# Distributed Algorithms for Energy-Efficient Even Self-Deployment in Mobile Sensor Networks

Yuan Song, Bing Wang, *Member, IEEE*, Zhijie Shi, *Member, IEEE*,
Krishna R. Pattipati, *Fellow, IEEE*, and Shalabh Gupta, *Member, IEEE*

**Abstract**—Even self-deployment is one of the best strategies to deploy mobile sensors when the region of interest is unknown and manual deployment is infeasible. A widely used distributed algorithm, Lloyd's method, can achieve even self-deployment. It however suffers from two critical issues when being used in mobile sensor networks. First, it does not consider limited sensor communication range. Second, it does not optimize sensor movement distances, and hence can lead to excessive energy consumption, a primary concern in sensor networks. This paper first formulates a locational optimization problem that achieves even deployment while it takes account of energy consumption due to sensor movement, and then proposes two iterative algorithms. The first algorithm, named Lloyd-$\alpha$, reduces the movement step sizes in Lloyd's method. It saves traveling distance while maintaining the convergence property. However, it leads to a larger number of deployment steps. The second algorithm, named Distributed Energy-Efficient self-Deployment (DEED), reduces sensor traveling distances and requires a comparable number of deployment steps as that in Lloyd's method. This paper further proposes an intuitive method to deal with limited sensor communication range that is applicable to all three methods. Extensive simulation using NS-2 demonstrates that DEED leads to up to 54 percent less traveling distance and 46 percent less energy consumption than Lloyd's method.

**Index Terms**—Mobile sensor networks, distributed algorithm, even self-deployment, Lloyd's method, centroidal voronoi tessellation

## 1 Introduction

Sensor nodes must be deployed appropriately to successfully accomplish their sensing tasks. When the region of interest is unknown or hostile (e.g., remote harsh fields, disaster areas or toxic urban regions), manual deployment is infeasible. In such cases, employing sensor mobility to achieve self-deployment is a suitable approach [1], [2], [3]. Even self-deployment, i.e., deploying the sensors evenly in the region, is one of the best known strategies in the absence of a prior knowledge of the region. For instance, it provides an optimal deployment for barrier coverage problem [4]. It also implies good coverage in "spot–sensing" applications [5], where each sensor node makes a measurement (e.g., temperature or humidity) at the precise location of the node.

According to Gersho's conjecture [6], for a given area and a set of sensors, the sensors are evenly distributed when they form a Centroidal Voronoi Tessellation (CVT) [7] (see more details on CVT in Section 2). Therefore, even self-deployment, which requires the sensors to form a CVT of the target area, differs from many existing studies where the goal is to maximize the coverage of the area [1], [2], [3]. For hostile or unknown fields, a centralized solution that pre-computes the final CVT and sensor final destinations is often infeasible due to the difficulty of gathering global knowledge and the lack of a centralized entity. Distributed algorithms that require no global information, but rather rely on sensors cooperation to form a CVT, are more desirable.

A widely used distributed algorithm to construct CVTs is Lloyd's method [8]. It is a simple, iterative, and distributed algorithm, derived from the locational optimization problem [9] that requires little a prior information on the region of interest. To the best of our knowledge, it is also the only distributed algorithm that has been applied to sensor networks for even deployment [10].

When the initial locations of all sensors and the boundaries of the area are available, each sensor can run Lloyd's method locally to compute the final CVT, and hence its final destination, and then move to the destination directly. This approach, however, may not always be feasible or desirable. First, a sensor may not know the initial locations of the other sensors. This is because sensors are often initially deployed by airdropping or being projected into the area, thereby making the initial sensor locations difficult to predict. Furthermore, due to the uncontrolled initial deployment, sensors may not form a connected network, making it difficult for a sensor to communicate its location to the other sensors. Second, the boundaries of the area may not be known beforehand and the sensors may have to sense the boundaries while moving in the area. Third, even if each sensor can pre-compute the final CVT, some mobile sensors may fail while moving to their final destinations, leading to an uneven deployment formed by the remaining sensors.

Due to the above reasons, in practice, a more robust and scalable way is to apply Lloyd's method iteratively while fully employing its distributed nature. Specifically, in each

- *Y. Song, B. Wang and Z. Shi are with the Department of Computer Science & Engineering, University of Connecticut, 371 Fairfield Way, U-4155 Storrs, CT 06269. E-mail: {yuan.song, bing, zshi}@engr.uconn.edu.*
- *K. R. Pattipati and S. Gupta are with the Department of Electrical & Computer Engineering, University of Connecticut, 371 Fairfield Way, U-4157 Storrs, CT 06269. E-mail: {krishna, shalabh.gupta}@engr.uconn.edu.*

*Manuscript received 22 June 2012; revised 4 Mar. 2013; accepted 25 Mar. 2013. Date of publication 10 Apr. 2013; date of current version 15 May 2014. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2013.46*

iteration, sensors update their neighbor information, sense unknown boundaries and compute their next destinations. In this way, sensors will form an even deployment eventually. Furthermore, sensor failures can be detected immediately and reflected in the following computations when sensors update the information from their neighbors.

Despite its scalability and robustness, Lloyd's method suffers from two critical issues when being used in mobile sensor networks. First, it relies on accurate Voronoi neighbor information (more details of Voronoi neighbors can be found in Section 2), which is often not available in mobile sensor networks since the sensors may be out of the communication ranges of their Voronoi neighbors. Second, iterative application of Lloyd's method is not energy efficient—it does not optimize sensor moving distances, and therefore the sensors may travel longer distances than necessary before reaching their desired destinations. Since mechanical movement of sensors is one of the dominant sources of energy consumption [11], they may waste a large amount of energy.

This paper addresses the issue of energy-efficient mobile sensor deployment to evenly cover a region of interest. The proposed method improves the energy efficiency of the iterative Lloyd's method by defining two cost metrics of energy consumption. The first one is the traveling distance of the sensors and the other one is the number of deployment steps, which roughly equals to the number of start/stop operations of each sensor. The latter has been shown to be another major energy consumption source during the deployment process [11]. The limited communication ranges of sensors are also considered.

The main contributions of the paper are as follows.

- The locational optimization problem is reformulated by incorporating the traveling distances of the sensors. The new formulation can achieve even deployment while taking account of the energy consumption.
- A new algorithm called Lloyd-$\alpha$ is proposed that reduces the movement step sizes in Lloyd's method and saves traveling distance while maintaining the convergence property.
- A new distributed algorithm, called Distributed Energy-Efficient Self-Deployment (DEED) algorithm, is proposed, that reduces the energy consumption by saving sensor traveling distances while maintaining a reasonable number of deployment steps.
- An intuitive method is proposed to deal with the incomplete Voronoi neighbor information due to limited communication ranges of sensors. Simulation results show that this method helps the convergence of both Lloyd's method (the original Lloyd's method and Lloyd-$\alpha$) and DEED algorithm.
- The performance of DEED algorithm is evaluated using both analysis and simulations. Extensive simulation results indicate that, compared to Lloyd's method, it reduces the traveling distance by up to 54 percent, and reduces the energy consumption by up to 46 percent.

The rest of the paper is organized as follows. Section 2 presents background information. Section 3 describes our new formulation of the locational optimization problem. Section 4 describes DEED and its theoretical analysis. Section 5 describes the method to deal with the incomplete neighbor information in Voronoi cell computation. Section 6 presents performance evaluation. Finally, Section 7 concludes the paper and presents future work.

## 2 Background

### 2.1 Voronoi Diagram and CVT

Given a region and a set of sensors, a Voronoi diagram divides the region into a set of Voronoi cells; each point in a cell is associated with its closest sensor. Specifically, consider a convex region, $A$, with a density function $\rho(x)$. Let $s_i$ denote the location of sensor $i$. Let vector $\mathbf{s} = [s_i]_{i=1}^N$ denote the location of all the sensors. The Voronoi cells, $\mathcal{V}(\mathbf{s}) = \{\mathcal{V}_i\}_{i=1}^N$, generated by $\mathbf{s}$ are defined as

$$\mathcal{V}_i = \{x \in A \,|\, \|x - s_i\| \leq \|x - s_j\|, \forall j \neq i\},$$

where $\mathcal{V}_i$ is the Voronoi cell generated by the $i$th sensor. All the sensors whose Voronoi cells are adjacent to $\mathcal{V}_i$ are called as the *Voronoi neighbors* of sensor $i$. Fig. 1a shows an example of a Voronoi tessellation generated by 16 sensors.

Note that due to limited communication ranges of sensors, the Voronoi neighbors of a sensor may not be its real neighbors defined by its communication range. For instance, in Fig. 1a, sensor 6 has six Voronoi neighbors. Due to limited communication range, only three of them (sensors 7, 8, and 5) are within the communication range of sensor 6, while the other three (sensor 4, 12, 16) are out of the communication range of sensor 6.

A CVT is a Voronoi tessellation where each generator of its Voronoi cells coincides with the mass centroid of the Voronoi cell. Fig. 1b shows a CVT corresponding to the Voronoi tessellation in Fig. 1a. It is obtained using Lloyd's method (see Section 2.3). Many CVTs may be derived from the initial deployment in Fig. 1a; Fig. 1b only illustrates one of them.

### 2.2 Optimization Problem for Even Self-Deployment

Consider the *CVT energy function* [7], [12] defined as

$$\mathcal{F}(\mathbf{s}) = \sum_{i=1}^{N} \int_{x \in \mathcal{V}_i} \|x - s_i\|^2 \rho(x)\, dx. \tag{1}$$

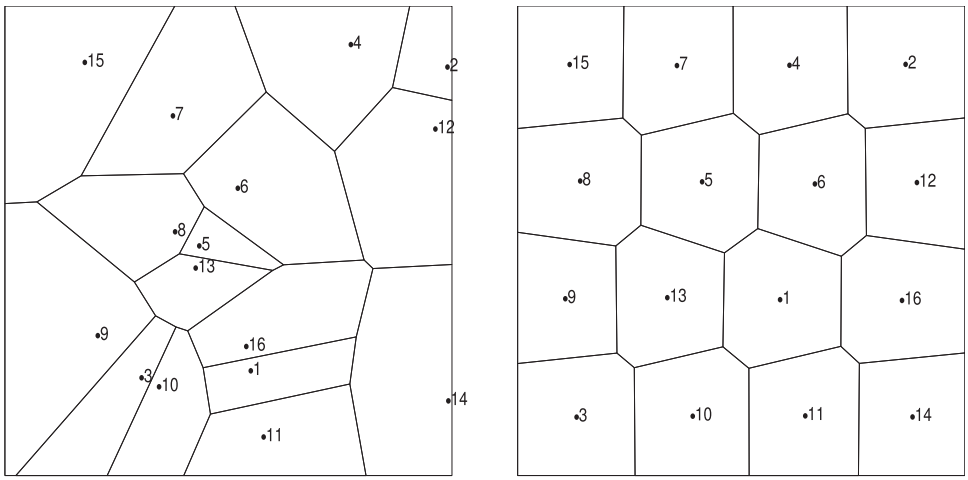
As shown in [7], [13], [14], the gradient of $\mathcal{F}(\mathbf{s})$ is

$$\nabla \mathcal{F}(s_i) = \frac{\partial F}{\partial s_i} = 2m_i(s_i - c_i), \tag{2}$$

where $m_i$ and $c_i$ are the mass and centroid of Voronoi cell $\mathcal{V}_i$, respectively, and

$$m_i = \int_{x \in \mathcal{V}_i} \rho(x)\, dx,$$
$$c_i = \frac{\int_{x \in \mathcal{V}_i} \rho(x) x\, dx}{\int_{x \in \mathcal{V}_i} \rho(x)\, dx},$$

where $\rho(x)$ denotes the density at point $x$. In our problem, $\rho(x) \equiv 1$.




(a) Voronoi Cells (b) CVT

Fig. 1. Illustration of Voronoi Cells and CVT with 16 sensors in a square area.

One can observe from (2) that a CVT corresponds to a critical point of $\mathcal{F}(\mathbf{s})$. The study in [7] shows that the necessary condition for $\mathcal{F}(\mathbf{s})$ to be minimized is that $\mathbf{s}$ forms a CVT. Recall that a CVT corresponds to an even deployment according to Gersho's conjecture [6]. Hence the sensor locations that minimize the CVT energy function $\mathcal{F}$ form an even deployment. Therefore, the even-deployment problem can be formulated as an unconstrained minimization problem as

$$\mathbf{s}_r = \arg\min_{\mathbf{s}} \mathcal{F}(\mathbf{s}), \tag{3}$$

where $\mathbf{s}_r$ corresponds to a CVT of the target area. Note that for a given set of sensors and target area, there may exist multiple CVTs which correspond to the local minimizers and the global minimizer of the CVT energy function [7]. Any CVT can form an even deployment.

To solve the even self-deployment problem defined in (3), iterative algorithms are generally used. In these algorithms, the locations of sensors are iteratively updated in the direction of the negative gradient of the CVT energy function, until the algorithm converges. If the algorithm is distributed, then the sensors compute the moving direction with the local information and move iteratively to minimize the CVT energy function. In particular, let $\mathbf{s}_k$ denote sensor positions in the $k$th iteration. Let the movement step of $N$ sensors be defined by the vector $\mathbf{p} = [p_i]_{i=1}^{N}$. Then, the movement step $\mathbf{p}_k$ in the $k$th iteration is computed as,

$$\mathbf{p}_k = \arg\min_{\mathbf{p}} \mathcal{F}(\mathbf{s}_k + \mathbf{p}), \tag{4}$$

and the sensor location vector is then updated as

$$\mathbf{s}_{k+1} = \mathbf{s}_k + \mathbf{p}_k. \tag{5}$$

Due to the iterative nature of the method, the sensors are required to be synchronized with each other in some fashion. The algorithm converges to a CVT when $\mathbf{p}_k = 0$.

### 2.3 Lloyd's Method

A widely used algorithm to construct a CVT is Lloyd's method [7]. In Lloyd's method, the sensors' locations are updated to the centroids of their Voronoi cells in each iteration. Subsequently, the Voronoi cells are computed again and the process is iterated until an approximate CVT of the target area is generated.[1] Since sensors can use distributed algorithms (e.g., those in [15]) to compute Voronoi cells and Voronoi neighbors, the movement step can be computed distributedly based on the neighbor information. To apply it to mobile sensor deployment, we let each iteration consist of two phases: i) neighbor discovery phase and ii) movement phase. In the neighbor discovery phase, sensors exchange their location information with their neighbors. At the end of the neighbor discovery phase, sensors compute their Lloyd movement step, referred to as *Lloyd step*, as

$$p_i = c_i - s_i, \quad i = 1, \ldots, N. \tag{6}$$

Note that a sensor may not be within the communication ranges of its Voronoi neighbors. The method described in Section 5 can be used to deal with such cases.

## 3 Energy-Efficient Even Self-Deployment

This section formulates a locational optimization problem that achieves even deployment while taking into account sensor traveling distances. Subsequently, it is shown that Lloyd's method provides an approximate solution to this problem.

### 3.1 Problem Statement

In order to save energy consumption, it is essential to reduce the traveling distances of sensors during self-deployment. In this regard, the iterative form in Eq. (4) is modified by adding a penalty function for the lengths of sensors' movement steps in each iteration. More specifically, the desired energy-efficient movement step in the $k$th iteration becomes

1. Note that in [10] Lloyd's method can be an asynchronous algorithm in which sensors compute movement step and change destination on the go. However, this requires the sensors to update Voronoi cell continuously, which is not practical in sensor networks.

$$\mathbf{p}_k = \arg\min_{\mathbf{p}} \left( \mathcal{F}(\mathbf{s}_k + \mathbf{p}) + \frac{1}{2}\mathbf{p}^T \Gamma_k \mathbf{p} \right), \tag{7}$$

where the second term on the right hand side, $\frac{1}{2}\mathbf{p}^T\Gamma_k\mathbf{p}$, represents the penalty function and $\Gamma_k$ is a diagonal matrix with positive elements.

The above method can be regarded as a proximal minimization algorithm [16]. When the algorithm converges, the sensor location vector $\mathbf{s}$ converges to the same minimizer as that of the energy function $\mathcal{F}(\mathbf{s})$, and hence forms a CVT. The larger the elements of $\Gamma_k$ are, the smaller are the movements steps and more distance saving is expected. However, as we will see, larger elements in $\Gamma_k$ also result in a larger number of deployment steps.

The introduction of the second order term $\mathbf{p}^T\Gamma_k\mathbf{p}$ in Eq. (7) requires that the iterative gradient method employs the second order information of $\mathcal{F}(\mathbf{s})$ to characterize the movement step $\mathbf{p}_k$. This leads to the application of Newton's method [17].[2] To solve for $\mathbf{p}_k$, $\mathcal{F}(\mathbf{s}_k + \mathbf{p})$ is approximated using Taylor's theorem as

$$\mathcal{F}(\mathbf{s}_k + \mathbf{p}) \approx \mathcal{F}(\mathbf{s}_k) + \mathbf{g}(\mathbf{s}_k)^T\mathbf{p} + \frac{1}{2}\mathbf{p}^T H(\mathbf{s}_k)\mathbf{p}, \tag{8}$$

where $H(\mathbf{s}_k)$ and $\mathbf{g}(\mathbf{s}_k)$ are the Hessian matrix and the gradient vector of $\mathcal{F}$ at $\mathbf{s}_k$, respectively. Substituting Eq. (8) into Eq. (7) yields

$$\mathbf{p}_k \approx \arg\min_{\mathbf{p}} \left( \mathcal{F}(\mathbf{s}_k) + \mathbf{g}(\mathbf{s}_k)^T\mathbf{p} + \frac{1}{2}\mathbf{p}^T(H(\mathbf{s}_k) + \Gamma_k)\mathbf{p} \right). \tag{9}$$

Setting the derivative of the argument in the right hand side of Eq. (9) to zero, we obtain the energy-efficient movement step (EE-step) as

$$\mathbf{p}_k = -(H(\mathbf{s}_k) + \Gamma_k)^{-1}\mathbf{g}(\mathbf{s}_k). \tag{10}$$

Computing the EE-step via Eq. (10) requires the computation of the Hessian matrix and its inverse, which in general needs global information of sensor locations. As will be shown later, this computation can be done via cooperation among sensors in a distributed manner in mobile sensor networks.

### 3.2 Lloyd Step versus EE-Step

This section shows that the movement step in Lloyd's method is an approximation to the EE-step with a non-optimal choice of $\Gamma_k$. Furthermore, decreasing the step size of Lloyd's method leads to less traveling distance while maintaining the convergence property.

Note that if the Hessian matrix $H$ is a diagonal matrix, $\mathbf{p}_k$ can be computed distributedly. Let $H'$ denote a diagonal matrix that only contains the diagonal elements of $H$. Approximating $H$ using $H'$ yields

$$\mathbf{p}_k \approx -(H'(\mathbf{s}_k) + \Gamma_k)^{-1}\mathbf{g}(\mathbf{s}_k). \tag{11}$$

2. Newton's method requires $\mathcal{F}$ to be at least $C^2$ [17]. A recent study [12] proved $C^2$ smoothness of $\mathcal{F}$ in any convex and most non-convex 2D domains.

Following the results in [13], [14], the $i$th element in the diagonal part of $H$ is $2m_i - \theta_i$, where $m_i$ is the mass of the Voronoi cell $\mathcal{V}_i$ in the $k$th iteration, and $\theta_i$ is a positive number (the exact form is described in [13], [14]). Setting the $i$th diagonal element of $\Gamma_k$ to $\theta_i$, the movement step $p_i$ of the $i$th sensor in the $k$th iteration becomes

$$p_i = -(2m_i - \theta_i + \theta_i)^{-1}\nabla\mathcal{F}(s_i), \quad i = 1, \ldots, N. \tag{12}$$

Substituting Eq. (2) into the above yields

$$p_i = -(2m_i)^{-1}2m_i(s_i - c_i) = c_i - s_i, \quad i = 1, \ldots, N \tag{13}$$

which is equivalent to the movement step in Lloyd's method as shown in Eq. (6). The above implies that the movement step in Lloyd's method is an approximation of the EE-step—it is obtained by approximating the Hessian matrix by keeping only the diagonal elements and choosing a specific form of $\Gamma_k$ (i.e., setting the $i$th diagonal element of $\Gamma_k$ to $\theta_i$). This choice of $\Gamma_k$ may not be optimal. In fact, we can expect to save more distance by increasing the diagonal elements of $\Gamma_k$. For example, if we increase the $i$th diagonal elements in $\Gamma_k$ to $(2m_i + \theta_i)$, then the movement step becomes

$$p_i = \frac{1}{2}(c_i - s_i), \quad i = 1, \ldots, N. \tag{14}$$

Equivalently,

$$\mathbf{p}_k \approx \frac{1}{2}(\mathbf{c}_k - \mathbf{s}_k), \tag{15}$$

which is half of the movement step in Lloyd's method.

### 3.3 Lloyd-$\alpha$ Method

By choosing $\Gamma_k$ appropriately, we can set the EE-step to an arbitrary fraction, $\alpha$, of Lloyd step. This method is referred to as Lloyd-$\alpha$ method. The convergence of the method is shown in Proposition 3.1.

**Proposition 3.1.** *Suppose the movement step of $N$ sensors in Lloyd's method is given by a vector $\mathbf{p} = [p_i]_{i=1}^N$. If the sensors take a movement step $\mathbf{p}' = [p_i']_{i=1}^N$ satisfying that $p_i' = \alpha_i p_i$, $\alpha_i \in (0, 1)$, $\forall i \in \{1, \ldots, N\}$ in all iterations, $\mathbf{k} = 1, 2, \ldots$, then the sequence of sensor positions $\mathbf{s}_1, \mathbf{s}_2, \ldots\ \mathbf{s}_k, \ldots$ converges to a CVT of the target area.*

**Proof.** Let $\mathcal{T} = \{\mathcal{T}_i\}_{i=1}^N$ denote an arbitrary tessellation of $N$ sensors in the area. We define

$$\mathcal{F}(\mathbf{s}, \mathcal{T}) = \sum_{i=1}^{N} \int_{x \in \mathcal{T}_i} \|x - s_i\|^2 \rho(x)\, dx, \tag{16}$$

when $\mathcal{T} = \mathcal{V}$, we obtain energy function (7). With the property of Voronoi tessellation, we have

$$\mathcal{F}(\mathbf{s}, \mathcal{V}) \leq \mathcal{F}(\mathbf{s}, \mathcal{T}), \tag{17}$$

with strict inequality if $\mathcal{T} \neq \mathcal{V}$. Let $\mathbf{c} = [c_i]_{i=1}^N$ and $\mathbf{m} = [m_i]_{i=1}^N$ denote the centroids and mass of $\mathcal{V}$ respectively. Applying the parallel axis theorem [10], we can rewrite $\mathcal{F}(\mathbf{s}, \mathcal{T})$ as

$$\mathcal{F}(\mathbf{s},\mathcal{T}) = \sum_{i=1}^{N} \int_{x\in\mathcal{T}_i} \|x-c_i\|^2 \rho(x)\,dx + m_i\|s_i - c_i\|. \quad (18)$$

Let $\mathbf{s}' = [s_i']_{i=1}^N$ denote the new positions of sensors in the beginning of next iteration, i.e., $\mathbf{s}' = \mathbf{s} + \mathbf{p}'$. Since $p_i' = \alpha_i p_i = \alpha_i(c_i - s_i), \alpha_i \in (0,1)$, we have $\|s_i' - c_i\| = (1-\alpha_i)\|s_i - c_i\| < \|s_i - c_i\|$. Together with equation (18), we have

$$\mathcal{F}(\mathbf{s}',\mathcal{T}) \le \mathcal{F}(\mathbf{s},\mathcal{T}), \quad (19)$$

with strict inequality if $\mathbf{s}$ is not a CVT. Let $\mathcal{V}'$ denote the Voronoi tessellation formed by $\mathbf{s}'$, from equations (17) and (19), we have

$$\mathcal{F}(\mathbf{s}',\mathcal{V}') \le \mathcal{F}(\mathbf{s}',\mathcal{V}) \le \mathcal{F}(\mathbf{s},\mathcal{V}). \quad (20)$$

The above equation shows that new positions of sensors will decrease the energy function until sensors form a CVT, which concludes the proof. □

Simulation results show that compared to the original Lloyd's method, Lloyd-$\alpha$ method using partial step sizes saves traveling distance. However, it requires a larger number of deployment steps. For example, if half of the movement step of Lloyd's method is used, i.e., $\alpha = 0.5$, then the number of deployment steps is approximately doubled. Thus, the excessive energy consumed in start/stop operations may cancel out the energy saved in the traveling distance. This limits the application of this algorithm only to the scenarios in which the cost of start/stop operations is relatively low. The study in [12] shows that incorporating more information from the second order term of the CVT energy function, i.e., the Hessian matrix, can achieve significantly less iterations before convergence (corresponding to less deployment steps in our context). This motivates us to propose a new distributed algorithm, as described in Section 4, that carefully chooses $\Gamma_k$ to reduce both the number of deployment steps and sensor traveling distances.

## 4 Distributed Energy Efficient Self-Deployment Algorithm

This section presents a new algorithm, named *Distributed Energy Efficient self-Deployment*. It is an iterative algorithm where in each iteration a sensor moves according to the EE-step. It has two main objectives: i) saving the traveling distances of the sensors and ii) reducing the number of deployment steps. To achieve these two objectives, $\Gamma_k$ needs to be chosen carefully. In the following, we first describe how to choose $\Gamma_k$, and then describe how to compute EE-step in a distributed manner. In the end, we present the workflow of DEED algorithm.

### 4.1 Choice of $\Gamma_k$

Recall that the EE-step is a movement step that minimizes the CVT energy function $\mathcal{F}$ in Eq. (1) such that the EE-step lies in the direction of decreasing $\mathcal{F}$. This implies that the matrix $H(\mathbf{s}_k) + \Gamma_k$ in Eq. (10) needs to be positive definite [17]. Since the Hessian matrix $H$ is sparse and is generally not positive definite, the elements of $\Gamma_k$ should be large enough to make the matrix $H(\mathbf{s}_k) + \Gamma_k$ positive definite. Reference [17] suggests that $\Gamma_k$ can take the form of $\epsilon_k \mathbf{I}$, where $\epsilon_k$ is a constant. Furthermore, if $\epsilon_k$ satisfies that $\epsilon_k = 0$ if the Hessian is positive definite and $\epsilon_k = -\lambda_{\min}(H) + \delta$ if the Hessian is positive indefinite, where $\lambda_{\min}(H)$ denotes the the minimum eigenvalue of $H$ and $\delta > 0$ is a small positive constant, then $\Gamma_k = \epsilon_k \mathbf{I}$ is the matrix with the minimum euclidean norm that makes $H(\mathbf{s}_k) + \Gamma_k$ positive definite [17]. The study of [12] demonstrates this choice of $\epsilon_k$ is effective in reducing the number of iterations. This choice of $\epsilon_k$, however, may not be favorable in saving sensor traveling distances. Indeed in the above, $\epsilon_k$ takes small values so that $H(\mathbf{s}_k) + \Gamma_k$ is close to $H(\mathbf{s}_k)$ in order to reduce the number of iterations (i.e., deployment steps in our context) [12], [17], while to save traveling distances, larger elements in $\Gamma_k$ are preferable as described in Section 3.2. To achieve the dual objectives of reducing the number of deployment steps as well as saving sensor traveling distances, we set $\Gamma_k = \epsilon_k \mathbf{I}$ and choose the value of $\epsilon_k$ as follows.

Let $a_{ij}$ denote the element on the $i$th row and $j$th column of the Hessian matrix $H(\mathbf{s}_k)$ in the $k$th iteration. Then the value of $\epsilon_k$ is chosen as

$$\epsilon_k = \max\left\{0, -\min_i \left\{ a_{ii} - \sum_j |a_{ij}| \right\}\right\} + \delta, \quad (21)$$

where $\delta > 0$ is a positive constant. If $H_+$ is defined as

$$H_+ = H(\mathbf{s}_k) + \Gamma_k = H(\mathbf{s}_k) + \epsilon_k \mathbf{I}, \quad (22)$$

then it can be shown that the choice of $\epsilon_k$ in Eq. (21) makes the matrix $H_+$ positive definite as stated in the following lemma.

**Lemma 1.** *When setting $\epsilon_k$ as in Eq. (21), $\lambda_{\min}(H_+) \ge \delta$, where $\lambda_{\min}(H_+)$ denotes the minimum eigenvalue of $H_+$. In addition, the matrix $H_+$ is positive definite.*

**Proof.** We first prove the first statement. Recall Gerschgorin Theorem [18]. Let matrix $B \in \mathbb{R}^{n\times n}$ be symmetric with eigenvalues $\lambda_1, \ldots, \lambda_n$. Then

$$\min_{1\le i\le n} \lambda_i \ge \min_{1\le i\le n} \left\{ b_{ii} - \sum_{j=1, j\ne i}^{n} |b_{ij}| \right\},$$

where $b_{ij}$ is the element of $B$ on the $i$th row and $j$th column.

Let $a_{ij}$ denotes the element of $H$ on $i$th row and $j$th column. let $u_j = \sum_{j=1, j\ne i}^{n} |a_{ij}|$. Since $H_+ = H + \epsilon \mathbf{I}$, according to Gerschgorin Theorem, we have

$$\begin{aligned}\lambda_{\min}(H_+) &\ge \min_{1\le i\le n} \{\epsilon + a_{ii} - u_j\} \\ &= \Big(\epsilon + \min_{1\le i\le n} \{a_{ii} - u_j\}\Big),\end{aligned}$$

i. if $\min_{1\le i\le n}\{a_{ii} - u_j\} < 0$, we have $\epsilon = -\min_{1\le i\le n} \{a_{ii} - u_j\} + \delta$ and

$$\lambda_{\min}(H_+) \geq \delta,$$

ii. if $\min_{1\leq i\leq n}\{a_{ii}-u_j\} \geq 0$, we have $\epsilon = \delta$ and

$$\lambda_{\min}(H_+) \geq \min_{1\leq i\leq n}\{a_{ii}-u_j\} + \delta \geq \delta.$$

The second statement that $H_+$ is a positive definite matrix follows directly from above and the symmetry of $H$. □

Setting $\epsilon_k$ as in Eq. (21) has the following three advantages compared to the choice of $\epsilon_k$ in [12]. First, to save sensor traveling distance, the value of $\epsilon_k$ should not be too small. When $H$ is positive indefinite, $\epsilon_k$ in Eq. (21) is always larger than $-\lambda_{\min}(H)$ (see the proof of Lemma 1), and hence is more desirable in saving sensor traveling distance. Second, the matrix $H_+$ and $\lambda_{\min}(H)$ are difficult to evaluate in a distributed manner. The widely adopted methods that compute the minimum modification to make the Hessian matrix positive definite (e.g., the modified Cholesky factorization [17], [19]) cannot be employed since they use centralized algorithms and have high complexities. In contrast, computing $\epsilon_k$ is easy to implement on distributed sensors. It needs global consensus but requires much less overheads. Third, as to be shown in Theorem 1, such choice of $\epsilon_k$ leads to the convergence of the Jacobi method which is the method used to distribute the computation.

Note that if a sensor is not within the communication ranges of its Voronoi neighbors or the network is disconnected, the Hessian matrix $H$ and $\epsilon_k$ may not be computed correctly. The method described in Section 5 can be used to deal with such cases.

The local convergence property of applying the EE-step on a general function is shown by the following proposition.

**Proposition 4.1.** *Let the function $\mathcal{F}(\mathbf{s}_k)$ satisfy the condition that its Hessian matrix $H$ is locally Lipschitz continuous around an optimal point set $\mathbf{s}^*$. Furthermore, let the sequence $\{\mathbf{s}_k\}$ and all elements of $H$ be bounded in a finite domain. Then if the starting point $\mathbf{s_0}$ is sufficiently close to the optimal set $\mathbf{s}^*$, then the sequence of iterative points generated by the solution of Eqs. (10) and (21) converges to $\mathbf{s}^*$, and the rate of convergence is at least linear.*

Before proving Proposition 4.1, we first prove a lemma.

**Lemma 2.** *Define $\mathbf{s}^*$ as the optimal point of function $\mathcal{F}(\mathbf{s})$ where $\mathbf{g}(\mathbf{s}^*) = 0$. Consider any functions $\mathcal{F}(\mathbf{s})$ that is twice differentiable and whose Hessian matrix $H(\mathbf{s})$ is locally Lipschitz continuous with Lipschitz constant $L$ around optimal point set $\mathbf{s}^*$ within region $\|\mathbf{s}-\mathbf{s}^*\| \leq r$. Do iterations as $\mathbf{s}_{k+1} = \mathbf{s}_k + \mathbf{p}_k$, where $\mathbf{p}_k$ is computed by $\mathbf{p}_k = -H_+^{-1}(\mathbf{s}_k)\mathbf{g}(\mathbf{s}_k)$ and $H_+ = H(\mathbf{s}_k) + \epsilon_k\mathbf{I}$. Then, if the starting point $\mathbf{s}_0$ satisfies the condition $\|\mathbf{s}_0-\mathbf{s}^*\| \leq r$, the sequence of iterates $\mathbf{s}_k$ follows inequality below*

$$\|\mathbf{s}_{k+1}-\mathbf{s}^*\| \leq \frac{L}{2\delta}\big(\|\mathbf{s}_k-\mathbf{s}^*\|^2 + 2\epsilon_k\|\mathbf{s}_k-\mathbf{s}^*\|\big).$$

**Proof.** We denote $\mathbf{g}(\mathbf{s}_k)$ and $\mathbf{g}(\mathbf{s}^*)$ as $\mathbf{g}_k$ and $\mathbf{g}^*$ respectively. From the definition of the iteration step and the optimality condition $\mathbf{g}^* = 0$, we have

$$\begin{aligned}
&\mathbf{s}_{k+1} - \mathbf{s}^* \\
&= \mathbf{s}_k + \mathbf{p}_k - \mathbf{s}^* \\
&= \mathbf{s}_k - \mathbf{s}^* - (H(\mathbf{s}_k) + \epsilon_k\mathbf{I})^{-1}\mathbf{g}_k \\
&= (H(\mathbf{s}_k) + \epsilon_k\mathbf{I})^{-1} \\
&\quad \big[(H(\mathbf{s}_k) + \epsilon_k\mathbf{I})\,(\mathbf{s}_k - \mathbf{s}^*) - (\mathbf{g}_k - \mathbf{g}^*)\big].
\end{aligned} \tag{23}$$

According to Taylor's theorem [17],

$$\mathbf{g}_k - \mathbf{g}^* = \int_0^1 H(\mathbf{s}_k + t(\mathbf{s}^* - \mathbf{s}_k))(\mathbf{s}_k - \mathbf{s}^*)dt. \tag{24}$$

Then choosing $\mathbf{s}_0$ so that $\|\mathbf{s}-\mathbf{s}^*\| \leq r$, for $k \geq 0$ we have

$$\begin{aligned}
&\|(H(\mathbf{s}_k) + \epsilon_k\mathbf{I})(\mathbf{s}_k - \mathbf{s}^*) - (\mathbf{g}_k - \mathbf{g}^*)\| \\
&= \left\| \int_0^1 [H(\mathbf{s}_k) + \epsilon_k\mathbf{I} - H(\mathbf{s}_k + t(\mathbf{s}^* - \mathbf{s}_k))](\mathbf{s}_k - \mathbf{s}^*)dt \right\| \\
&\leq \int_0^1 (\|H(\mathbf{s}_k) - H(\mathbf{s}_k + t(\mathbf{s}^* - \mathbf{s}_k))\| + \epsilon_k)\|\mathbf{s}_k - \mathbf{s}^*\|dt \\
&\leq \int_0^1 (Lt\|\mathbf{s}_k - \mathbf{s}^*\| + \epsilon_k)\|\mathbf{s}_k - \mathbf{s}^*\|dt \\
&= \frac{1}{2}L\|\mathbf{s}_k - \mathbf{s}^*\|^2 + \epsilon_k\|\mathbf{s}_k - \mathbf{s}^*\|.
\end{aligned} \tag{25}$$

Let $\lambda_{\max}(H)$ denote maximum eigenvalue of $H$.

$$\begin{aligned}
\|(H(\mathbf{s}_k) + \epsilon_k\mathbf{I})^{-1}\| &= \lambda_{\max}\big((H(\mathbf{s}_k) + \epsilon_k\mathbf{I})^{-1}\big) \\
&= \frac{1}{\lambda_{\min}(H(\mathbf{s}_k) + \epsilon_k\mathbf{I})}.
\end{aligned}$$

Since Lemma 1 tells us that $\lambda_{\min}(H_+) \geq \delta$, we have

$$\|(H(\mathbf{s}_k) + \epsilon_k\mathbf{I})^{-1}\| \leq \frac{1}{\delta}. \tag{26}$$

By substituting in (23) and (25), we obtain

$$\|\mathbf{s}_{k+1} - \mathbf{s}^*\| \leq \frac{L}{2\delta}\big(\|\mathbf{s}_k - \mathbf{s}^*\|^2 + 2\epsilon_k\|\mathbf{s}_k - \mathbf{s}^*\|\big). \tag{27}$$

□

We now prove Proposition 4.1.

**Proof.** Let $L$ denote the local Lipschitz constant of $H$. Since all elements in $H$ are bounded. Thus the value of $\epsilon$ computed by (21) should be upper bounded. We denote this upper bound as $\overline{U}_\epsilon$. Since the sequence $\{\mathbf{s}_k\}$ are all bounded in a finite domain, $\|\mathbf{s}_k - \mathbf{s}^*\|$ should also be upper bounded. We denote the upper bound as $\overline{U}_d$. Then, following from Lemma 2, we have,

$$\begin{aligned}
\|\mathbf{s}_{k+1} - \mathbf{s}^*\| &\leq \frac{L}{2\delta}\big(\|\mathbf{s}_k - \mathbf{s}^*\|^2 + 2\epsilon\|\mathbf{s}_k - \mathbf{s}^*\|\big) \\
&\leq \widetilde{U}\|\mathbf{s}_k - \mathbf{s}^*\|,
\end{aligned}$$

where $\widetilde{U} = \frac{L(\overline{U}_d + 2\overline{U}_\epsilon)}{2\delta}$. Choosing $\mathbf{s}_0$ so that $\|\mathbf{s}_0 - \mathbf{s}^*\| \leq 1/(2\widetilde{U})$, we can inductively deduce that the sequence converges to $\mathbf{s}^*$, and the rate of convergence is linear. □

It is seen from the proof of Proposition 4.1 that the local convergence rate depends on $\epsilon_k$. A large $\epsilon_k$ will result in a linear rate of convergence which leads to a larger number of deployment steps, while a small $\epsilon_k$ results in nearly quadratic rate of convergence which leads to a smaller number of deployment steps.

### 4.2 Distributed Realization

This section describes how to compute the EE-steps distributively among sensors. From Eq. (10) and Eq. (22), it is observed that the EE-step, $\mathbf{p}_k$, is the solution to a system of linear equations

$$H_+ \mathbf{p}_k = -\mathbf{g}(\mathbf{s}_k). \tag{28}$$

Therefore, we can use distributed iterative methods such as the Jacobi method [20], [21] or the GaBP method [22] to obtain $\mathbf{p}_k$. In this paper, the Jacobi method is chosen since it is simpler and more suitable for distributed sensor network applications. The Jacobi method decomposes $H_+$ into two matrices, a diagonal matrix $\mathcal{D}$ and a remainder matrix $\mathcal{R}$, i.e., $H_+ = \mathcal{D} + \mathcal{R}$. Then Eq. (28) is rewritten as

$$(\mathcal{D} + \mathcal{R})\mathbf{p}_k = -\mathbf{g}(\mathbf{s}_k).$$

Multiplying $\mathcal{D}^{-1}$ on both sides yields

$$(I + \mathcal{D}^{-1}\mathcal{R})\mathbf{p}_k = -\mathcal{D}^{-1}\mathbf{g}(\mathbf{s}_k).$$

The Jacobi method uses an iterative approach. More specifically, let $\mathbf{p}_k(t)$ denote the solution obtained at the $t$th iteration. Then,

$$\mathbf{p}_k(t+1) = -\mathcal{D}^{-1}(\mathcal{R}\mathbf{p}_k(t) + \mathbf{g}(\mathbf{s}_k)). \tag{29}$$

Note that $\mathcal{D} = \mathcal{D}' + \epsilon_k \mathbf{I}$ where $\mathcal{D}'$ is the diagonal part of the Hessian matrix $H(\mathbf{s}_k)$. Thus, we get

$$\mathbf{p}_k(t+1) = -(\mathcal{D}' + \epsilon_k \mathbf{I})^{-1}(\mathcal{R}\mathbf{p}_k(t) + \mathbf{g}(\mathbf{s}_k)). \tag{30}$$

We observe two facts from the explicit formula of Hessian $H$ [13], [14]. First, $H$ is a sparse matrix in the sense that if two sensors are not Voronoi neighbors, then the elements corresponding to these two sensors are zeros. Second, if the location information of all the Voronoi neighbors is given, then each sensor can compute all the elements of its corresponding two rows[3] of $H$ and thus $\mathcal{D}'$ and $\mathcal{R}$.

Therefore, if the Voronoi neighbors can share the information of their movement steps computed in iteration $t$ via communication and since $\mathbf{g}(\mathbf{s}_k)$ can be computed distributively, the Jacobi iterative process in Eq. (30) can be carried out in a distributed manner, though it needs more communication overheads. The Jacobi iterative process converges to the solution of Eq. (28) when $H_+$ is strictly diagonally dominant [20]. The following lemma states that this condition holds if the choice of $\epsilon_k$ follows Eq. (21);

**Lemma 3.** *$H_+$ is a strictly diagonally dominant matrix if $\epsilon_k$ is computed as in Eq. (21).*

**Proof.** According to equation (21),

- if $H$ is strictly diagonally dominant itself, $\epsilon = \delta$ and $H_+ = H + \epsilon\mathbf{I}$ is still diagonally dominant since $\delta > 0$.
- if $H$ is not strictly diagonally dominant, i.e., there is at least one row $k$ in the $H$ satisfying $a_{kk} - \sum_i |a_{ki}| \leq 0, -\min_i\{a_{ii} - \sum_j |a_{ij}|\} \geq 0$ and $\epsilon = -\min_i\{a_{ii} - \sum_j |a_{ij}|\} + \delta$. It is trivial to show that for any row $k$ that satisfies $a_{kk} - \sum_i |a_{ki}| \leq 0$, $\epsilon = -\min_i\{a_{ii} - \sum_j |a_{ij}|\} + \delta > a_{kk} - \sum_i |a_{ki}|$ since $\delta > 0$. Hence, for those rows, we have $\epsilon + a_{kk} > \sum_i |a_{ki}|$ which makes $H_+$ strictly diagonally dominant. □

Combining Lemma 3 and the results in [20], we have the following theorem regarding the convergence result of Jacobi method.

**Theorem 1.** *The Jacobi iterative process in Eq. (30) converges to the solution of Eq. (28) from any initial step vector and the approximate number of steps needed for convergence is*

$$-\frac{1}{\ln(\lambda_{\max}(\mathcal{D}^{-1}\mathcal{R}))},$$

*where $\lambda_{\max}(\mathcal{D}^{-1}\mathcal{R})$ is the largest eigenvalue of the matrix $\mathcal{D}^{-1}\mathcal{R}$.*

#### 4.2.1 Choice of the Initial Value $\mathbf{p}(\mathbf{0})$

In order to reduce the number of iteration steps in the Jacobi method and thus the communication energy consumption of sensors, the initial value, $\mathbf{p}(\mathbf{0})$, needs to be chosen carefully to minimize the initial error. In our simulations in Section 6, the initial value is set to be the movement step computed using the Lloyd's method according to Eq. (6). In this way, sensors can compute the initial steps distributively and the choice leads to a smaller number of iterations ($\sim$7) in our simulations.

#### 4.2.2 Propagating $\epsilon_k$ over the Network

Note that the Jacobi process requires $\epsilon_k$ to be known ahead. Since the $\max$ operation in computing $\epsilon_k$ in Eq. (21) needs global information, this requires extra packet exchanges to propagate $\epsilon_k$ over the network which leads to excessive energy consumption. To address this issue, observe that the $\max$ operation is over Hessian rows. Thus, a simple method is proposed in which $\epsilon_k$ propagates together with the Jacobi iterations. In this method, a sensor does not need to know the value of $\epsilon_k$ which is the global max before the Jacobi iteration. Instead, in the first iteration of the Jacobi process, it computes $\epsilon_k$ following Eq. (21) which is its local max, i.e., max value computed from its corresponding two rows. Then starting from the first iteration, it always inserts the current local max to the packet used in Jacobi process for message exchange. Other sensors will update their local max to the larger one received. The method requires the sensors to send out their corresponding diagonal elements in Hessian for their neighbors to update their steps with the updated local max. In this fashion, the global max will

3. Note that since sensors move in two-dimensional space, there are two rows corresponding to each sensor in the Hessian matrix.

spread over the network together with the Jacobi packets without extra overhead. One should note that even when $\epsilon_k$ is not propagated over all the network, the way $\epsilon_k$ is computed and propagated does not violate the convergence property of Jacobi process shown in Theorem 1.

**Algorithm 1** One iteration of DEED algorithm at each sensor

▷*Discovery phase starts*
Discover neighbors and exchange locations with neighbors
▷*Discovery phase ends*
▷*Jacobi phase starts*
Compute the corresponding rows of the Hessian matrix and its Lloyd steps
**for all** *computation sub-phases* **do**
  ▷*Computation sub-phase starts*
  Delay a random period and exchange following information with neighbors
  - $\epsilon$ (local max)
  - diagonal elements on its row
  - Lloyd steps for the first computation sub-phase or computed EE-steps at the end of previous computation sub-phase

  **repeat** Listen to the information from neighbors
    **if** neighbor information received **then**
      Save received EE-steps of the neighbor
      **if** a larger $\epsilon$ is received **then**
        Update previous computed EE-steps and received steps using received $\epsilon$ and diagonal elements
        Apply received $\epsilon$ in future computation
      **end if**
    **end if**
  **until** information of all neighbors received or the end of the computation sub-phase is reached
  **if** fail to hear from any neighbor **then**
    Skip following computation sub-phases and set EE-steps to Lloyd steps
  **end if**
  Compute new EE-steps based on received EE-steps of neighbors using equation (30)
  ▷*Computation sub-phase ends*
**end for**
▷*Jacobi phase ends*
▷*Movement phase starts*
Move according to EE-steps computed in the Jacobi phase
▷*Movement phase ends*

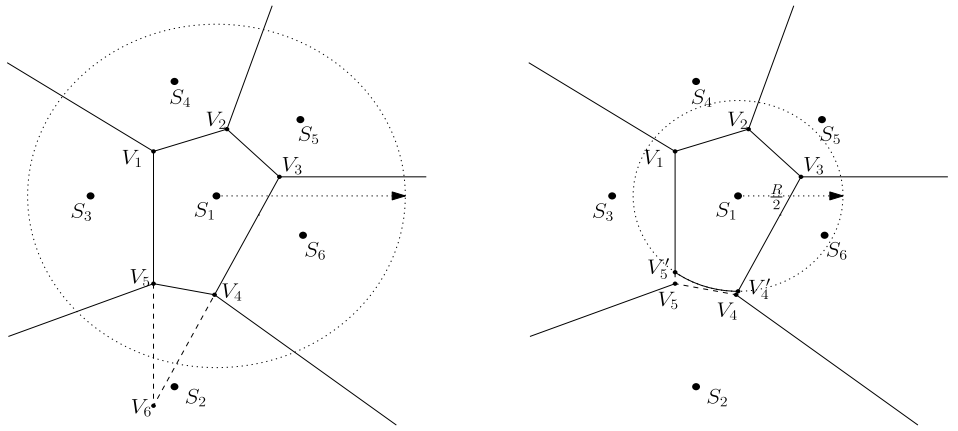
### 4.3 Workflow of DEED Algorithm

In DEED algorithm, sensors move in iterations. The workflow of one iteration of the DEED algorithm at each sensor is described in Algorithm 1. Each iteration starts with a neighbor discovery phase, followed by a Jacobi phase to perform Jacobi computations and finally a movement phase. The Jacobi phase is separated from neighbor discovery phase as the Hessian information can only be known after the neighbor discovery phase. The Jacobi phase is further divided into a predefined number of smaller "computation" sub-phases, each of which corresponds to an iteration in the Jacobi iterative process. In each Jacobi iteration, a new EE-step is computed using equation (30). Then in the following movement phase, sensors take the movement step as computed in the Jacobi phase. If a sensor fails to hear from one of its neighbors in any computation sub-phase, it simply takes a Lloyd movement step instead.

## 5 Computing Voronoi Cells with Limited Communication Range

The distributed Voronoi cell computation relies on the Voronoi neighbor information [15]. However, in mobile sensor networks, the communication ranges of sensors are limited. A sensor may not be within the communication ranges of its Voronoi neighbors. As a consequence, the distributed computation of Voronoi cells may not be feasible. Fig. 2a shows an example where $S_1$, $S_2$, $S_3$, $S_4$, $S_5$ and $S_6$ are six sensors. The computed Voronoi cell of $S_1$ should be the polygon $V_1V_2V_3V_4V_5$. However, since $S_1$ is located outside the communication range of $S_2$ and is not aware of the existence of $S_2$, it computes its Voronoi cell as the polygon $V_1V_2V_3V_6$. This incorrect computation of Voronoi cells may lead to oscillatory movement in both Lloyd's method and DEED; thereby increases the traveling distance and prevents algorithms from converging. For example, in Lloyd's method, we can see that $S_1$ moves more than needed towards $S_2$ and may travel back after it hears from $S_2$.

To address the issue of distributed Voronoi cell computation, an intuitive algorithm is proposed as follows. This algorithm is applicable to both Lloyd's method (the original Lloyd's method and Lloyd-$\alpha$) and DEED. Let $R$ denote the communication ranges of sensors; a circle with radius $R/2$ and centered at a sensor is referred to as the "$R/2$-circle" of the sensor. Clearly, if two sensors are out of the communication range of each other, then their $R/2$-circles do not overlap. In the proposed method, each sensor uses the overlapped area of its computed Voronoi cell and its $R/2$-circle as its Voronoi cell. For example, the resultant Voronoi cell computed by $S_1$ in Fig. 2a is the polygon $V_1V_2V_3V_4'V_5'$ in Fig. 2b. In this special case, the resultant Voronoi cell is very close to the correct one. The intuition behind this algorithm is that the resultant Voronoi cell will not contain points that are in the Voronoi cells of other sensors. In this way, a sensor with no neighbors will stay at its position, while a sensor with incomplete neighbor information will take steps that tend to form an even deployment within the connected neighbors bounded by the $R/2$ circles of the sensor and its neighbors. Since there is no overlapped area between the $R/2$-circles of two sensors out of each other's communication range, the Hessian matrix $H$ and $\epsilon_k$ in Eq. (21) computed for DEED algorithm will be locally correct within the connected part of the network. As the sensors spread out and learn about new neighbors gradually, the global even deployment is finally achieved.




(a) Problem caused by limited communication range (b) Using $R/2$-circle to bound Voronoi cell

Fig. 2. An example showing incorrect Voronoi cell computation due to limited communication range of sensor $R$ and the proposed algorithm to deal with limited communication range.

To make sure that all sensors finally compute their correct Voronoi cells and form a global even deployment, it is assumed that in the final even deployment, the $R/2$-circles of all sensors cover the whole area. In practice, if the boundaries of the target area are known, then the required communication range $R$ of sensors can be computed a priori. Even if the boundaries are unknown, a larger area that contains the target area can be used to compute the value of $R$. The transmission power of the sensor can be adjusted to get the required $R$. In general, sensors may have different communication ranges, the value of $R$ here should be the minimum communication range of all sensors. For a sensor with anisotropic communication range, the value of $R$ should be chosen as the radius of the maximum circle within the communication range of the sensor.

One should note that in [3], an approach where sensor moves at most $R/2$ during movement is proposed to solve the same problem. The method we proposed employs the $R/2$-circle instead of limiting the step size is to approximate the actual Voronoi cell so that the accuracy of Voronoi related computation can be improved.

From the simulations it is observed that Lloyd method and DEED algorithm do not converge in a large number of deployment steps if the network formed by the initial deployment of the sensors is disconnected. When the above method is used to deal with the limited communication range, then both algorithms converge quickly.

## 6 Performance Evaluation

This section compares the performance of Lloyd's method and DEED algorithm through simulation in NS-2 [23] (version 2.35). The simulation settings are described first, followed by the simulation results.

### 6.1 Simulation Settings

The DEED and Lloyd's algorithms are implemented in two agent models. The mobile node model is used with minor modifications so that the agents can control the movement of sensors directly. Sensors are synchronized by scheduling the start time of their next step in every step.

The settings of the physical model follow IEEE 802.11p [24], the MAC protocol is IEEE 802.11. The sensors use omnidirectional antenna. The two ray ground model is used as the propagation model that considers reflection from the ground [25].

The computation of Voronoi partition is based on the "qvoronoi" program from Qhull [26]. Modifications have been made so that the program can compute Voronoi cells in a bounded region. At the end of the neighbor discovery phase, each sensor calls the program and uses its collected neighbor information as the input to the program. Thus, each sensor computes the Voronoi tessellation of the area from its local point of view. Subsequently, each sensor determines the intersection area of the computed Voronoi cell and its $R/2$-circle, which is used as the Voronoi cell in the computation of its movement step. To simplify the computation, the $R/2$ circle of each sensor is approximated by a regular hexagon.

#### 6.1.1 Stopping Criteria

In both algorithms, a stopping criteria is defined for the movement of sensors. If the distance between a sensor and the centroid of its Voronoi cell (i.e., the movement step of Lloyd's method) is less than 1m, then the sensor stops moving. Thus, when all sensors are close to the centroids of their Voronoi cells, i.e., close to a CVT, then they stop moving and the even deployment is completed. The reason for choosing the threshold of 1 m as the stopping criteria is two-fold. First, lowering the threshold further leads to an increasing number of deployment steps with little progress towards achieving a CVT. Second, based on the results obtained from the animation tool—network animator (NAM) [23], it is observed that with this criteria, the sensors visually form an even deployment. Furthermore, the deployment quality is visually much better than using a larger threshold, e.g., 2 m.

#### 6.1.2 Energy Consumption

The energy consumption statistics is accumulated over three sources—communication, movement, and start/stop operations (roughly equals to the number of deployment

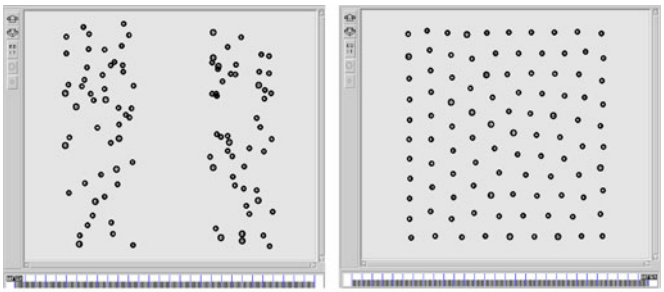



Fig. 3. Illustration of an initial and final deployment for the simulation scenario with 100 sensors (results captured in NAM). The final deployment is obtained using DEED.

steps). The unified energy consumption setting is used as that in [3]. The moving distance and the number of deployment steps are normalized into message complexity. Thus, the energy consumed by movement is presented by how many packets can be transmitted with the same amount of energy. As calculated from Robomote [11], moving a sensor one meter consumes a similar amount of energy as transmitting 300 messages. Thus, unless otherwise stated, the energy consumed by movement is defined as 300 messages/meter. The energy consumption in start/stop operations differs in different systems. In the unified energy consumption result as reported later, unless otherwise stated, the start/stop operation is equivalent to movement by one meter, i.e., 300 messages/step.

### 6.2 Simulation Scenarios

The simulations consist of two scenarios. The first scenario contains $100$ mobile sensors that have to be evenly deployed in a $500$ m $\times$ $500$ m area. In order to make sure that the $R/2$-circles of sensors cover the whole area when they form an even deployment, the communication ranges of sensors are set to $160$ m by adjusting the transmission power of the sensors. The area is divided into three stripes with the width of the middle stripe being $165$ m. The sensors are randomly distributed in the other two stripes. In this way, the initial deployment of the sensors forms a disconnected network topology. The initial deployment and desired deployment of the scenario are shown in Figs. 3a and 3b respectively, both of which are captured in the network animator from one of the obtained results.

The second scenario contains $25$ mobile sensors that needs to be evenly deployed in a $200$ m $\times$ $200$ m area. Similarly, the communication ranges of sensors are set to $90$ m. As before, three stripe areas are used for initial deployment and the width of the middle area is $95$ m.

For each scenario, all algorithms use the same initial locations and run one by one. The simulation time is sufficiently long (approximately the time of $200$ deployment steps) to make sure that all sensors achieve the steady state within the simulation time. The process is repeated for $100$ runs. In each run, we collect three statistics: the traveling distance, the number of deployment steps, and the energy consumption of sensors. These statistics are collected when all sensors stop.

Now we briefly discuss the choice of the lengths of the phases for the scenario that contains $100$ sensors (the same setting is used for the scenario that contains $25$ sensors). In the neighbor discovery phase, the possible packet loss is dealt with a simple mechanism. We let sensors send packets of their location information twice and wait a random time period before any transmission. Since the transmission time of one packet under the simulation setting is less than 1 ms, the length of the neighbor discovery phases is set to $10$ s, which is enough for all $100$ sensors to complete all transmissions. The speed of the sensor is set to be $1$m/s. The length of the movement phase is set to $750$ s so that all sensors can complete the movement before next iteration starts. In DEED, the Jacobi phase consists of three computation sub-phases. We let sensors send one packet in each computation subphase and set the length of the Jacobi phase to $10$ s. We choose multiple values for the positive constant $\delta$ in (21), and find that it has little impact on the simulation results. The results reported below use $\delta = 10$.

### 6.3 Simulation Results

For both simulation scenarios, the results are generated from $100$ simulation runs. Both DEED and Lloyd's method converge quickly when the method described in Section 5 is used to adjust Voronoi cells to deal with limited sensor communication range (when not using this method, both methods fail to converge within the given simulation time). All simulation results reported below use the adjusted Voronoi cells.

For the sake of clarity, the simulation runs are indexed in increasing order according to the average traveling distance under DEED algorithm.

Figs. 4, 5 and 6 presents the results generated from Scenario I with $100$ sensors in a $500$ m $\times$ $500$ m area.

Fig. 4a plots the average traveling distance over all sensors for each simulation run. The "Optimum" curve represents the average length of the optimal traveling path in Lloyd's method, i.e., the average linear distance between the initial locations and the final destinations in Lloyd's method. It is to be noticed that the traveling distance under DEED is closer to the optimum one. It results in $9$-$40$ percent less traveling distances as compared to Lloyd's method for all simulation runs with an average saving of $19$ percent. In comparison with Lloyd-$0.8$, the average distance saving from DEED is $13$ percent. Note that in some cases, DEED results in even less traveling distances than the optimum one. This is because the sensors converge to different destinations under DEED (recall that the CVT energy function can have multiple minima in the same area).

Fig. 4b plots the number of deployment steps for different algorithms. Observe that DEED requires similar numbers of deployment steps when compared to Lloyd's method and the average difference is small.

The performance of Lloyd-$\alpha$ method was also measured with respect to different step sizes $\alpha$. The results are plotted in Fig. 5 and are obtained by averaging over all simulation runs. Fig. 5a shows the average traveling distance versus the Lloyd step fraction value $\alpha$, while Fig. 5b shows the total number of deployment steps. As

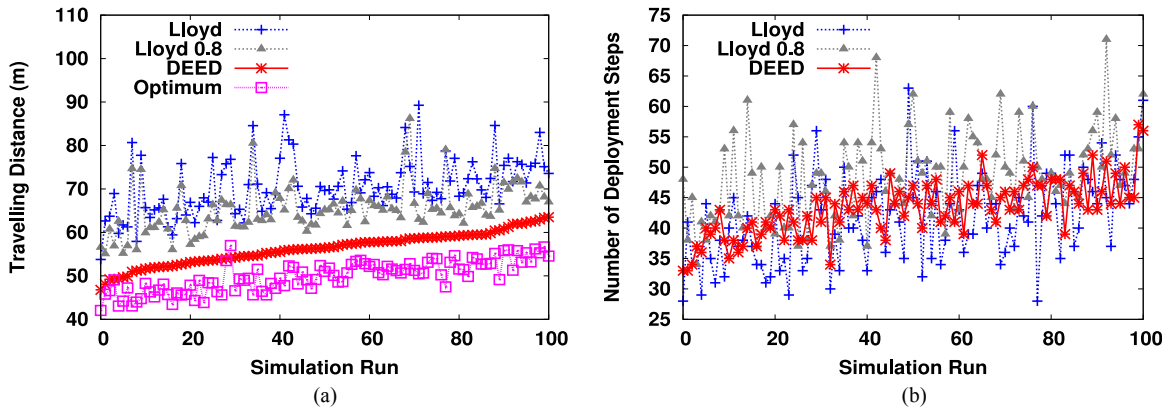



Fig. 4. Average travelling distance and number of deployment steps over 100 simulation runs for the scenario with 100 sensors.

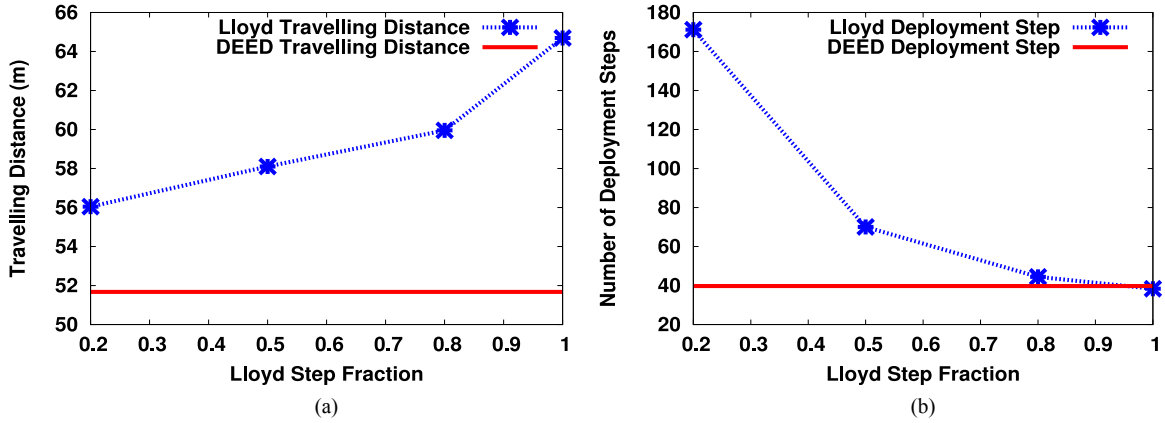



Fig. 5. Comparison of Lloyd-$\alpha$ methods with different step fractions $\alpha$.

expected, more distance savings are obtained for smaller step sizes. Lloyd-$\alpha$ algorithm requires approximately $1/\alpha$ times the number of steps than that of Lloyd's method. Among the tested Lloyd algorithms, Lloyd-0.8 is considered the most energy-efficient because it has the highest ratio of the distance saving over the increment in the number of deployment steps. For comparison, the average traveling distance and the number of deployment steps for DEED are also shown in Figs. 5a and 5b, respectively. The results indicate that DEED outperforms Lloyd's method and all its variants in both the average traveling distance and the number of deployment steps.

Fig. 6a plots the message complexity and Fig. 6b plots the average unified energy consumption over all sensors. As seen from Figs. 6a and 6b, DEED requires more message exchange compared to Lloyd's method and Lloyd-0.8 method due to the additional Jacobi phase; however, it still saves overall energy. This is due to the less traveling distance under DEED. Lloyd-0.8 algorithm consumes more energy on average than Lloyd's method due to larger

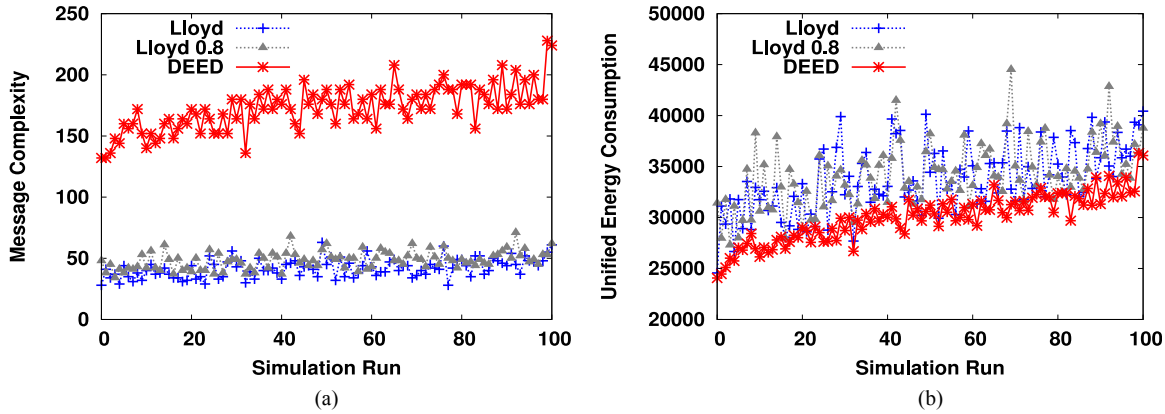



Fig. 6. Average message complexity and unified energy consumptions over 100 simulation runs for the scenario with 100 sensors (represented using the number of messages that can be transmitted).

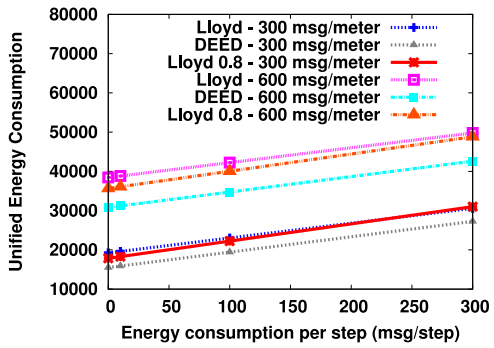



Fig. 7. Unified energy consumption under different energy consumption models.

number of deployment steps. When compared to Lloyd's method, the energy saving from DEED is up to 28 percent with an average saving of 13 percent. When compared to Lloyd-0.8, the average energy saving is 15 percent.

All the results presented so far assume that the energy consumption for moving a sensor one meter and the energy consumption per deployment step are both equivalent to the energy consumption of sending 300 messages. We next vary the energy consumption model as follows. The energy consumption for moving a sensor one meter is set to equivalence of sending either 300 or 600 messages; the energy consumption per deployment step is varied to be the equivalence of sending zero (zero energy consumption per deployment step) to 300 messages. The results are shown in Fig. 7. Clearly, DEED saves energy under all the settings. It is to be noticed that while Lloyd-0.8 consumes more energy than Lloyd's method under the default simulation settings, in other settings where the energy consumption of start/stop operation is relatively low, Lloyd-0.8 requires less energy consumption on average due to the distance savings when compared to Lloyd's method.

The results obtained in the second scenario with 25 sensors in 200 m $\times$ 200 m area are similar to those in the first scenario. We briefly summarize the result here. DEED algorithm performs the best overall. When compared to Lloyd's method, it saves up to 54 percent traveling distance with an average saving of 28 percent. The energy saving is up to 46 percent with an average saving of 18 percent.

We also explore different initial deployment settings, where sensors start from four corners of the target area, e.g., in the 500 m $\times$ 500 m target area, initial locations of sensors are set to be in four 100 m $\times$ 100 m squares located in the four corners of the area respectively. The obtained results are similar and thus are omitted in the interest of space.

## 7 Conclusion and Future Work

This paper studied the problem of energy-efficient even self-deployment in mobile sensor networks. In order to address the issue of energy-efficient deployment, which is still a challenge in the widely used Lloyd's method, a new algorithm, DEED algorithm, is proposed. Simulation results demonstrate that DEED performs well in different scenarios. Specifically, it leads to up to 54 percent less traveling distance and 46 percent less energy consumption than Lloyd's method. As future work, we will explore even self-deployment of sensors in areas with obstacles.

## Acknowledgments

This work was partially supported by US National Science Foundation CAREER award 0746841 and Qualtech Systems Incorporated. The authors would like to thank the anonymous reviewers for their insightful comments.

## References

[1] N. Heo and P.K. Varshney, "Energy-Efficient Deployment of Intelligent Mobile Sensor Networks," *IEEE Trans. Systems, Man and Cybernetics*, vol. 35, no. 1, pp. 78-92, Jan. 2005.

[2] Y. Zou and K. Chakrabarty, "Sensor Deployment and Target Localization Based on Virtual Forces," *Proc. IEEE INFOCOM*, pp. 1293-1303, July 2003.

[3] G. Wang, G. Cao, and T.F.L. Porta, "Movement-Assisted Sensor Deployment," *IEEE Trans. Mobile Computing*, vol. 5, no. 6, pp. 640-652, June 2006.

[4] B.L.A. Saipulla, C. Westphal, and J. Wang, "Barrier Coverage of Line-Based Deployed Wireless Sensor Networks," *Proc. IEEE INFOCOM*, pp. 127-135, 2009.

[5] X. Chu and H. Sethu, "A New Distributed Algorithm for Even Coverage and Improved Lifetime in a Sensor Network," *Proc. IEEE INFOCOM*, pp. 361-369, June 2009.

[6] A. Gersho, "Asymptotically Optimal Block Quantization," *IEEE Trans. Information Theory*, vol. 25, no. 4, pp. 373-380, July 1979.

[7] V.F.Q. Du and M. Gunzburger, "Centroidal Voronoi Tessellations: Applications and Algorithms," *SIAM Rev.*, vol. 41, no. 4, pp. 637-676, 1999.

[8] S. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. Information Theory*, vol. 28, no. 2, pp. 129-137, Mar. 1982.

[9] K.S.A. Okabe, B. Boots, and S.N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, second ed. John Wiley & Sons, 2000.

[10] J. Cortés, S. Martínez, T. Karatas, and B. Francesco, "Coverage Control for Mobile Sensing Networks," *IEEE Trans. Robotics and Automation*, vol. 20, no. 2, pp. 243-255, Apr. 2004.

[11] G.T. Sibley, M.H. Rahimi, and G.S. Sukhatme, "Robomote: A Tiny Mobile Robot Platform for Large-Scale Ad-Hoc Sensor Networks," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, pp. 1143-1148, May 2002.

[12] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang, "On Centroidal Voronoi Tessellation—Energy Smoothness and Fast Computation," *ACM Trans. Graphics*, vol. 28, no. 4, article 101 Aug. 2009.

[13] M. Iri, K. Murota, and T. Ohya, "A Fast Voronoi-Diagram Algorithm with Applications to Geographical Optimization Problems," *Proc. IFIP Conf. System Modelling and Optimzation*, pp. 273-288, 1984.

[14] Y. Asami, "A Note on the Derivation of the First and Second Derivatives of Objective Functions in Geographical Optimization Problems," *J. Faculty of Eng.*, vol. 61, no. 1, pp. 1-13, 1991.

[15] B.A. Bash and P.J. Desnoyers, "Exact Distributed Voronoi Cell Computation in Sensor Networks," *Proc. Sixth Int'l Smp. Information Processing in Sensor Networks (IPSN)*, 2007.

[16] R.T. Rockafellar, "Monotone Operators and the Proximal Point Algorithm," *SIAM J. Control and Optimization*, vol. 14, pp. 877-898, 1976.

[17] J. Nocedal and S. Wright, *Numerical Optimization*, second ed. Springer, 2006.

[18] R.B.S.J.E. Dennis Jr., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.

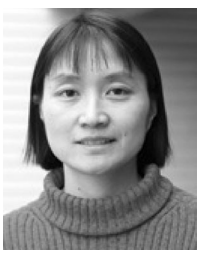
[19] R. Schnabel and E. Eskow, "A Revised Modified Cholesky Factorization Algorithm," *SIAM J. Optimization*, vol. 9, no. 4, pp. 1135-1148, 1999.

[20] D.M. Young and L.A. Hageman, *Applied Iterative Methods*. Academic Press, 1981.

[21] A. Jadbabaie, A. Ozdaglar, and M. Zargham, "A Distributed Newton Method for Network Optimization," *Proc. 48th IEEE Conf. Decision and Control*, pp. 2736-2741, Dec. 2009.

[22] O. Shental, P.H. Siegel, J.K. Wolf, D. Bickson, and D. Dolev, "Gaussian Belief Propagation Solver for Systems of Linear Equations," *Proc. IEEE Int'l Symp. Information Theory*, pp. 1863-1867, Aug. 2008.
[23] The Network Simulator Ns-2, http://www.isi.edu/nsnam/ns/, 2014.
[24] "IEEE P802.11p/D2.01, Draft Amendment for Wireless Access in Vehicular Environments (WAVE)," Feb. 2007.
[25] T.S. Rappaport, *Wireless Communications, Principles and Practice*. Prentice Hall PTR, Jan. 1996.
[26] Qhull, http://www.qhull.org, 2014.

**Yuan Song** received the BE and ME degrees both in telecommunications engineering from the Xidian University, in 2005 and 2008, respectively. He is currently working toward the PhD degree in the Department of Computer Science and Engineering, University of Connecticut. His research interests include wireless communication, network security and mobile computing.



**Bing Wang** received the BS degree in computer science from the Nanjing University of Science & Technology, China, in 1994, and the MS degree in computer engineering from the Institute of Computing Technology, Chinese Academy of Sciences in 1997. She then received the MS degrees in computer science and applied mathematics, and the PhD degree in computer science from the University of Massachusetts, Amherst, in 2000, 2004, and 2005, respectively. She is currently an associate professor of computer science and engineering at the University of Connecticut. Her research interests include computer networks, multimedia, and distributed systems. She received the US National Science Foundation (NSF) CAREER award in 2008. She is a member of the IEEE.

**Zhijie Shi** received the BS and MS degrees from Tsinghua University, China, in 1992 and 1996, respectively and the PhD degree from Princeton University in 2004. He is currently an associate professor of computer science and engineering at the University of Connecticut. He received the US National Science Foundation (NSF) CAREER award in 2006. His current research interests include underwater sensor networks, sensor network security, hardware mechanisms for secure and reliable computing, side channel attacks and countermeasures, and primitives for cipher designs. He is a member of the IEEE and the ACM.

**Krishna R. Pattipati** received the BTech degree in electrical engineering with highest honors from the Indian Institute of Technology, Kharagpur, in 1975, and the MS and PhD degrees in systems engineering from the University of Connecticut (UCONN), Storrs, in 1977 and 1980, respectively. He was with ALPHATECH, Inc., Burlington, Massachusetts from 1980 to 1986. He has been with the University of Connecticut, where he is currently the UTC Professor in Systems Engineering. His current research activities are in the areas of agile planning, diagnosis and prognosis techniques for cyber-physical systems, multi-object tracking, and combinatorial optimization. A common theme among these applications is that they are characterized by a great deal of uncertainty, complexity, and computational intractability. He is a cofounder of Qualtech Systems, Inc., a firm specializing in advanced integrated diagnostics software tools (TEAMS, TEAMS-RT, TEAMS-RDS, TEAMATE), and serves on the board of Aptima, Inc. He was selected by the IEEE Systems, Man, and Cybernetics (SMC) Society as the Outstanding Young Engineer of 1984, and received the Centennial Key to the Future award. He has served as the editor-in-chief of the *IEEE Trans. Systems, Man, and Cybernetics*–Part B from 1998 to 2001, vice-president for Technical Activities of the IEEE SMC Society from 1998 to 1999, and as vice-president for Conferences and Meetings of the IEEE SMC Society from 2000 to 2001. He was corecipient of the Andrew P. Sage Award for the Best SMC Transactions Paper for 1999, the Barry Carlton Award for the Best AES Transactions Paper for 2000, the 2002 and 2008 NASA Space Act Awards for "A Comprehensive Toolset for Model-based Health Monitoring and Diagnosis," and "Real-time Update of Fault-Test Dependencies of Dynamic Systems: A Comprehensive Toolset for Model-Based Health Monitoring and Diagnostics," and the 2003 AAUP Research Excellence Award at UCONN. He also won the best technical paper awards at the 1985, 1990, 1994, 2002, 2004, 2005, and 2011 IEEE AUTOTEST Conferences, and at the 1997 and 2004 Command and Control Conference. He is an elected fellow of the IEEE and of the Connecticut Academy of Science and Engineering.

**Shalabh Gupta** (M'04) received the MS degrees in mechanical and electrical engineering and the PhD degree in mechanical engineering in 2006 from Pennsylvania State University (Penn State), University Park. He is currently an assistant professor in the Department of Electrical and Computer Engineering at the University of Connecticut. His research efforts are directed toward opening new mathematical fields of data understanding, pattern discovery and adaptive decision-making, using multidisciplinary concepts derived from languages and automata theory, symbolic dynamics, and statistical mechanics. His research interests include the science of autonomy, swarm robotics, intelligent systems, machine learning, network science, and fault diagnosis & prognosis in complex systems. He is a member of the American Society of Mechanical Engineers (ASME) and the Institute of Electrical and Electronics Engineers (IEEE).