areas of two nodes. In Fig. 6b, when the line becomes a real boundary of an obstacle or a wall, the node on the right still needs to keep the distance to the line as $\sqrt{3}r/2$ to avoid the coverage gap. Thus, the distance between a real node and a virtual node needs to be adjusted to $\sqrt{3}r/2$ instead of $\sqrt{3}r$. We revise our algorithm as follows for deploying nodes in an area with obstacles:. Let r_d denote the deployment radius. In each sector, if the nearest neighbor of the node is a virtual node, then the node sets its deployment radius r_d to $\frac{r}{2}$. If its nearest neighbor is a real node, then its r_d still equals its sensing radius r. Then, the node runs the TRI algorithm by replacing r with r_d. Fig. 6 shows an example of the layout after the nodes are deployed in a partially bounded area with irregularly shaped obstacles. We can see that the distance between the real nodes is still $\sqrt{3}r$, whereas the distance between the real nodes and the virtual nodes (obstacles or boundaries) is $\sqrt{3}r/2$.

Besides avoiding the obstacles in the working area, the location of nodes should also follow the occurring distribution of the interested events. The higher the frequency or density of the event occurring within a given area, the more sensor nodes are needed to monitor the status change in that area. On the contrary, it is unnecessary to monitor the event-blank area where interested events will never exist. In other words, the nodes should be deployed with different densities based on the requirements of tasks. For example, suppose the sensor nodes have two different types of tasks: sensing and communication. For different tasks or regions, the coverage range of a node may be different. As shown in Fig. 7, a bunch of sensors are thrown into a contaminated chemistry factory to monitor the density of leaked chemicals in which contaminated sources are located in separated buildings which may not be uniformly distributed. The task of some nodes with high density is to cover and monitor the contaminated area and the task of the other nodes located in the event-blank area with low density is to provide

TABLE 2
Adaptive Triangular Deployment Algorithm

Adaptive Triangular Deployment Algorithm
for each round
  for each node $i = 1$ to n
    Broadcast "Hello" message containing its location information to its one-hop neighbors;
    Receive "Hello" message from nearby nodes and obtain their location information;
    Detect obstacles or boundaries within its coverage area and obtain location of virtual nodes;
    Divide its coverage area into 6 sectors;
    for each sector $s = 1$ to 6
      Adjust its sensing radius r_d adaptively based on the requirement of tasks or location of virtual nodes;
      Calculate the threshold value THR;
      Calculate location vector $\overrightarrow{\delta x}$ and $\overrightarrow{\delta y}$ to its nearest neighbor/virtual node in the coordinate system;
      if $(0 < |\delta x_s| \leq |\sqrt{3}r_d - THR|)$ **or** $(|\delta x_s| \geq |\sqrt{3}r_d + THR|)$
        Calculate and store the movement vector for sector s, $\overrightarrow{\delta v_s} = \frac{\overrightarrow{\delta x} - \sqrt{3}r_d \frac{\overrightarrow{\delta x}}{|\overrightarrow{\delta x}|} + \overrightarrow{\delta y}}{2}$;
      else
        $\overrightarrow{\delta v_s} = \overrightarrow{0}$;
      end if
    Transfer movement vectors of all sectors into uniform coordinates;
    Add them up to obtain the total movement vector for node i;
    Check the state diagram to decide if move or not and make transitions on the state diagram;
  Move;

communications between any two separated contaminated buildings. Thus, the algorithm should not only maximize the coverage area where the nodes need to be deployed uniformly, but also adjust the node density based on the different distribution of event occurring. It is easy to revise the TRI algorithm to make it suitable for adaptive deployment based on different requirements. In the TRI algorithm, in order to obtain the maximum no-gap coverage, each node tries to adjust the distance to its nearest neighbors to $\sqrt{3}r$. When the nodes move into a highly concerned region and find that they need to be deployed more densely, they set a new shorter deployment radius, say, r_d, instead of the sensing radius r, and then run the ATRI algorithm by replacing r with r_d. Note that the deployment radius r_d of each sensor can be decided by itself based on the measurement obtained from the environment. When a region does not need to be sensed with high density, the nodes extend to a sparser density by choosing a longer deployment radius.

By applying the distance threshold, movement state diagram, and adaptive adjustment strategies, we obtain the ATRI algorithm that is summarized in Table 2. As will be seen in our simulation results, by incorporating these strategies, our ATRI algorithm can drive the nodes to avoid obstacles in the area and deploy them with different densities based on the requirements of tasks.

5.4 Discussions on Some Practical Issues: Synchronizing Sensors and Reducing Packet Collision

So far, we have assumed that the sensors are well synchronized and the location information can be exchanged between the sensors without packet collision. However, in practice, the clocks of sensors can be imprecise for several reasons. First, the clocks may not be initially well synchronized. The sensors may be turned on at different times. The clock may also be affected by changes in the environment, such as temperature and pressure, or the battery voltage. Without global synchronization, some sensors with a "faster clock" may reach the maximum rounds and stop moving, whereas others still seek and try to move to better positions to improve the coverage. In order to make all of the sensors move at the same pace, the sensors need to be globally synchronized. Synchronization in sensor networks has been studied in the literature by many researchers. For instance, in [29], Li and Rus proposed a fully localized diffusion-based synchronization method which scales well in a large network. This algorithm can be adopted in our deployment algorithm for synchronization purposes.

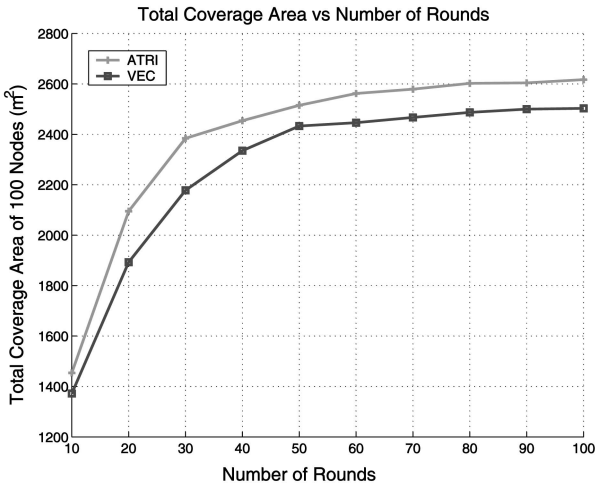
Moreover, in some applications, a large number of sensors are initially placed in a compact area. When every sensor wants to obtain the shared channel to broadcast its location information, the channel may become so busy that many packets collide. Though our main focus in this paper is on the movement planning of sensors, nevertheless, we next briefly discuss how sensors can reliably exchange location information in such a situation. A simple way to reduce collision is to deploy sensors as sparsely as possible if the environment and the application permit. As studied in some existing work [30], [31], another feasible solution is to adaptively adjust the contention window size based on how busy the channel is. However, both solutions cannot completely avoid collision. Unlike some existing work on MAC protocols, here, we are more concerned with the reliability than the throughput or the channel utilization during the deployment phase. In order to avoid collision, instead of using a flat topology, we can introduce a hierarchy into the network similar to [32]. During each round of the ATRI algorithm, we let some sensors act as clusterheads and poll other sensors to avoid packet collision. Since the network topology keeps changing before the maximum round of the ATRI algorithm is reached, the clusterheads are not fixed and should be selected at the beginning of each round. To become a clusterhead, the sensors first operate in a contention-based mode and

compete with each other to obtain the channel. Once a node obtains the channel by successfully sending out a broadcast message, it becomes a clusterhead. All sensors that receive the broadcast message stop trying to send out the message and become cluster members. After becoming a clusterhead, the sensor polls all sensors one by one. If the polled sensor is in the transmission range of the clusterhead, then the sensor sends its location information to the clusterhead. Otherwise, the channel stays idle for a short period and, then, the clusterhead will poll the next sensor. Though the procedure of electing the clusterhead is contention-based, after a sensor is selected as the clusterhead, the sensors in the cluster stop competing with each other and can upload their location information to the clusterhead in a contention-free manner. After acquiring the positions of all of the sensors in its transmission range, the clusterhead broadcasts all newly updated location information to all of the nodes in its transmission range. Thus, by running the polling protocol, inner cluster collision can be avoided.

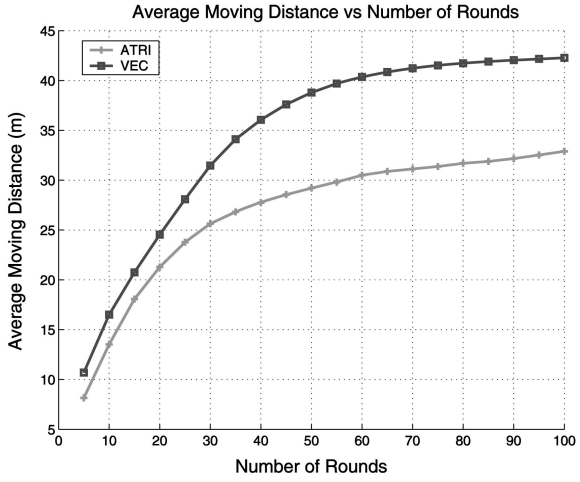
Unlike traditional one-hop wireless networks such as WLANs and Bluetooth networks, a sensor network may consist of multiple clusters. Each clusterhead may not know the activities of other clusterheads. Thus, if two nearby clusterheads broadcast packets at the same time, the packets may collide. The intercluster collision problem basically is a hidden terminal problem, which has been extensively investigated in the literature. Busy-tone-based approaches can solve the hidden terminal problem by using a busy tone to warn the nodes not to send packets. A busy tone can be a simple unmodulated SINE wave transmitted in a separate narrow-band channel. Tobagi and Kleinrock proposed a busy tone multiple access (BTMA) scheme [33] to solve the hidden terminal problem by requiring a receiver to power up a busy tone to warn hidden nodes. Haas and Deng presented a dual BTMA (DBTMA) scheme [34], which uses two physically separate tones: one indicating transmitting busy and another indicating receiving busy. Other nodes that hear the busy tone will postpone their transmission to avoid collision. The idea of busy tones can be used to solve the intercluster collision problem. Once a clusterhead obtains the channel, it broadcasts a busy tone in a separate channel with a much longer transmission range than that for transmitting regular data to disable the transmission of all nearby clusterheads and sensors. Since the busy tone is transmitted in a much simpler waveform and at a lower frequency band than regular data, the busy tone can be transmitted with more energy efficiency, even if it needs to be sent further than regular data. Thus, by using the combination of the polling protocol and the busy-tone scheme, the inner cluster and intercluster packet collision can be avoided and location information can be exchanged reliably.

6 Performance Evaluations

This section presents a set of experiments designed to evaluate the performance and cost of the proposed algorithm. Besides the ideal flat open area, the simulation is also run in more practical environments, where irregularly shaped obstacles may block the movement of the nodes. In addition, we implement the adaptive deployment based on different deployment requirements of the regions.



(a)



(b)

Fig. 8. The total coverage area and average moving distance of the ATRI and VEC algorithms for 100 runs of simulations. (a) Total coverage area versus number of rounds. (b) Average moving distance versus number of rounds.

All of the sensor nodes are equipped with Chipcon CC2420 Zigbee transceivers [35], which can reach as far as 50 m away. Each sensor node can sense the occurrence of events within a radius of 3 m away from itself. At the beginning of the experiments, the sensors are randomly placed within a $1m \times 1m$ compact square, which is centered at point $(25m, 25m)$. Then, the nodes explode to a large evenly deployed layout. In order to limit the oscillation of the nodes, the same movement state diagram depicted in Fig. 5 is used in all scenarios. The distance threshold function $\epsilon(Rd_{cur}) = \frac{\sqrt{3}}{4} r_d \times e^{-Rd_{cur}/Rd_{total}}$, where Rd_{cur} and Rd_{total} are the numbers of current rounds and total rounds. We measure the total coverage area and the average moving distance per node and compare them with existing algorithms.

6.1 Performance and Cost Evaluation

In this section, we compare the performance and cost of the VEC and ATRI algorithms. The VEC algorithm has similar

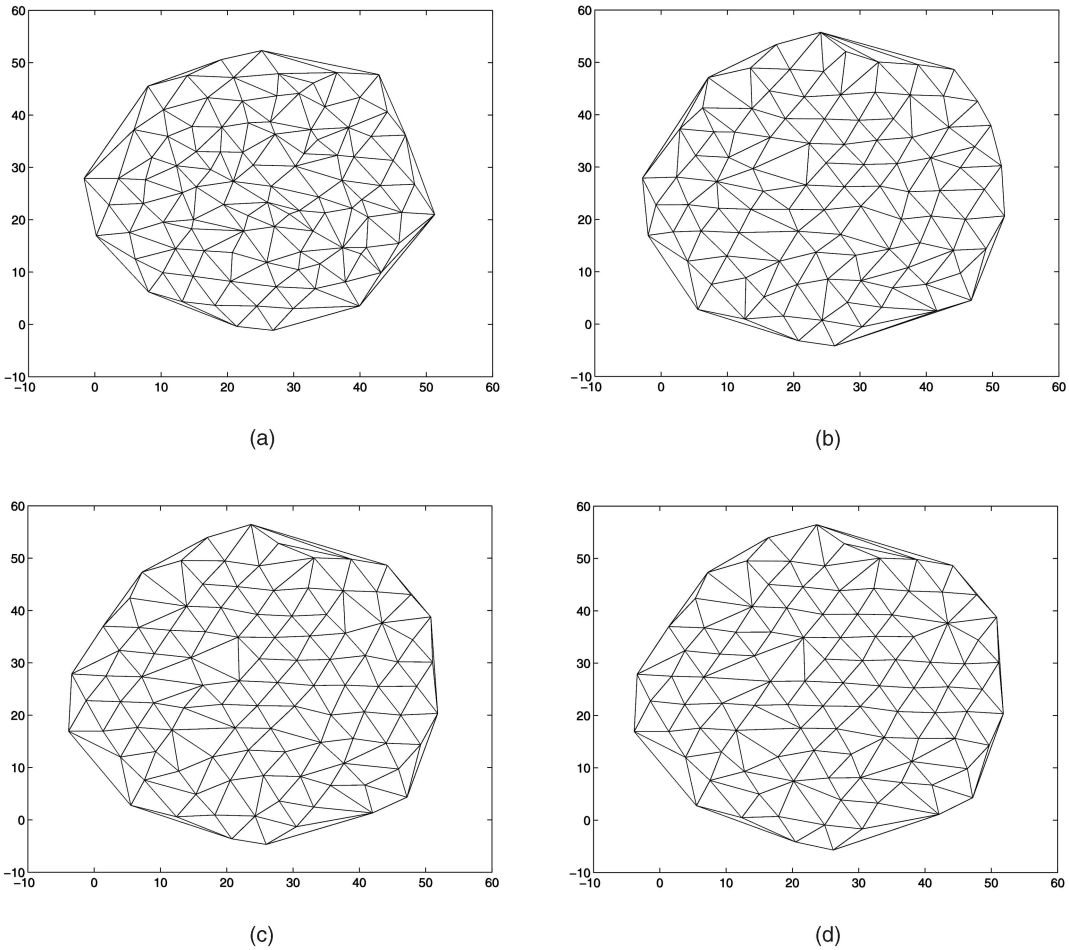


Fig. 9. Triangular layout of 100 nodes at rounds (a) 20, (b) 40, (c) 60, and (d) 80.

performance as the VOR and MiniMax algorithms in a bounded area. In addition, like the ATRI algorithm, the VEC algorithm can be used in both unbounded and bounded areas, whereas the VOR and MiniMax algorithms have to know the boundary information. For the sake of simplicity, we did not take into account the communication cost for exchanging location information and assume that the location information is error-free, though the VEC algorithm needs global location information and is more vulnerable to inaccurate location information than the ATRI algorithm. In order to see the effects of various node densities, 100 nodes are randomly placed into a $1m \times 1m$ square around point $(25m, 25m)$ at the beginning and, then, they explode from a compact area into a large area. Both algorithms run for 100 rounds. In order to evaluate the performance and cost of the two algorithms, two metrics are measured for each simulation round: total coverage area and average moving distance, which are defined as the coverage area of 100 nodes and the accumulated moving distance per node from the beginning of the simulation, respectively. Fig. 8a shows the total coverage area of both algorithms as the simulation rounds increase. We can see that the total coverage of both algorithms increases rapidly to as high as $2,200m^2$ before round 40 and then goes smoothly after round 40. At round 100, ATRI stops at about $2,600m^2$, whereas VEC is close to $2,500m^2$. From rounds 10 to 100, ATRI always leads VEC for about $50m^2$ to $100m^2$. Fig. 8b describes the average moving distance when the simulation rounds range from 10 to 100. The average moving distance has a similar increasing trend to the total coverage area as the simulation rounds increase. In both algorithms, after round 60, the nodes are deployed evenly and their average distance is close to $\sqrt{3}r$. Most nodes do not need to move, except for minor adjustments. As shown in Fig. 9, after round 60, most nodes form the equilateral triangle layout. There are no obvious changes between the layout of round 60 and that of round 80 because the layout of the nodes is already very close to the ideal equilateral triangular layout after round 60. In addition, from the total coverage area of both algorithms, we also calculate the optimum average moving distances and plot them in Fig. 8b. As discussed earlier, given a fixed total coverage area, the optimum average moving distance can be calculated by (7). Recall that the optimum average moving distances can only be obtained when the working environment is well known and each node knows its expected target before the deployment. Without the map of the environment, the nodes have to move in a zigzag manner, which makes the average moving distances of both algorithms longer than the optimum values. However, compared to the VEC algorithm, ATRI still saves up to 50 percent of the optimum average moving distance from rounds 10 to 100.

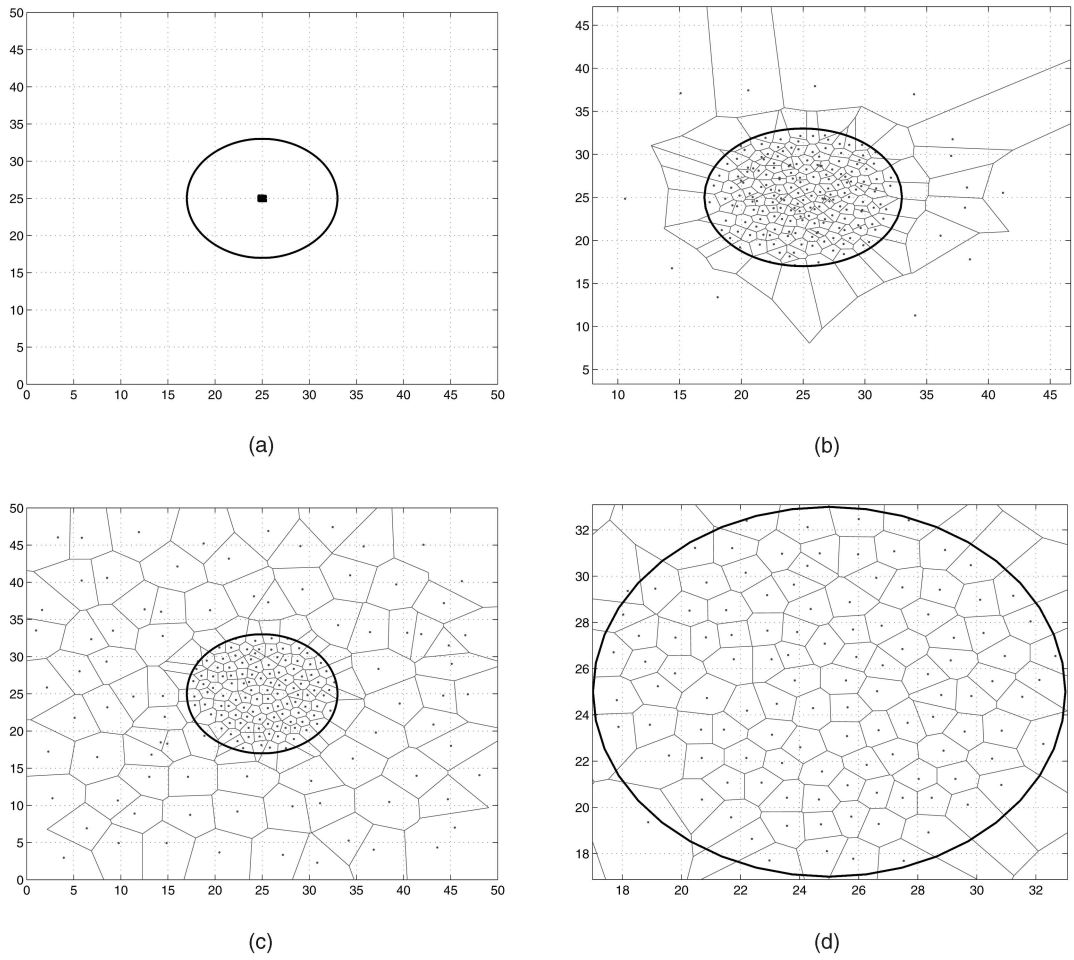


Fig. 10. Simulation results for nonuniform deployment. (a) Round 0. (b) Round 20. (c) Round 60. (d) Zoom-in snapshot of the contaminated area.

6.2 Nonuniform Deployment

In this section, we simulate the scenario that some dangerous chemical is leaking at point $(25m, 25m)$. Without loss of generality, we assume that the contaminated area is disk shaped, which is centered at point $(25m, 25m)$. The radius of the contaminated area is $10m$, which is unknown to the sensors before the deployment. In order to detect the region of the contaminated area, 200 sensor nodes are thrown within the compact square area close to the origin of the chemical. The circle around $(25m, 25m)$ in Fig. 10a represents the contaminated area. The sensor nodes within the contaminated area need to be more densely deployed than in the noncontaminated area to detect the small change in chemical density. Within the contaminated area, the sensing radius is $1m$, whereas the sensing radius within the noncontaminated area is $3m$. The whole sensing area is bounded within $50m \times 50m$, which is also centered at point $(25m, 25m)$. Figs. 10b and 10c plot the deployment layout at round 20 and round 60, respectively. Fig. 10d is the zoom-in snapshot of the contaminated area at round 60. We can see that, at round 60, the sensor nodes in both the contaminated area and the noncontaminated area are deployed evenly with the corresponding deployment radius. Our adaptive algorithm can also be used in various irregularly shaped contaminated areas or highly concerned areas. In addition, in a more complicated case where the contaminated area enlarges or shrinks from time to time, the sensor nodes can also change their deployment density dynamically to satisfy the requirement.

6.3 Exploring the Area with Obstacles

In order to show that the ATRI algorithm works well in the sensing environment with obstacles, a circular and a triangular obstacle are placed in the sensing area. As shown in Fig. 11, the circular obstacle is centered at point $(35m, 35m)$ with a radius of $8m$. The vertices of the triangular obstacle are $(10m, 20m)$, $(20m, 10m)$, and $(20m, 20m)$, respectively. Figs. 11a, 11b, 11c, and 11d illustrate the deployment layouts at rounds 0, 20, 40, and 60, respectively, where each "$+$" symbol denotes the position of its corresponding node. In the beginning, similarly to the previous scenario, 200 sensor nodes are randomly thrown into a $1m \times 1m$ square around point $(25m, 25m)$. As the simulation runs, we can see in Figs. 11b, 11c, and 11d that the total coverage enlarges round by round. At round 60, the nodes are evenly deployed and no nodes enter the triangular or the circular region during the deployment. We can see that, although obstacles block the movement of some nodes, the ATRI algorithm still performs very efficiently. During the simulation, we measure the total coverage area and the average moving distance at rounds 20, 40, and 60. In addition, we also calculate the optimum average moving distance by plugging the total

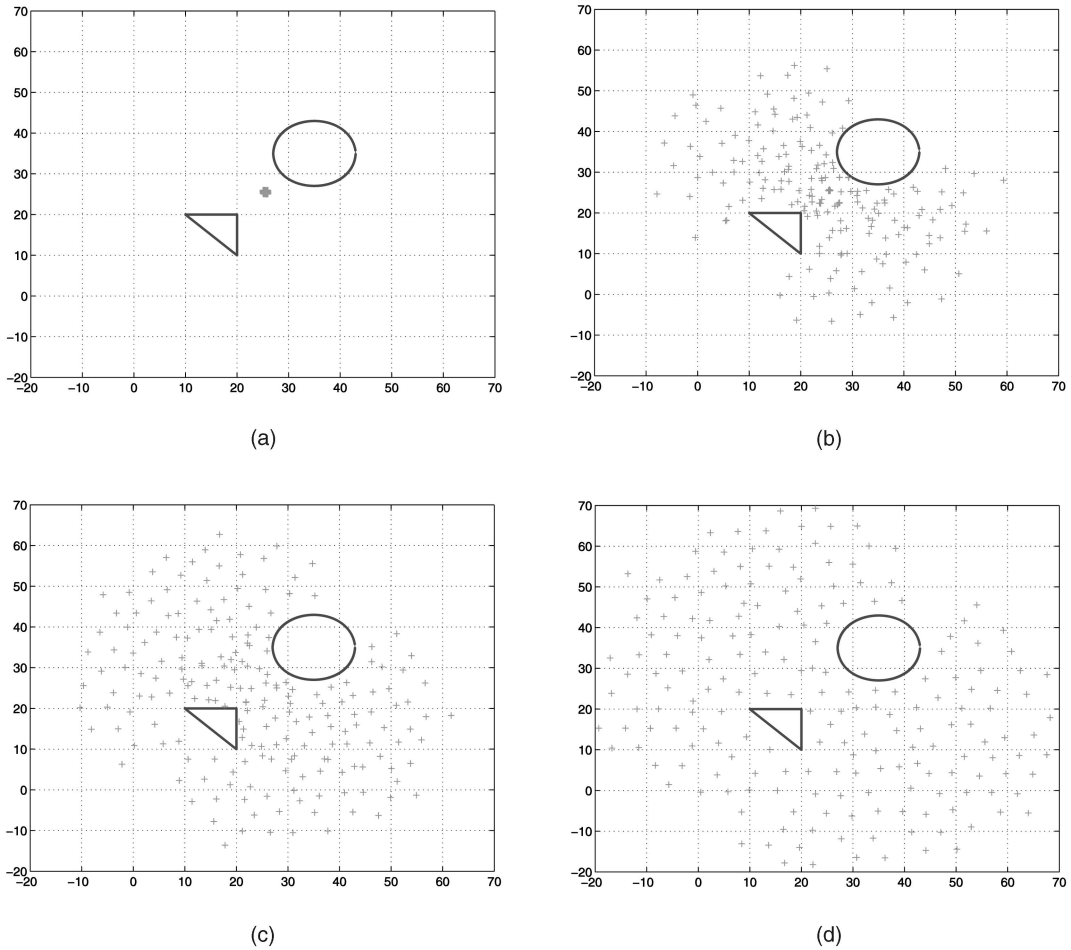


Fig. 11. Simulation results for the environment with obstacles. (a) Initial snapshot. (b) Round 20. (c) Round 40. (d) Round 60.

coverage area into (7). All three metrics are shown in Table 3.

In practice, the optimum average moving distance is difficult to reach unless the desired optimum position of each node is known before the deployment and no obstacles block the movement of the nodes. In the situation where the map of the environment is unavailable and the movement of some nodes is blocked by obstacles, we can see that the movement cost of the ATRI algorithm is very reasonable compared to the optimum value.

7 Conclusions

In this paper, we have proposed a new adaptive deployment algorithm for unattended mobile sensor networks, namely, the Adaptive Triangular Deployment (ATRI) algorithm. We have introduced an equilateral triangle deployment layout of sensor nodes and proved that it can produce the maximum no-gap coverage area in a plane. By using only the location information of one-hop neighbors for the adjustment of each node, the algorithm can make the overall deployment layout close to equilateral triangulations. In order to reduce the back-and-forth movement of nodes, the distance threshold strategy and movement state diagram strategy are adopted, which limit the oscillation and reduce the total movement distance of nodes. The ATRI algorithm can be used in both bounded and unbounded areas. It also supports adaptive deployment. Without the map and information of the environment, nodes can avoid obstacles and adjust the density dynamically based upon different requirements. Without control from the human being or central controller, each node can make decisions itself. In addition, the ATRI algorithm is run in a completely distributed fashion by each node and based only on the location information of nearby nodes.

TABLE 3
Exploring Sensing Area with Obstacles: Moving Distance versus Coverage Area

Number of rounds	20	40	60
Total coverage area (m^2)	2032	3504	4415
Average moving distance (m)	17.45	33.33	46.13
Optimum average moving distance (m)	16.95	22.26	25.00

Acknowledgments

This research work was supported in part by US National Science Foundation Grants CCR-0207999 and ECS-0427345 and US Army Research Office Grant W911NF-04-1-0439. The authors would like to thank the anonymous reviewers of this paper for very insightful suggestions and comments for improving the paper.

REFERENCES

[1] http://robotics.eecs.berkeley.edu/ pister/SmartDust/, 2004.
[2] http://www-mtl.mit.edu/research/icsystems/uamps/, 2004.
[3] http://www.isi.edu/scadds/pc104testbed/guideline.html, 2004.
[4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Comm. Magazine,* pp. 102-114, Aug. 2002.
[5] E. Biagioni and K. Bridges, "The Application of Remote Sensor Technology to Assist the Recovery of Rare and Endangered Species," *Int'l J. High-Performance Computing Applications,* special issue on distributed sensor networks, vol. 16, no. 3, Aug. 2002.
[6] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat Monitoring: Application Driver for Wireless Communications Technology," *Proc. 2001 ACM SIGCOMM Workshop Data Comm. in Latin America and the Caribbean,* Apr. 2001.
[7] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," *Proc. First ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA '02),* Sept. 2002.
[8] S. Chessa and P. Santi, "Crash Faults Identification in Wireless Sensor Networks," *Computer Comm.,* vol. 25, no. 14, pp. 1273-1282, Sept. 2002.
[9] L. Schwiebert, S.K.S. Gupta, and J. Weinmann, "Research Challenges in Wireless Networks of Biomedical Sensors," *Proc. MobiCom 2001,* pp. 151-165, 2001.
[10] G. Asada, T. Dong, F. Lin, G. Pottie, W. Kaiser, and H. Marcy, "Wireless Integrated Network Sensors: Low Power Systems on a Chip," *Proc. 24th European Solid State Circuits Conf. (ESSCIRC '98),* Oct. 1998.
[11] Wireless Integrated Networks Sensors (WINS) Project, http://www.janet.ucla.edu/WINS/, 2004.
[12] R. Min, M. Bhardwaj, N. Ickes, A. Wang, and A. Chandrakasan, "The Hardware and the Network: Total-System Strategies for Power-Aware Wireless Microsensors," *Proc. IEEE CAS Workshop Wireless Comm. and Networking,* Sept. 2002.
[13] The Ultra Low Power Wireless Sensor Project, http://www-mtl.mit.edu/ jimg/project_top.html, 2004.
[14] Mars Exploration Rover Project, http://mars.jpl.nasa.gov, 2004.
[15] A. Cerpa and D. Estrin, "ASCENT: Adaptive Self-Configuring Sensor Network Topologies," *Proc. IEEE INFOCOM '02,* June 2002.
[16] *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,* IEEE Standard 802.11, June 1999.
[17] *Understanding GPS—Principles and Applications,* E.D. Kaplan, ed. Artech House, 1996.
[18] D. Niculescu and B. Nath, "Ad Hoc Positioning System (APS)," *Proc. IEEE Global Comm. Conf. (GLOBECOM '01),* 2001.
[19] D. Niculescu and B. Nath, "Ad Hoc Positioning System (APS) Using AoA," *Proc. IEEE INFOCOM '03,* 2003.
[20] G.D. Coughlan, J.E. Dodd, and B.M. Gripaios, *The Ideas of Particle Physics: An Introduction for Scientists.* Cambridge Univ. Press, 1991.
[21] S. Capkun, M. Hamdi, and J.P. Hubaux, "GPS-Free Positioning in Mobile Ad Hoc Networks," *Proc. 34th Hawaii Int'l Conf. System Sciences (HICSS '01),* Jan. 2001.
[22] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry Algorithms and Applications.* Springer-Verlag, 1997.
[23] A. Howard, M.J. Mataric, and G.S. Sukhatme, "Mobile Sensor Network Deployment Using Potential Fields: A Distributed Scalable Solution to the Area Coverage Problem," *Proc. Sixth Int'l Conf. Distributed Autonomous Robotic Systems (DARS '02),* pp. 299-308, 2002.
[24] S. Poduri and G.S. Sukhatme, "Constrained Coverage for Mobile Sensor Networks," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA '04),* pp. 165-172, May 2004.
[25] G. Wang, G. Cao, and T. La Porta, "Movement-Assisted Sensor Deployment," *Proc. IEEE INFOCOM '04,* Mar. 2004.
[26] Y. Zou and K. Chakrabarty, "Sensor Deployment and Target Localizations Based on Virtual Forces," *Proc. IEEE INFOCOM '03,* 2003.
[27] N. Bulusu, J. Heidemann, and D. Estrin, "Adaptive Beacon Placement," *Proc. 21st Int'l Conf. Distributed Computing Systems (ICDCS '01),* Apr. 2001.
[28] J. Borenstein and Y. Koren, "Obstacle Avoidance with Ultrasonic Sensors," *IEEE J. Robotics and Automation,* vol. 4, no. 2, pp. 213-218, 1988.
[29] Q. Li and D. Rus, "Global Clock Synchronization in Sensor Networks," *Proc. IEEE INFOCOM '04,* Mar. 2004.
[30] F. Cali, M. Conti, and E. Gregori, "Dynamic Tuning of the IEEE 802.11 Protocol to Achieve a Theoretical Throughput Limit," *IEEE/ACM Trans. Networking,* vol. 8, no. 6, pp. 785-799, Dec. 2000.
[31] Y. Kwon, Y. Fang, and H. Latchman, "A Novel MAC Protocol with Fast Collision Resolution for Wireless LANs," *Proc. IEEE INFOCOM '03,* pp. 853-862, Apr. 2003.
[32] Z. Zhang, M. Ma, and Y. Yang, "Energy-Efficient Multi-Hop Polling in Clusters of Two-Layered Heterogeneous Sensor Networks," *Proc. 19th IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS '05),* Apr. 2005.
[33] F.A. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels (Part II: The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution)," *IEEE Trans. Comm.,* vol. 23, pp. 1417-1433, 1975.
[34] Z.J. Haas and J. Deng, "Dual Busy Tone Multiple Access (DBTMA)—A Multiple Access Control Scheme for Ad Hoc Networks," *IEEE Trans. Comm.,* vol. 50, no. 6, pp. 975-984, 2002.
[35] Chipcon CC2420 RF Transceiver, http://www.chipcon.com/, 2006.

Ming Ma received the BEng degree in electrical engineering from the University of Science and Technology of China, Hefei, China, in 2002. Since then, he has been working toward the PhD degree in the Department of Electrical and Computer Engineering at the State University of New York, Stony Brook. His research interests include wireless sensor networks, wireless mesh networks, and wireless local area networks. He is a student member of the IEEE and the IEEE Communication Society.

Yuanyuan Yang received the BEng and MS degrees in computer science and engineering from Tsinghua University, Beijing, and the MSE and PhD degrees in computer science from Johns Hopkins University, Baltimore. She is a professor of computer engineering and computer science at the State University of New York, Stony Brook. She is currently an editor of the *IEEE Transactions on Computers* and the *Journal of Parallel and Distributed Computing* and has served as an editor of the *IEEE Transactions on Parallel and Distributed Systems*. She has also served on the program/organizing committees of many international conferences in her areas of research, which include wireless networks, optical networks, high-speed networks, and parallel and distributed computing systems. Her research has been supported by the US National Science Foundation and US Army Research Office. She has published more than 170 papers in major journals and refereed conference proceedings and holds six US patents in these areas. She is a senior member of the IEEE and a member of the IEEE Computer Society.